COMPUTER SOFTWARE APPLICATION

TRADE THEORY

NSQF LEVEL - 5

HANDBOOK FOR CRAFTS INSTRUCTOR TRAINING SCHEME

Directorate General of Training

DIRECTORATE GENERAL OF TRAINING
MINISTRY OF SKILL DEVELOPMENT & ENTREPRENEURSHIP
GOVERNMENT OF INDIA

NATIONAL INSTRUCTIONAL MEDIA INSTITUTE, CHENNAI

Post Box No. 3142, CTI Campus, Guindy, Chennai - 600 032

Published by

National Instructional Media Institute
Post.Box.No. 3142,
Guindy, Chennai - 600032
Email : chennai-nimi@nic.in
Website: www.nimi.gov.in

First Edition, 2024

Rs. 505/-

Printed in India at
National Instructional Media Institute
Post. Box. No. 3142,
Guindy,
Chennai - 600032

Disclaimer

The information contained herein has been obtained from sources reliable to Directorate General of Training, New Delhi. NIMI disclaims all warranties to the accuracy, completeness or adequacy of such information. NIMI shall have no liability for errors, omissions, or inadequacies in the information contained herein, or for interpretations thereof. Every effort has been made to trace the owners of the copyright material included in the book. The publishers would be greatfull for any omissions brought to their notice for acknowledgements in future editions of the book. No entity in NIMI shall be responsible for any loss whatsoever, sustained by any person who relies on this material. The material in this publication is copyrighted. No parts of this publication may be reproduced, stored or distributed in any form or by any means either on paper or electronic media, unless authorized by NIMI.

A Comprehensive Training Program
under Crafts Instructor Training Scheme (CITS)
for Instructors

HANDBOOK ON TECHNICAL INSTRUCTOR TRAINING MODULES

अतुल कुमार तिवारी, I.A.S.
सचिव

ATUL KUMAR TIWARI, I.A.S.
Secretary

सत्यमेव जयते

भारत सरकार
कौशल विकास एवं उद्यमिता मंत्रालय
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT
AND ENTREPRENEURSHIP

Foreword

In today's rapidly evolving world, the role of skilled craftsmen and women is more crucial than ever. The Craft Instructor Training Scheme (CITS) stands at the forefront of this transformation, shaping the educators who will train the next generation of artisans and technicians. This book aims to provide an in-depth understanding of the subject, exploring its significance, methodologies, and impact on vocational training.

The Craft Instructor Training Scheme was established with the objective of enhancing the quality of instruction in industrial training institutes and other vocational training institutions. By equipping instructors with advanced skills and knowledge, the scheme ensures that they are well-prepared to impart high-quality training to their students. This, in turn, contributes to the creation of a highly skilled workforce capable of meeting the demands of modern industry.

The initial chapters provide the importance of specialized instructor training. Following this, detailed chapters delve into the curriculum covering advanced techniques, safety protocols, and instructional strategies. Each section is designed to offer both theoretical insights and practical applications, ensuring a well-rounded understanding of the subject.

The book offers recommendations for overcoming obstacles and enhancing the effectiveness of the program, with the ultimate goal of producing highly skilled instructors capable of shaping the future workforce.

This book is intended for a diverse audience, including current and aspiring instructors, vocational training administrators, policymakers, and industry stakeholders. It serves as a valuable resource for understanding the intricacies of the subject and its pivotal role in vocational education.

I extend my heartfelt gratitude to all contributors who have shared their experiences and expertise, enriching this book with their valuable insights. Special thanks to the contribution of the development team, reviewers and NIMI that have supported this endeavor, providing essential data and resources.

It is my sincere hope that this book will inspire and guide readers in their efforts to enhance vocational training, ultimately contributing to the development of a skilled and competent workforce.

ATUL KUMAR TIWARI, I.A.S.
Secretary, MSDE

त्रिशलजीत सेठी
महानिदेशक
Trishaljit Sethi, IPoS
Director General

सत्यमेव जयते

भारत सरकार
कौशल विकास एवं उद्यमशीलता मंत्रालय
प्रशिक्षण महानिदेशालय
GOVERNMENT OF INDIA
MINISTRY OF SKILL DEVELOPMENT & ENTREPRENEURSHIP
DIRECTORATE GENERAL OF TRAINING

FOREWORD

The Craftsmen Training Scheme (CTS) implemented by the Directorate General of Training (DGT) provides skill training to the youth and ensures a steady flow of skilled manpower for the industry. It aims to raise quantitatively and qualitatively the industrial production by systematic training, and to reduce unemployment among the youth by providing them with employable skills.

The Craft Instructor Training Scheme (CITS) is an indispensable part of the Craftsmen Training Scheme (CTS). It offers comprehensive training both in 'skills' and in 'training methodology' to the instructor trainees to make them conversant with techniques of transferring hands-on skills.

I congratulate NIMI for taking the initiative of preparation of the course content for CITS. This will help institutionalize the mechanism for imparting training to the trainers all across the ecosystem. I also extend my gratitude to the Instructors and Officials of National Skill Training Institutes (NSTIs) and the DGT for their invaluable contribution in preparation of the CITS course content.

As we navigate the complexities of a rapidly changing world and the technological disruptions, the significance of CTS and CITS has increased manifold. It not only empowers individuals with practical skills but also lays the foundation for a prosperous future. I am confident that this book will serve as a guiding light to all instructor trainees for skill development and nation-building.

Trishaljit

(Trishaljit Sethi)

PREFACE

The Craft Instructor Training Scheme is an indispensable module of the Craftsmen Training Scheme, which has been an integral part of the Indian skill development industry since its inception. This program aims to equip instructors with the necessary skills and teaching methodology to effectively transfer hands-on skills to trainees and promote a holistic learning experience. The first Craft Instructor Training Institute was established in 1948, followed by six more institutes across India in 1960. Today, these institutes, including the National Skill Training Institute (formerly Central Training Institute for Instructors), offer the CITS course, which is mandated by the Directorate General of Training (DGT).

The Craft Instructor training program is designed to develop skilled manpower for industries. The course aims to offer instructors an opportunity to improve their instructional skills, engage learners effectively, offer impactful mentoring, and make efficient use of resources, leading to a more skilled workforce in various industries. The program emphasizes collaborative and innovative approaches to teaching, resulting in high-quality course delivery. Overall, the Craft Instructor Training Scheme is a pivotal program that helps instructors grow in their careers and make a significant contribution to society. This program is essential for developing skilled manpower and promoting a robust learning environment that benefits both trainees and instructors alike.

ACKNOWLEDGEMENT

National Instructional Media Institute (NIMI) sincerely acknowledges with thanks for the co-operation and contribution extended by the following experts to bring out this Instructional material **(Trade Theory)** for **CITS Computer Software Application (NSQF Level - 5)** under the **IT&ITES** Sector for **Instructors**.

MEDIA DEVELOPMENT COMMITTEE MEMBERS

Shri. Hemant Kumar Singh	-	Training Officer, NSTI, Kanpur.

COORDINATORS

Shri. G.C. Ramamurthy	-	Joint Director, CD - Section, DGT.
Shri. T.V. Rajasekar	-	Joint Director, NIMI, Chennai.
Shri. Shiv Kumar	-	Training Officer, CD - Section, DGT.

NIMI records its appreciation of the Data Entry, CAD, DTP Operators for their excellent and devoted services in the process of development of this Instructional Material.

NIMI also acknowledges with thanks, the invaluable efforts rendered by all other staff who have contributed for the development of this Instructional Material.

NIMI is grateful to all others who have directly or indirectly helped in developing this IMP.

ABOUT THE TEXT BOOK

The Vocational Instructor Training Program is a comprehensive initiative designed to equip aspiring students with the necessary skills and knowledge to effectively teach in vocational education settings. This program encompasses a range of pedagogical strategies, instructional techniques, and subject-specific content tailored to the diverse vocational fields. Participants engage in coursework that covers curriculum development, assessment methods, classroom management, and the integration of industry-relevant technologies. Practical experience and hands-on training are emphasized, allowing participants to apply theoretical concepts in real-world teaching environments. Through collaborative learning experiences and mentorship opportunities, aspiring vocational instructors develop the confidence and competence to facilitate engaging and impactful learning experiences for their students. This training program aims to cultivate a new generation of educators who are not only proficient in their respective vocational fields but also adept at fostering the success and employability of their students in today's competitive workforce.

This text book covers communication, self-management, information and communication technology, entrepreneurial and green skills. It has been developed as per the learning outcome-based curriculum.

G C Rama Murthy,
Joint Director,
Curriculum Development, DGT,
MSDE, New Delhi.

CONTENT

MODULE 1 : Network Architecture

LESSON 01 - 17 : Network Architecture

Objectives

At the end of this lesson, you will be able to:

- set up LAN & configure various network devices
- manage various network applications
- secure network & handle various cyber security issues

Layering & Protocols

Layering, in the context of networking and communication systems, refers to the organization of complex systems into multiple, hierarchical layers, each with specific functions and responsibilities. This concept is essential for modular design and interoperability. The most common layered model in networking is the OSI

Sewing Machines, their models, parts and their functions

Layer	Name	Protocols
7	Application	SMTP, HTTP, FTP, POP3, SNMP
6	Presentation	MPEG, ASCH, SSL, TLS
5	Session	NetBIOS, SAP
4	Transport	TCP, UDP
3	Network	IPV5, IPV6, IPSEC, ICMP, ARP, MPLS.
2	Data Link	RAPA, PPP, ATM,Frame Relay, Fiber Cable, etc.
1	Physical	RS232, 100BaseTX, ISDN, 11.

(Open Systems Interconnection) model, which has seven layers:

Protocols

Protocols are a set of rules and conventions that determine how data is exchanged and communicated between devices or systems.

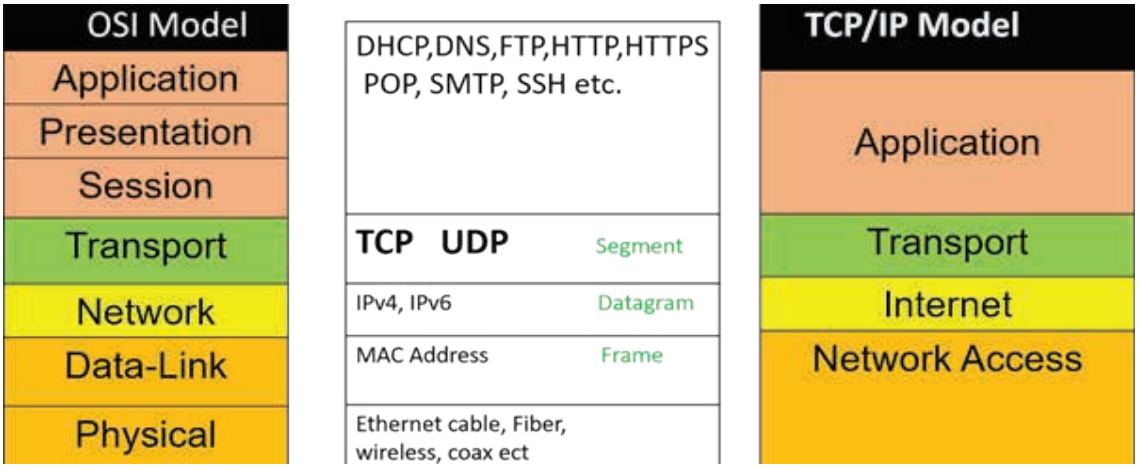
They provide a standardized way for various entities to understand and interact with each other

Protocols operate at specific layers of the network stack, ensuring that each layer's functionality is well-defined and can operate independently of the layers above and Below

Internet Protocols

HTTP (Hypertext Transfer Protocol) - Used for transmitting web pages and other resources over the World Wide Web.

HTTPS (Hypertext Transfer Protocol Secure) - An encrypted version of HTTP, ensuring secure communication for activities such as online banking and e-commerce.



TCP / IP (Transmission Control Protocol / Internet Protocol) - The backbone of the internet and most networks, TCP/IP provides a reliable and connection-oriented method for data transmission. It includes protocols like TCP (for reliable data delivery) and IP (for routing and addressing).

Network Protocols

Ethernet - A protocol that defines how data is placed over a physical network medium, such as via wired connections.

IPv4 and IPv6 - Internet Protocol versions 4 and 6, respectively, used to route data packets across networks.

Communication Protocols

SMTP (Simple Mail Transfer Protocol): Used for sending and receiving email.

POP3 (Post Office Protocol 3) and IMAP (Internet Message Access Protocol): Protocols used by e-mail clients to retrieve messages from mail servers.

File Transfer Protocols

- **FTP (File Transfer Protocol):** Used for transferring files between a client and a server on a computer network.

Application Layer Protocols

- **DNS (Domain Name System)** - Converts human-readable domain names into IP addresses to facilitate internet navigation.
- **SNMP (Simple Network Management Protocol)** - Used for managing and monitoring network devices and their performance.

Wireless Protocols

- **Bluetooth** - A short-range wireless protocol used for connecting devices such as smartphones, keyboards, and headphones.
- **Wi-Fi** - A protocol for wireless local area networking, enabling devices to connect to the Internet and other devices within a specified area.

Security Protocols

- **TLS/SSL (Transport Layer Security / Secure Sockets Layer)** - Protocols that provide secure communication over a computer network, commonly used for web browsing.
- **Physical Layer Protocols:** Examples include Ethernet and Wi-Fi standards, which define how data is transmitted over physical media.
- **Data Link Layer Protocols:** Ethernet and Wi-Fi also operate at this layer, along with protocols like ARP (Address Resolution Protocol) for mapping IP addresses to MAC addresses.
- **Network Layer Protocols:** Internet Protocol (IP) is a prominent example, used for addressing and routing packets across networks. Additionally, ICMP (Internet Control Message Protocol) handles network management tasks like ping and traceroute.

- **Transport Layer Protocols:** TCP (Transmission Control Protocol) ensures reliable, connection-oriented communication, while UDP (User Datagram Protocol) offers connectionless, lightweight communication.
- **Application Layer Protocols:** These include HTTP for web browsing, SMTP for email, and FTP for file transfer, among many others.

OSI & Internet Architecture

OSI Model

OSI stands for Open Systems Interconnection. It has been developed by ISO International Organization for Standardization in the year 1984.

In the OSI reference model, the communications between a computing system are split into seven different abstraction layers: Physical, Data Link, Network, Transport, Session, Presentation, and Application.

Layer			Protocol data unit (PDU)	Function
Host layers	7	**Application**	Data	High-level protocols such as for resource sharing or remote file access, e.g.**HTTP**
	6	**Presentation**		Translation of data between a networking service and an application; including character encoding, **data compression** and **encryption/decryption**
	5	**Session**		Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	**Transport**	Segment, Datagram	Reliable transmission of data segments between points on a network, including **segmentation, acknowledgement** and **multiplexing**
Media layers	3	**Network**	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	**Data link**	Frame	Transmission of data frames between two nodes connected by a physical layer
	1	**Physical**	Bit, Symbol	Transmission and reception of raw bit streams over a physical medium

1 Physical Layer

- It is the lowest layer of the OSI model.
- It is responsible for the actual physical connection between the devices.
- This layer converts the digital bits into electrical, optically or via radio waves.

Some Functions of Physical layer

Data Transmission - The various transmission modes possible are Simplex, full-duplex and half-duplex.

Line Configuration - It helps in providing Physical Medium and Interface decisions

Signals - It determines the type of the signal used for transmitting the information.

Topology - It helps in Physical Topology (Mesh, Star, Bus, Ring) decisions (Topology through which we can connect the devices with each other)

Physical Layer devices - are Modem, Hub, Repeater and Cables.

2 Data Link Layer(DDL)

The data link layer is responsible for the node-to-node delivery of the message.

- The data link provides a efficient & reliable communication between two or more devices.
- Defines the format of the data on the network.
- It is the responsibility of the data link layer to transmit it to the Host using its MAC address.
- It contains two sub-layers

Logical Link Control Layer(LLC)

- LLC is responsible for transferring the packets to the Network layer of the receiver that is receiving.
- It also provides flow control, responsible for multiplexing, acknowledgement and even error-checking functions of DDL

Media Access Control Layer(MAC)

- A MAC (Media Access Control) layer is a link between the LLC (Logical Link Control) layer and the network's physical layer.
- Used for transferring the packets over the network.

Some Functions of Data-Link Layer

Framing - Framing is a function of the data link layer. it is adds a header to the frame that contains a destination address. The frame is transmitted to the destination address mentioned in the header.

Flow Control - Flow control is the main functionality of the data-link layer. This prevents a fast sender from overwhelming a slower receiver by using techniques like sliding window protocols.

Physical Addressing - These addresses are used to identify the source and destination devices within the same local network. The Data Link Layer assigns unique addresses.

Error Control - It checks for errors that might occur during transmission using techniques like checksums or cyclic redundancy checks (CRC). If errors are detected, the frame can be retransmitted.

Access Control - When two or more devices are connected to the same communication channel, the MAC sub-layer of the data link layer helps to determine which device has control over the channel at a given time.

Switch & Bridge are Data Link Layer devices

3 Network Layer

- The Network Layer deals with logical addressing, routing, and forwarding of data packets between different networks.
- It also takes care of packet routing.
- It's layer is responsible for routing and forwarding the packets.
- It is used to route the network traffic are known as Network layer protocols.

Some Functions of Network Layer

- Logical Addressing-A network layer adds the source and destination address to the header of the frame. Addressing is used to identify the device over the internet. these addresses are hierarchical and provide a way to uniquely identify devices across different networks.
- Routing-The Network Layer is responsible for determining the best path that data packets. This involves analysing network topology, congestion, and other factors to make efficient routing decisions.

4 Transport Layer

- The main responsibility of the transport layer is to transfer the data as a whole.
- it is ensures that messages are transmitted in the order in which they are sent and there is no duplication of data.
- It is responsible for determining the optimal path for data packets to travel from the source to the destination across interconnected networks.
- The Internet Protocol (IP) operates at this layer.
- The transport layer is called as Heart of the OSI model.
- Protocol Use : TCP, UDP NetBIOS, PPTP.

The two protocols used in this layer are

1 Transmission Control Protocol (TCP)

TCP is a connection-oriented protocol that offers reliable, ordered, and error-checked data delivery.

2 User Datagram Protocol (UDP)

UDP is a connectionless protocol that provides a lightweight way of sending data between devices without the overhead of establishing a connection and ensuring reliable delivery.

Some Functions of Transport Layer

- **Segmentation and Reassembly** - When the transport layer receives a message from the upper layer, it divides the message into several segments, and each segment is assigned a sequence number that uniquely identifies each segment. When the message reaches the destination, the transport layer reassembles the message based on their sequence numbers.
- **Service Point Addressing** - Computers run many programs simultaneously, so the transfer of data from source to destination is not only from one computer to another but also from one process to another. The transport layer adds a header that includes the address known as the service-point address or port address. The responsibility of the network layer is to transmit data from one computer to another and the responsibility of the transport layer is to transmit the message to the right process.

5 Session Layer

- The session layer is used to establish, maintain, and synchronous interaction between communicating devices.

Some Functions of Session Layer

Dialog Controller - The session layer allows the two systems to begin communication with each other in half-duplex or full-duplex.

Synchronization - The session layer adds some checkpoints when transmitting the data in a sequence. If some error occurs in the middle of the data transmission, then the transmission will take place again from the checkpoint. This process is known as synchronization and recovery.

6 Presentation Layer

- A presentation layer is primarily concerned with the syntax and semantics of the information exchanged between two systems.
- its layer also handles the encryption and decryption that the application layer requires.
- It deals with data format conversion, encryption, and data compression.

Some Functions of Presentation Layer

Data Translation - data format conversions between the sender's and receiver's systems For example, it could translate between ASCII and EBCDIC character encoding.

Encryption/ Decryption - Data encryption translates data into another form or code.

Protocol Use : JPEG, MPEG, GIF

7 Application Layer

- Application layer serves as a window for users and application processes to access network services.
- Web browsers and other internet-connected apps, like file transfer, email, remote access, and more.
- protocols at the application layer, known as HTTP, FTP, SMB/CIFS, TFTP, and SMTP.

Some Functions of Application Layer

File transfer, access, and management (FTAM) - File transfer access and management : This application allows a user to access file in a remote host, retrieve files in remote host and manage Controlling files from a remote computer.

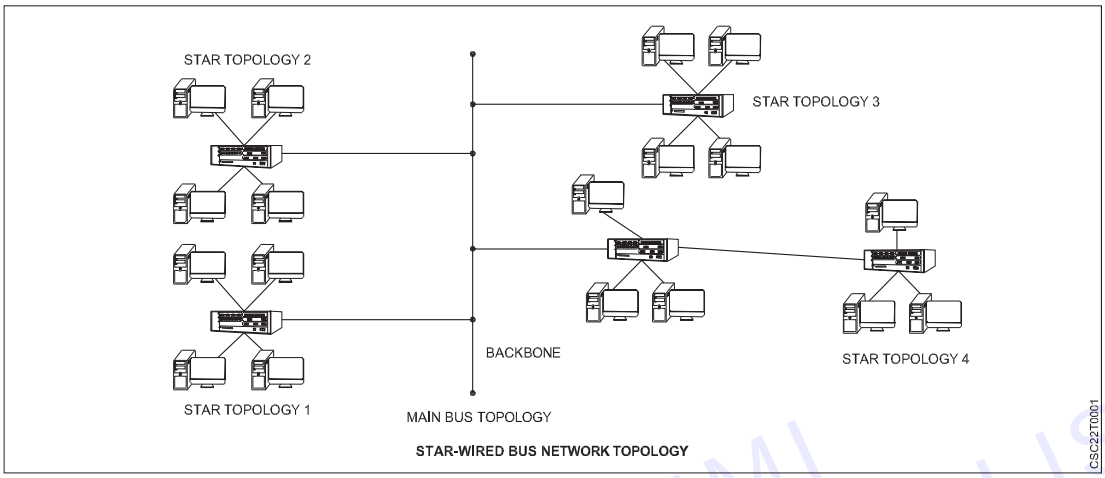
Network Topology

A topological space is a set endowed with a structure, called a topology, that allows defining continuous deformation of subspaces, and, more generally, all kinds of continuity. Euclidean spaces, and, more generally, metric spaces, are examples of a topological space, as any distance or metric defines a topology.

1 **Bus Topology** - In a bus topology, all devices are connected to a single central cable, which serves as the communication channel.

Data is transmitted along this cable and received by all devices on the network.

This topology is less common today due to its limitations in terms of scalability and fault tolerance.



Materials Needed:

1 Coaxial cable or twisted-pair cable (for modern implementations).

2 Network devices (computers, printers, etc.).

3 Terminators (to prevent signal reflections).

Steps to Set Up a Bus Topology:

- **Choose a Central Cable:** Select a suitable cable to serve as the central bus. In older implementations, coaxial cable (such as RG-58) was commonly used. However, in more modern setups, twisted-pair cables (such as Ethernet cables) are used.
- **Install Network Interface Cards (NICs):** Ensure that each device you want to connect to the network has a network interface card (NIC) installed. NICs allow devices to connect to the network and transmit/receive data.
- **Connect Devices:** Connect each device to the central cable using T-connectors or similar connectors. One end of the T-connector attaches to the central cable, and the other end connects to the device's NIC. Repeat this step for all devices on the network.
- **Use Terminators:** At each end of the central cable, install terminators to prevent signal reflections. Signal reflections can cause network interference and degradation of the signal quality. Terminators are typically 50-ohm resistors that absorb the signal.
- **Test the Network:** After connecting all devices and ensuring that terminators are in place, power on the devices and test the network. You should be able to transmit data between devices on the network.

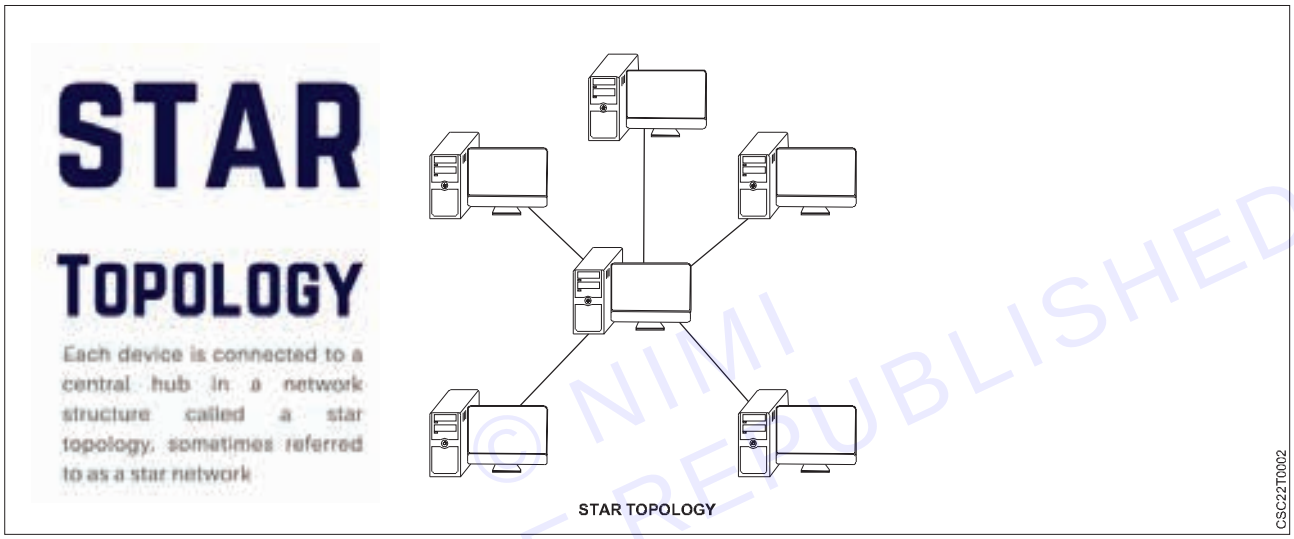
Advantages of Bus Topology:

- Simplicity: Bus topologies are straightforward to set up and understand.
- Cost-Effective: They require less cabling compared to some other topologies, making them cost-effective for small networks.

Disadvantages of Bus Topology:

- **Single Point of Failure:** If the central cable fails or gets damaged, the entire network can be disrupted.
- **Limited Scalability:** Adding more devices to a bus network can lead to signal degradation and performance issues.
- **Security Concerns:** Since all devices share the same cable, it can be easier for unauthorized users to tap into the network.
- **Performance Issues:** Bus topologies can suffer from signal collisions, especially as the number of devices increases, leading to performance degradation.

2 **Star Topology** - In a star topology, all the devices are connected to a central hub or switch. Each device communicates directly with the central hub, which manages and controls the network traffic. While this topology is easy to set up and manage, the entire network may be affected if the central hub fails.



Scenario: Setting up a Local Area Network (LAN) using a Star Topology Requirements:

- Central switch or hub
- Multiple devices (computers, printers, etc.)

Steps to set up a Star Topology:

1 **Select a Central Hub/ Switch:** Choose a central device that will serve as the hub or switch for your network. This device should have enough ports to accommodate all the devices you plan to connect.

2 **Connect Devices to the Central Hub/ Switch:** Use Ethernet cables to connect each device to one of the available ports on the central hub or switch. These devices can include computers, printers, and any other networked equipment. Each device has its cable running directly to the central hub.

3 **Configure Network Settings:** Configure the network settings on each connected device. This includes setting IP addresses, subnet masks, and any other network-specific configurations. You may also set up DHCP (Dynamic Host Configuration Protocol) on the central hub if you want it to automatically assign IP addresses to connected devices.

4 **Testing and Troubleshooting:** Test the network connections to ensure that all devices can communicate with each other. Troubleshoot any connectivity issues by checking cables, configurations, and the central hub's status.

Advantages of a Star Topology:

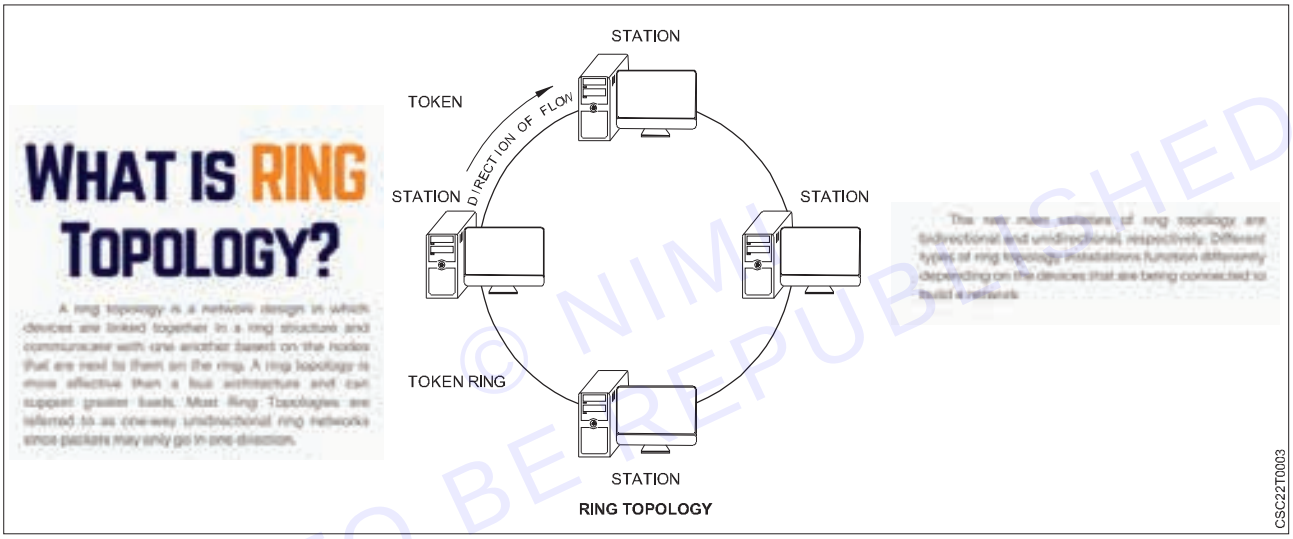
- **Easy to manage:** Each device connects directly to the central hub, making it simple to add or remove devices without disrupting the entire network.

- **Scalability:** You can easily expand the network by adding more devices or ports to the central hub as needed.
- **Fault Isolation:** If one device or cable fails, it doesn't affect the rest of the network. Only the malfunctioning device or cable needs to be addressed.

Disadvantages of a Star Topology:

- **Single Point of Failure:** If the central hub or switch fails, the entire network may become inaccessible. Redundancy measures can mitigate this risk.
- **Cost:** Setting up a star topology can be more expensive than some other topologies, as it requires a central hub with enough ports for all devices.
- **Cable Length:** The maximum cable length between a device and the central hub is limited, which may be a constraint in large networks.

3 **Ring Topology** - A ring topology is a type of network topology in which each device is connected to exactly two other devices, creating a closed loop or ring. Data travels in a unidirectional or bidirectional manner around the ring until it reaches its intended destination. While ring topologies are less common than other topologies like bus or star, they have their own advantages and use cases. Here's how you can use a ring topology:



1 Physical Setup:

To create a ring topology, you'll need to physically connect your devices in a ring-like fashion. This can be done with Ethernet cables, fiber optic cables, or wireless connections, depending on your network requirements and the available infrastructure.

2 Redundancy:

One of the key advantages of a ring topology is redundancy. If a cable or device fails, the data can still travel in the opposite direction around the ring to reach its destination. This inherent redundancy can help maintain network reliability.

3 Configuration:

Configure your network devices accordingly. In a ring topology, each device should be aware of the devices immediately before and after it in the ring. This ensures that data packets are forwarded in the correct direction.

4 Data Transmission:

Data packets travel around the ring, passing through each device until they reach their destination. Devices examine the destination address of each packet and determine whether to forward it to the next device or retain it.

5 Token Passing (Optional):

In some ring networks, a token-passing protocol is used to manage access to the network. Devices take turns sending data by passing a token around the ring. Only the device holding the token can transmit data, which helps avoid collisions and ensures orderly data transmission.

6 Monitoring and Maintenance:

Regularly monitor the network for any issues, such as cable faults or device failures. Ring topologies are known for their fault tolerance, but it's still important to address any problems promptly to maintain network performance.

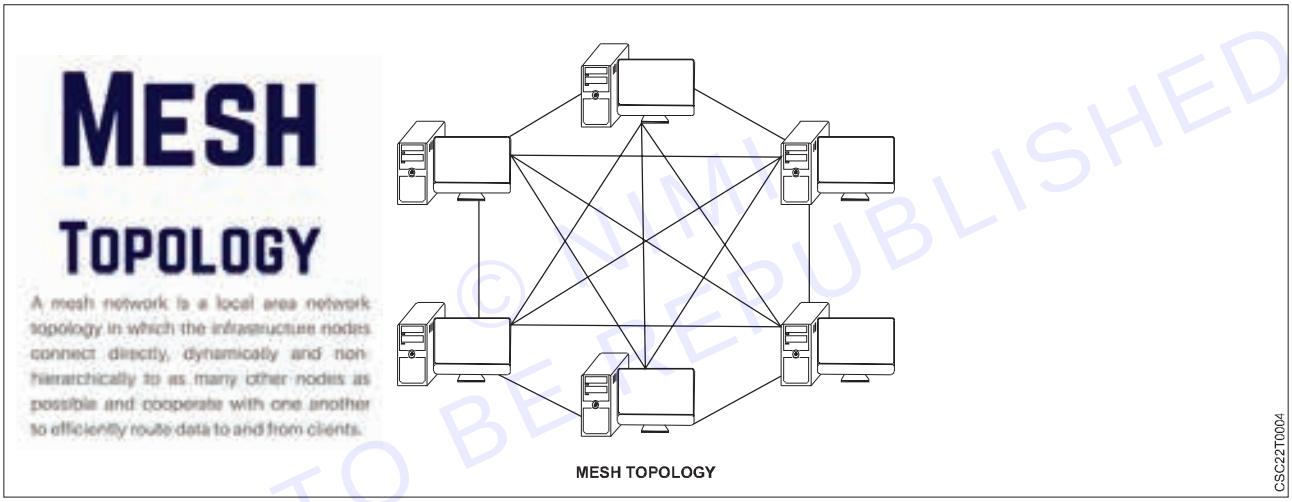
Use Cases:

Ring topologies are well-suited for certain scenarios, such as:

- **Fiber Optic Networks:** Fiber optic rings are commonly used for high-speed, long-distance data transmission due to their reliability and fault tolerance.
- **Industrial Control Systems:** Ring topologies are used in industrial environments where continuous operation is critical, as they can provide redundancy and fault tolerance.
- **Token Ring Networks:** Token ring networks, a specific type of ring topology, were once popular in early LANs, though they have largely been replaced by Ethernet networks.

4 **Mesh Topology** - Mesh topology involves connecting each device to every other device in the network.

- It provides high redundancy and fault tolerance as multiple paths exist for data transmission.
- However, it can be complicated to implement and requires more cabling and resources.



MESH TOPOLOGY

Scenario: Imagine you're tasked with designing a computer network for a medium-sized company that requires a high level of reliability and redundancy. You decide to implement a mesh network topology to ensure seamless communication even if some network links or devices fail.

Here's how you can use a mesh topology for this network:

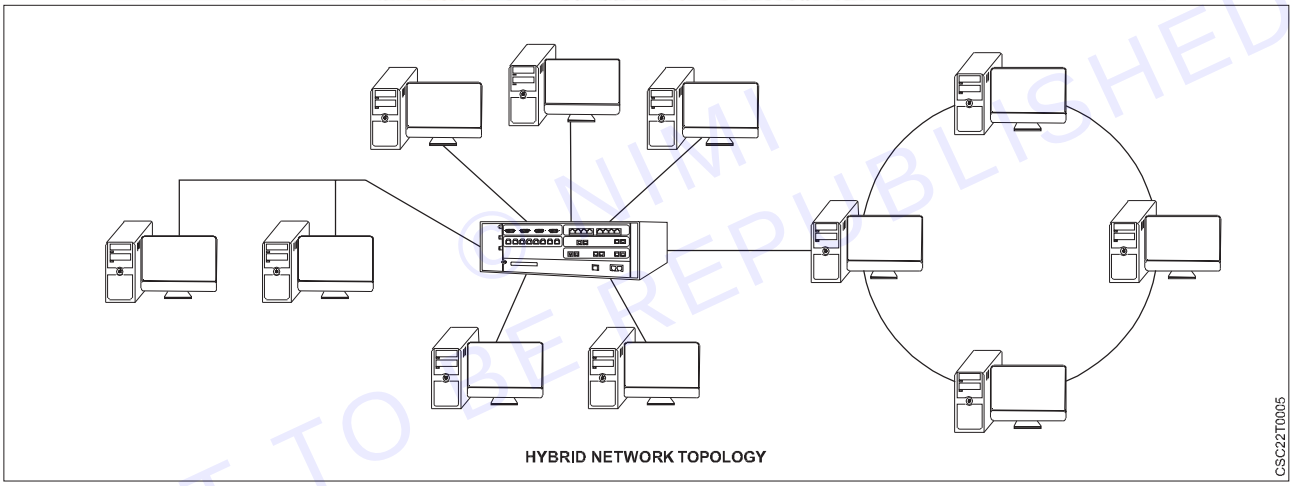
- **Identify the Devices:** First, identify all the devices that need to be connected in the network. These may include computers, servers, printers, and network switches.
- **Calculate the Number of Connections:** In a full mesh topology, every device connects to every other device. To calculate the number of connections needed, you can use the formula:

  Number of connections = (n * (n - 1)) / 2

  Where 'n' is the number of devices. This will give you the total number of connections required.
- **Install Network Cables:** Physically install network cables to connect each device to every other device in the network. Use appropriate networking hardware like switches or routers to facilitate these connections.
- **Configure Network Devices:** Configure the network devices (switches and routers) to enable communication between all devices. Set up routing tables and ensure that data can flow through multiple paths, providing redundancy.

- **Implement Redundancy:** Mesh topology inherently provides redundancy, but you can further enhance it by using redundant network links and devices. If one link or device fails, traffic can automatically reroute through an alternate path.
- **Testing and Monitoring:** Thoroughly test the network to ensure that all devices can communicate with each other. Implement network monitoring tools to keep an eye on the network's performance and detect any issues.
- **Security Measures:** Implement appropriate security measures, such as firewalls, access controls, and encryption, to protect the network from unauthorized access and data breaches.
- **Documentation:** Maintain detailed documentation of the network layout, including the connections, IP addresses, and configurations. This documentation is essential for troubleshooting and future expansion.
- **Regular Maintenance:** Schedule regular maintenance and updates to keep the network running smoothly. Perform routine checks for hardware failures or potential bottlenecks.

5 **Hybrid Topology** - Hybrid topology is a combination of two or more basic topologies



HYBRID NETWORK TOPOLOGY

This approach permits organizations to tailor their network design to meet specific needs. For instance, a combination of star and mesh topologies could provide both centralized control and redundancy. example of how you might use a hybrid topology in a network:

Let's say you're setting up a network for a medium-sized company that has multiple departments, each with different connectivity needs. You want to balance cost-effectiveness, redundancy, and ease of management. In this scenario, a hybrid topology could be ideal:

- **Core Network:** Start with a backbone network that follows a ring topology. This core network connects all the major departments and serves as the main data highway. A ring topology offers redundancy; if one link fails, data can still flow through the other path.
- **Departmental Networks:** Each department can have its own local area network (LAN) with a star topology. In a star topology, each device connects directly to a central hub (like a switch). This makes it easy to manage and expand each department's network independently.
- **Critical Servers:** Place critical servers, such as file servers or database servers, in a mesh topology within the core network. Mesh topology provides fault tolerance and redundancy, ensuring that if one server goes down, the network can still access the services through alternate paths.
- **Wireless Access Points:** For mobile devices and guest access, use wireless access points distributed throughout the building, following a mesh or star topology. This ensures comprehensive wireless coverage with redundancy.

- **Remote Offices:** If the company has remote offices or branches, consider connecting them to the core network through secure VPN connections. This can create a hub-and-spoke topology, with the core network being the hub and remote offices as spokes.
- **Internet Connectivity:** Connect the entire network to the internet through a dedicated firewall and router. This connection can be implemented in a bus or star topology, depending on your specific needs for scalability and redundancy.

By using a hybrid topology in this way, you can create a network that is cost-effective, easily scalable, and provides redundancy where needed. It allows you to tailor the network design to the specific requirements of different parts of the organization while ensuring reliable and efficient data flow across the entire infrastructure.

Link & Medium Access protocols, IEEE 802 standards, Performance issues

A link protocol, also known as a data link protocol or link layer protocol, operates on the data link layer (Layer 2) of the OSI model.

Its primary functions include framing, error detection and correction, flow control and addressing.

Link protocols are responsible for packaging higher-level data into frames that can be transmitted over a physical medium.

They also handle acknowledgements, retransmissions, and other mechanisms to ensure reliable data transfers between directly connected nodes.

Common examples of link protocols include

Ethernet - A widely used wired link protocol that uses MAC addresses to identify devices on a network and employs CSMA/CD (Carrier Sense Multiple Access with Collision Detection) for medium access control.

HDLC (High-Level Data Link Control) - A synchronous link protocol used primarily in point-to-point and multipoint communication networks.

PPP (Point-to-Point Protocol) - A protocol used for establishing a direct connection between two nodes over various physical medium.

Medium Access Control (MAC) Protocol

MAC protocols are a subset of link protocols.

They deal specifically with the access to and control of a shared communication medium, such as a wired or wireless channel.

MAC protocols determine how devices on a network compete for the right to transmit data in order to minimize collisions and ensure fair and efficient access to the medium.

CSMA / CD (Carrier Sense Multiple Access with Collision Detection) - Used in Ethernet networks, devices listen for carrier signals before transmitting. If a collision is detected, the devices back off and try again after a random time.

CSMA / CA (Carrier Sense Multiple Access with Collision Avoidance) - Used in wireless networks to avoid collisions. Devices sense the channel and wait for it to be cleared before transmitting to prevent simultaneous transmissions.

Token passing - In networks using token passing protocols like Token Ring, a special token is passed between devices, granting the holder the right to transmit. This ensures an orderly access to the medium.

TDMA (Time Division Multiple Access) - The available transmission time is divided into time slots, and each device is assigned a specific time slot for transmission.

FDMA (Frequency Division Multiple Access) - Different devices are assigned different frequency bands within a shared medium.

CDMA (Code Division Multiple Access) - Each device uses a unique code to transmit data simultaneously over the same frequency band, and receivers use the corresponding code to distinguish between signals.

IEEE

It seems like you've mentioned "IEEE," which stands for the Institute of Electrical and Electronics Engineers. IEEE is a professional organization that develops and publishes standards for a wide range of industries, including electronics, electrical engineering, telecommunications, and computer networking. In the context of networking, IEEE standards play a crucial role in ensuring compatibility, interoperability, and efficient communication between devices and systems.

Performance issues in network architecture can significantly impact a network's efficiency, reliability, and user experience.

These issues may arise from various factors and may have far-reaching consequences. Here are some common performance issues in network architecture.

Bandwidth Limitations - Insufficient bandwidth can lead to slower data transfer rates, causing delays and bottlenecks in network traffic. This issue is especially relevant when there is a large volume of data to be transmitted or when dealing with multimedia content.

Latency - Latency refers to the delay between sending a data packet and receiving a response. High latency can lead to sluggish performance, particularly in real-time applications like video conferencing, online gaming, and VoIP (Voice over Internet Protocol) calls.

Packet Loss: Packet loss happens when data packets fail to reach their intended destination. This can happen due to network congestion, faulty hardware, or other issues. It can result in data retransmissions and degraded application performance.

Jitter - Jitter is the variation in latency leading to irregular delays in packet arrival. In real-time applications, consistent latency is vital, and jitter can cause disruptions and poor quality.

Network Congestion - When a network experiences heavy traffic, it can become congested, causing delays and packet loss. Network congestion can occur due to insufficient bandwidth, improper network design, or sudden spikes in usage.

Network Adaptors. Circuit switching - packet switching

Network Adapters

A network adapter, also known as a network interface card (NIC) or network interface controller, is a hardware component that allows computers, servers or other devices to connect to a network.

It provides the required physical interface for the device to transmit and receive data over a network.

Network adapters typically have a unique identifier called a MAC (Media Access Control) address, which is used to distinguish devices on a network.

They can be integrated into a computer's motherboard or added as an expansion card.

Here are some key points about network adapters:

1 **Purpose:** Network adapters are essential for connecting devices to a network, whether it's a local area network (LAN), a wide area network (WAN), or the internet. They are used in computers, servers, routers, switches, and various other networked devices.

2 **Types of Network Adapters:**

- **Ethernet NIC:** This is the most common type of network adapter and is used for wired Ethernet connections. It typically has an RJ-45 port for connecting to Ethernet cables.
- **Wireless NIC:** These adapters are used for wireless connections and are often integrated into laptops and mobile devices. They connect to Wi-Fi networks.
- **Fiber Optic NIC:** These are specialized network adapters designed for high-speed fiber optic connections, commonly used in data centers and high-performance computing environments.

3 **Functions:** Network adapters perform several important functions, including:

- **Data Link Layer Processing:** They handle the framing, addressing, and error-checking of data packets at the data link layer (Layer 2 of the OSI model).
- **Media Access Control (MAC) Address:** Each network adapter has a unique MAC address, which is used to identify it on the network.
- **Packet Transmission and Reception:** Network adapters transmit data onto the network and receive incoming data, allowing devices to communicate with each other.
- **Driver and Software Support:** To function properly, network adapters require drivers and software that enable communication between the hardware and the operating system.

4 **Installation:** In most cases, network adapters are installed inside a computer as a separate hardware component, either as an expansion card or integrated into the motherboard. Wireless NICs can also be in the form of USB dongles that plug into a USB port.

5 **Configuration:** Network adapters often require configuration settings, such as IP addresses, subnet masks, and gateway addresses, to properly communicate on a network. These settings can be configured manually or obtained automatically via protocols like DHCP (Dynamic Host Configuration Protocol).

6 **Duplex Modes:** Network adapters support different duplex modes, including full-duplex and half-duplex. Full-duplex allows simultaneous two-way communication, while half-duplex permits communication in one direction at a time.

7 **Virtualization:** In virtualized environments, virtual network adapters (virtual NICs) are created for virtual machines (VMs). These virtual adapters connect VMs to virtual networks, allowing them to communicate within the virtualized infrastructure.

8 **Network Speed and Standards:** Network adapters support various speeds and standards, such as Fast Ethernet (100 Mbps), Gigabit Ethernet (1 Gbps), and 10 Gigabit Ethernet (10 Gbps). The specific speed depends on the hardware and the network infrastructure.

Circuit Switching

Circuit switching is a method of communication in which a dedicated communication path or circuit is established between two devices for the duration of their conversation.

This circuit remains active even if no actual data is being transmitted, resulting in continuous connection.

Traditional telephone networks are the classic example of circuit switching.

Once a connection is established, the devices can exchange data without the need for addressing or routing during the call.

This method guarantees constant bandwidth but can be inefficient when the circuit is tied up for the entire duration of the call, even if there are pauses in the conversation.

Following key features:

- **Dedicated Circuit:** When two parties want to communicate, a dedicated physical connection is established between them. This connection remains reserved exclusively for their use throughout the conversation.
- **Resource Reservation:** Before communication begins, the network allocates resources, such as bandwidth and a dedicated communication path, to the established circuit. This resource reservation ensures a consistent and guaranteed quality of service for the entire duration of the communication.
- **Connection Establishment:** Circuit switching involves a three-phase process: circuit establishment, data transfer, and circuit teardown. During the circuit establishment phase, the network sets up the connection, including determining the path through which data will flow.
- **Constant Bandwidth:** Since the circuit is dedicated to the conversation, the bandwidth is constant and not shared with other users. This ensures predictable and steady data transfer rates, making circuit switching suitable for voice calls and real-time applications.
- **Example Usage:** Traditional telephone networks, such as the Public Switched Telephone Network (PSTN), rely heavily on circuit switching. When you make a phone call, a dedicated circuit is established between your phone and the recipient's phone for the duration of the call.

- **Inefficiency:** Circuit switching is less efficient for data communication compared to packet switching because the dedicated circuit is reserved even when there is silence or no data transmission. This inefficiency makes it less suitable for data services like internet browsing.
- **Scalability Challenges:** Circuit switching can be challenging to scale as the number of users and communication sessions grows. It requires significant infrastructure to maintain dedicated circuits for all potential connections.
- **Robustness:** Circuit-switched networks are generally robust and provide high call quality since the dedicated circuit ensures a constant, reliable connection.

Message Switching:

Message switching is a method of data communication where complete messages or data units are transmitted as a whole from the source to the destination. Unlike packet switching, which breaks data into smaller packets for transmission, message switching sends entire messages from one point to another. Here are some key characteristics of message switching:

- **Whole Message Transmission:** In message switching, the entire message is sent as a single unit. This message could be a text message, a file, or any other data unit.
- **Store-and-Forward:** Message switching typically involves a store-and-forward mechanism. The message is received at an intermediate node (message switch) and stored temporarily before being forwarded to the next hop. This intermediate storage allows for some degree of buffering and error handling.
- **Connectionless:** Message switching is connectionless, meaning that there is no dedicated or established path between the sender and receiver before sending the message. Each message is handled individually and can take different routes to reach its destination.
- **Variable Delivery Times:** Since messages can take different paths and may be temporarily stored at intermediate nodes, delivery times for messages in message switching networks can vary. Some messages may be delivered quickly, while others may experience delays.
- **Less Efficient:** Message switching is generally less efficient than packet switching, especially when it comes to utilizing network resources. This is because it sends entire messages even if they are relatively small, leading to less efficient use of bandwidth.
- **Historical Significance:** Message switching was one of the earliest forms of data communication used in telegraph and telex systems. It predates modern computer networking technologies like packet switching and was prevalent during the early days of long-distance communication.

Packet Switching:

Packet Switching, on the other hand, is a more efficient and flexible method of communication commonly used in modern computer networks, including the Internet.

Data is divided into smaller packets, each containing a portion of the data, together with source and destination addresses.

These packets are then individually routed through the network based on the current network conditions and available routes.

This allows for more efficient use of network resources, as different packets can take different routes and be interleaved over the same communication lines. Packet switching also permits multiple conversations (sessions) to share the same physical network infrastructure simultaneously.

Key characteristics of packet switching include:

- **Dividing Data:** When data is sent across a network using packet switching, it is broken down into small packets. These packets are typically a few hundred bytes in size.
- **Routing:** Each packet is treated independently and can take a different route to reach its destination. Routers and switches within the network make decisions about how to forward each packet based on its destination address.
- **Efficiency:** Packet switching is efficient because it allows multiple devices to share a network's resources simultaneously. It avoids the need for dedicated communication paths between sender and receiver, as is the case in circuit switching.

- **Robustness:** If a link or node in the network fails, packet-switched networks can often reroute packets through alternative paths, making them more resilient to network failures.
- **Scalability:** Packet switching is highly scalable, making it suitable for networks of various sizes, from small local area networks (LANs) to global-scale networks like the internet.
- **Common Protocols:** Common networking protocols that use packet switching include the Internet Protocol (IP) for the internet and Ethernet for LANs.

Internetworking - bridges - Internet protocol - Addressing- Routing Protocols

Internetworking

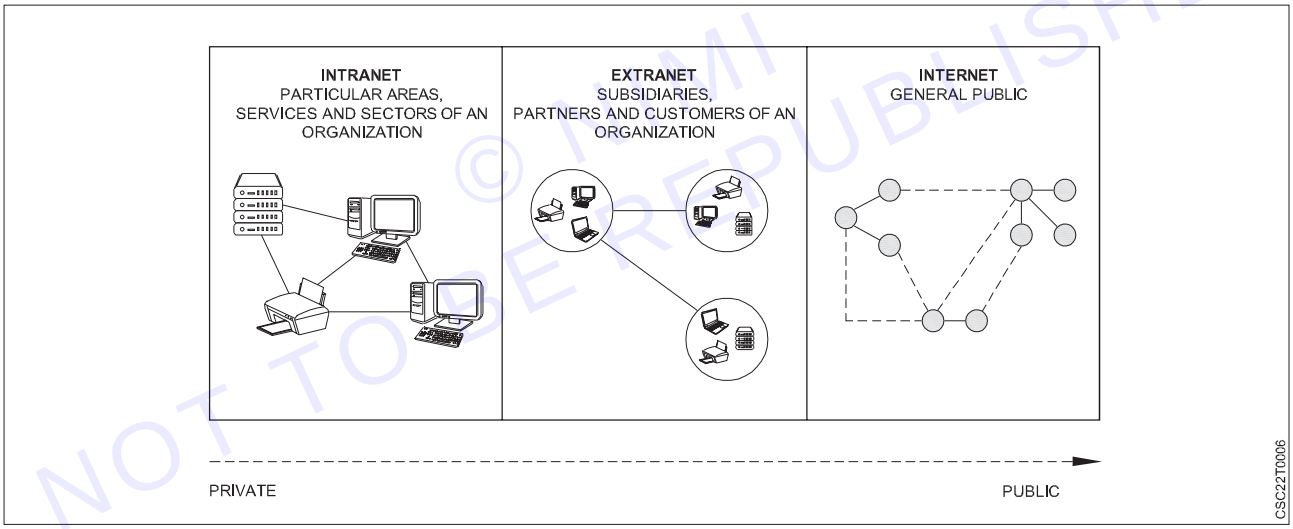
Internetworking is the practice of interconnecting multiple computer networks, such that any pair of hosts in the connected networks can exchange messages irrespective of their hardware-level networking technology. The resulting system of interconnected networks are called an internetwork, or simply an internet.

There is chiefly 3 units of Internetworking

1 Extranet

2 Intranet

3 Internet



An Extranet is a private network that extends some of an organization's internal network resources and services to external users or organizations, typically on the internet.

It's essentially a controlled and secure extension of an organization's Intranet (internal network) to include specific external parties such as customers, suppliers, business partners, or other authorized users.

Key characteristics of an Extranet include:

- **Selective Access:** Extranets are designed to provide access only to authorized users, allowing organizations to share specific information, collaborate on projects, or conduct transactions securely with external parties.
- **Security:** Security measures, such as encryption, firewalls, and user authentication, are in place to protect the confidentiality and integrity of data shared over the Extranet. This is essential to ensure that sensitive information remains secure.

- **Collaboration:** Extranets facilitate collaboration and communication between an organization and its external partners. This can include sharing documents, project management, joint planning, and more.
- **Shared Resources:** Organizations often use Extranets to share resources like databases, applications, and project management tools with external entities. This can streamline operations and improve efficiency.
- **Customization:** Extranets can be customized to meet the specific needs of the organization and its external partners. Access permissions, available resources, and user interfaces can be tailored accordingly.
- **Remote Access:** Since Extranets are typically accessed over the internet, authorized users can connect from remote locations, making it convenient for collaboration with external parties who may be geographically dispersed.
- **Examples:** Examples of Extranets include customer portals where customers can access their account information and place orders, supplier portals where suppliers can check inventory levels and submit purchase orders, and partner collaboration platforms for joint projects.

Intranet

An Intranet is a private, internal network within an organization that uses internet-based technologies and protocols but is restricted to authorized users, typically employees and sometimes trusted partners or suppliers.

It serves as a secure platform for sharing information, resources, and communication among members of the organization.

Here are some key characteristics and purposes of an Intranet:

- **Internal Use:** Intranets are designed for use within an organization. They are not accessible to the general public or the broader internet.
- **Secure Environment:** Intranets employ various security measures like firewalls, user authentication, and encryption to protect sensitive data and ensure that unauthorized users cannot access the network.
- **Information Sharing:** Intranets serve as a central platform for sharing company-specific information such as policies, procedures, news, announcements, and internal documents.
- **Communication:** Intranets often include communication tools such as email, instant messaging, discussion forums, and collaboration software to facilitate internal communication and collaboration among employees.
- **Resource Access:** Employees can access resources like shared files, databases, company directories, and applications through the Intranet. This streamlines workflow and makes it easier to access necessary tools.
- **Knowledge Management:** Intranets are often used to store and manage knowledge resources, such as documents, manuals, training materials, and employee directories.
- **Corporate Culture:** They can be a platform for promoting and reinforcing an organization's corporate culture, values, and mission through internal communications and engagement initiatives.
- **Cost-Efficiency:** Intranets can reduce the need for physical paperwork, streamline processes, and improve efficiency, resulting in cost savings for the organization.
- **Customization:** Organizations can tailor the Intranet to their specific needs, creating custom web applications, portals, and interfaces that suit their business requirements.
- **Scalability:** Intranets can grow and evolve with the organization, accommodating increasing user numbers and expanding resources as needed.

Internet

The internet, often simply referred to as the "Internet," is a global network of interconnected computer networks that spans the entire planet. It is a vast and decentralized network that connects billions of devices and computers worldwide, allowing them to communicate, share information, and access a wide range of services and resources.

Key characteristics of the internet include:

- **Global Connectivity:** The internet connects individuals, organizations, and devices from all corners of the world. It is not limited by geographic boundaries, making it a truly global network.

- **Decentralization:** The internet is not controlled by a single entity or organization. Instead, it is made up of a multitude of interconnected networks, each managed by various entities, including internet service providers (ISPs), companies, and governments.
- **Protocols:** The internet relies on a set of standardized communication protocols, such as TCP/IP (Transmission Control Protocol/Internet Protocol), which enable devices to exchange data packets seamlessly.
- **Access via Web Browsers:** Most users access the internet through web browsers like Google Chrome, Mozilla Firefox, or Microsoft Edge. The World Wide Web (WWW) is a significant part of the internet, allowing users to access websites and web-based applications.
- **Information and Services:** The internet offers a vast array of resources and services, including websites, email, social media, online shopping, streaming media, search engines, cloud computing, and much more.
- **Communication:** The internet facilitates various forms of communication, such as email, instant messaging, video conferencing, and social networking, connecting people across the globe.
- **Openness:** The internet is built on the principles of openness and accessibility, allowing anyone with an internet connection to access information and contribute to it.
- **Security and Privacy:** While the internet provides valuable services, it also raises concerns about security and privacy. Users and organizations must take measures to protect their data and online activities.
- **Evolution:** The internet is constantly evolving, with new technologies, standards, and services regularly emerging. This ongoing evolution has transformed the way people work, communicate, and access information.

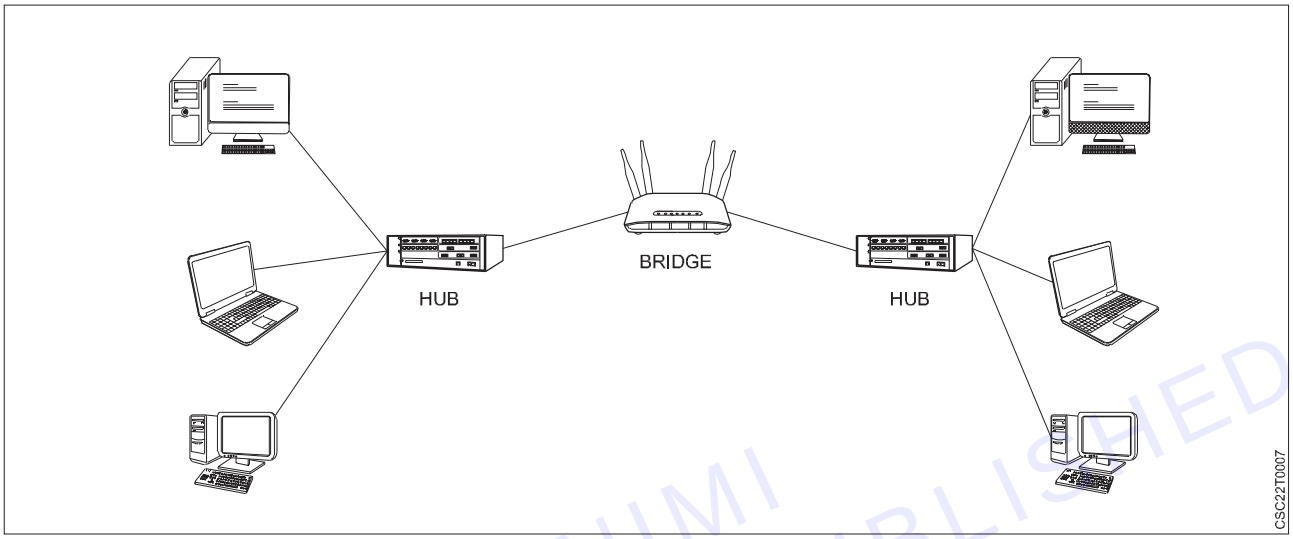
Aspect	Internet	Intranet	Extranet
Accessibility	Public access over the global network	Private network for internal use only	Partially private network with limited external access
Users	Anyone with internet access	Restricted to organization employees	Combination of internal and external users
Purpose	Information sharing, global access	Internal communication and collaboration	Collaborative communication with external partners
Security	Limited control, more security risks	Greater control, higher security measures	Controlled access, security measures in place
Content	Public websites, diverse content	Company-specific information and tools	Shared company resources and selected information
Authentication	Typically username and password	User authentication for employees	User authentication for both internal and external users
Examples	Google, Facebook, Wikipedia	Corporate intranet, internal company tools	Customer portals, supplier collaboration platforms

Bridge

A network bridge is a computer networking device that creates a single, aggregate network from multiple communication networks or network segments.

This function is called network bridging. Bridging is distinct from routing.

A bridge within a computer network is a hardware device employed to link numerous Local Area Networks (LANs) into a larger unified LAN. This process of merging networks is referred to as bridging. These bridges are physical devices that function at the data link layer of the OSI model and are sometimes referred to as switches operating at the second layer.



There are three primary types of bridges in computer networks:

- **Transparent Bridge:** This type of bridge operates inconspicuously on the network, filtering traffic based on MAC addresses. Its purpose is to extend network coverage and segment LANs seamlessly.
- **Source Routing Bridge:** A source routing bridge relies on the sender specifying the route for data frames through the network. The bridge simply follows the designated route as instructed.
- **Translational Bridge:** The translational bridge serves as a bridge with the additional capability of translating between different network protocols or formats. It facilitates communication between networks that use distinct protocols or data formats.

Advantages of bridges in computer networks

- Bridges are capable of extending networks by connecting two different network topologies.
- They establish separate collision domains, leading to enhanced bandwidth utilization.
- Bridges can serve as a buffer when various MAC protocols are employed on different network segments.
- They offer high reliability and ease of maintenance, allowing the network to be divided into multiple LAN segments.
- Bridges are straightforward to install, requiring no additional hardware or software aside from the bridge itself.
- They exhibit a higher level of protocol transparency when compared to other networking protocols.

Disadvantages

- Higher cost compared to hubs and repeaters.
- Slower data transfer speeds.
- Reduced performance due to the need for extra processing to identify device MAC addresses on the network.
- Inability to perform individual data filtering because it deals with bulk or broadcasted traffic.
- Elevated broadcast traffic during data broadcasting, which may result in the formation of broadcast storms within the network.

Internet Protocol

It is a protocol defined in the TCP/IP model used for sending the packets from source to destination

The Internet Protocol (IP) is a fundamental communication protocol used in computer networks, including the global network that we know as the Internet.

These protocols work together to enable data transmission and communication across connected networks.

Addressing

An IP address is a unique address that identifies a device on the internet or a local network. IP stands for "Internet Protocol," which is the set of rules governing the format of data sent via the internet or local network.

The first IP was IPv4 that was commercially used. IPv4 was entirely exhausted by the internet users and internet service providers. Thus to satisfy the ever-increasing need of IP Addresses, Internet Engineering Task Force (IETF) came up with the new IPv6 in 1995, standardized in 1996. At present both IPv4 and IPv6 are in use, and both are entirely different from each other regarding providing addresses.

IPv4 uses a 32-bit address format (e.g., 192.168.1.1), while IPv6 uses a 128-bit address format (e.g., 2001:0db8 :85a3:0000:0000:8a2e:0370:7334).

Routing

A routing protocol is a set of rules and algorithms used by routers in a network to determine the best path for forwarding data packets from the source to the destination. These protocols enable efficient communication between different devices within a network by dynamically adapting to changes in network topology, such as link failures or new connections.

Routing protocols are particularly important in larger networks, where there can be multiple paths between the source and destination.

The primary goal of routing protocols is to find the most optimal path for data transmission based on factors such as shortest path, available bandwidth, latency, and reliability.

The routing algorithm initializes and maintains the routing table for the process of path determination. Here are some key aspects of routing protocols:

UDP - TCP- Congestion Control - Presentation aspects

- It provides connectionless service and end-to-end delivery of transmission.
- It is an unreliable protocol as it discovers the errors but not specify the error.
- User Datagram Protocol discovers the error, and ICMP protocol reports the error to the sender that user datagram has been damaged.
- UDP consists of the following fields:

Source port address: The source port address is the address of the application program that has created the message.

Destination port address: The destination port address is the address of the application program that receives the message.

Total length: It defines the total number of bytes of the user datagram in bytes.

Checksum: The checksum is a 16-bit field used in error detection.

- UDP does not specify which packet is lost. UDP contains only checksum; it does not contain any ID of a data segment.

Transmission Control Protocol (TCP)

- It provides a full transport layer services to applications.
- It creates a virtual circuit between the sender and receiver, and it is active for the duration of the transmission.
- TCP is a reliable protocol as it detects the error and retransmits the damaged frames. Therefore, it ensures all the segments must be received and acknowledged before the transmission is considered to be completed and a virtual circuit is discarded.

- At the sending end, TCP divides the whole message into smaller units known as segment, and each segment contains a sequence number which is required for reordering the frames to form an original message.
- At the receiving end, TCP collects all the segments and reorders them based on sequence numbers

Congestion Control

Congestion control in computer networks is a set of techniques and strategies used to manage and relieve network congestion, ensuring that the network operates efficiently and effectively even during periods of high traffic.

Congestion can happen when the demand for network resources, such as bandwidth and processing capacity, exceeds the available supply, leading to degraded performance, increased latency, and packet loss.

Here are some key aspects of congestion control in network environment.

1 Traffic Policing and Shaping

- Traffic Policing- Network devices, such as routers and switches, can enforce traffic limits by discarding or marking packets that exceed specified rate limits. This prevents excessive traffic from entering the network.
- Traffic Shaping Rather than discarding excessive traffic, traffic shaping smooths out traffic bursts by buffering and releasing packets at a controlled rate. This helps in avoiding sudden spikes in network congestion.

2 Quality of Service (QoS)

- QoS mechanisms prioritize certain types of traffic over others based on predefined rules. This ensures that critical or real-time traffic (e.g., VoIP, video conferencing) receives better treatment than less time-sensitive traffic.

3 Window-Based Congestion Control (TCP)

- In the Transmission Control Protocol (TCP), which is widely used for reliable data transmission, window-based congestion control mechanisms adjust the rate at which a sender transmits data based on feedback from the network.
- TCP uses techniques like the Slow Start, Congestion Avoidance, and Fast Recovery algorithms to dynamically adapt the sender's transmission rate to the network's congestion level.

4 Explicit Congestion Notification (ECN):

- ECN allows routers to mark packets as they pass through congested areas of the network. The sender receives this feedback and can adjust its transmission rate accordingly, avoiding further congestion.

5 Random Early Detection (RED) and Active Queue Management (AQM):

- RED is a queue management algorithm that helps routers manage congestion by dropping or marking packets when the queue length exceeds a certain threshold. This encourages sources to reduce their sending rates.
- AQM extends RED by actively managing the queue length to maintain an optimal balance between low latency and high throughput.

6 Load Balancing:

- Distributing incoming network traffic across multiple paths or resources helps prevent any single point of congestion. Load balancers ensure that no individual component becomes overwhelmed.

7 Multipath Routing

- Using multiple paths for data transmission can help avoid congestion on a single path. Multipath routing algorithms dynamically select paths based on current network conditions.

8 Congestion-Aware Routing:

- Some routing algorithms take congestion into account when selecting paths for data transmission. They avoid routes with high congestion and prefer paths with lower traffic loads.

9 Feedback Mechanisms:

- Congestion control often relies on feedback from routers, switches, and endpoints to adjust transmission rates and routing decisions.

10 Network Monitoring and Measurement:

- Monitoring tools continuously assess network performance and congestion levels. This data is crucial for making informed decisions about congestion control strategies.

Telnet, FTP - e-mail - DNS

Congestion control is a complicated and ongoing challenge in network engineering.

Effective congestion control mechanisms ensure that networks can handle varying levels of traffic while providing a consistent, reliable experience for users and applications.

TELNET

Telnet (Telecommunication Network) is a protocol used for remotely accessing and managing devices over a network, typically the internet. It allows a user to establish a text-based connection to a remote server or device and interact with it as if they were directly connected to the device's console

Telnet operates on the Application Layer of the OSI model and uses a client-server architecture.

The client application (telnet client) initiates a connection to the remote server (telnet server) using the Telnet protocol. Once the connection is established, the user can send text commands to the server and receive text-based responses.

Telnet sessions are typically used for tasks such as remote administration, configuration, and troubleshooting of devices such as routers, switches, servers, and other network equipment.

Advantages of Telnet

It provides remote access to one's computer system.

Telnet allows the user to have more access with less problems in data transmission.

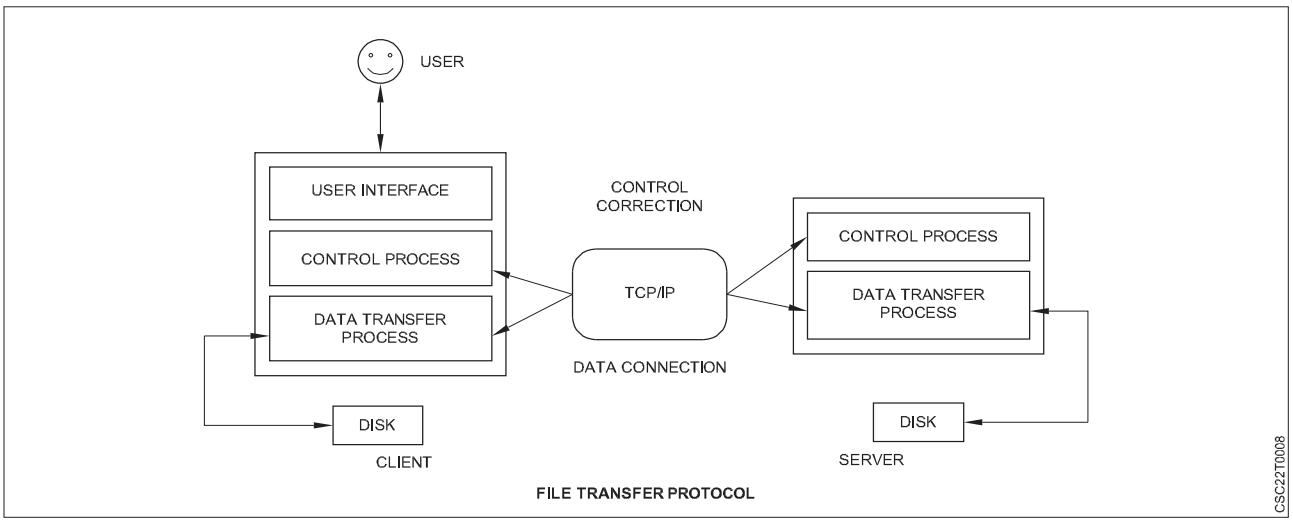
telnet saves a lot of time.

The oldest system can be connected to a newer system with different operating systems with telnet.

Disadvantages of Telnet

- As it is somewhat complex, it becomes difficult for beginners to understand.
- Some capabilities are disabled because of not proper interlinking of the remote and local devices.
- Data is sent here in plain text form, that's why it is not so secure.

FTP(File transfer protocol)



- File Transfer Protocol (FTP) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network.
- FTP is built upon a client–server model architecture using separate control and data connections between the client and the server

The main purpose of FTP is to facilitate the uploading and downloading of files between computers. It's often used for tasks such as

- Uploading website files to a web server.
- Downloading software updates from a remote server.
- Sharing files between users on a network.
- Backing up files to a remote server.

Advantages of FTP

Speed - One of the biggest benefits of FTP is speed. The FTP is one of the fastest way to transfer files from one computer to another.

Efficient: It is more efficient as we do not need to complete all the operations to get the whole file.

Security -To access the FTP server, we need to login with username and password. So we can say that FTP is more secure.

Back & forward movement - FTP allows us to transfer the files back and forth. Suppose you are a manager of the company, you send some information to all the employees, and all of them send information back on the same server.

Disadvantages of FTP

- File size limit is the drawback of FTP only 2GB size files can be transferred.
- Multiple receivers are not supported by the FTP.
- FTP does not encrypt the data this is one of the biggest drawbacks of FTP.
- FTP is unsecured, we use login IDs and passwords making it secure but they can be attacked by hackers.

E-mail

A way of sending electronic messages or data from one computer to another.

A worldwide e-mail network allows people to exchange e-mail messages very quickly

Email, short for "electronic mail," is a method of exchanging digital messages over the Internet.

It allows people to send and receive messages and files to and from each other using electronic devices such as computers, smartphones, and tablets.

Email has become one of the most common and widely used forms of communication both in personal and professional contexts.

DNS

DNS stands for Domain Name System, and it is a fundamental technology used on the Internet to translate human-readable domain names into IP addresses.

Computers and servers communicate with one another using IP addresses, which are numerical identifiers for devices on a network.

However, remembering and typing the IP addresses for every website you want to visit would be impractical for humans.

DNS acts as a distributed directory system that allows users to access websites and other online resources using easily memorable domain names (like www.example.com) rather than numerical IP addresses.

Here is how it works:

1 **User Input** - When you enter a domain name (eg, www.example.com) into your web browser, your device needs to know the IP address associated with that domain to establish a connection.

2 **DNS Query** - Your device sends a DNS query to a DNS resolver (typically provided by your Internet Service Provider or a third-party service), asking for the IP address of the domain.

3 **DNS Resolving Process** - The DNS resolver doesn't have the IP address cached in most cases, so it begins a process to find the IP. It first checks its cache to see if it has recently resolved this domain. If not, it proceeds to find out that information.

4 **Authoritative DNS Server** - The authoritative DNS server for the domain holds the information about the domain's IP address (and potentially other records like mail server settings). It answers to the resolver's query with the required IP address.

5 **Response to User** - The resolver receives the IP address from the authoritative server and caches it for future use. It then sends back the IP address to your device.

6 **Establishing Connection** - With the IP address in hand, your device can now establish a connection with the web server hosting the website you want to visit

7 **Recursive Query** - If the resolver does not have the answer, it sends a series of queries to different DNS servers. It begins by asking the root DNS servers for information about the top-level domain (TLD), then proceeds to the authoritative DNS servers responsible for the TLD. These authoritative servers direct the resolver to the DNS servers responsible for a specific domain (e.g., example.com).

Multimedia Applications

Multimedia applications are software programs or tools that integrate different forms of media, such as text, audio, video, images, and animations, to create interactive and engaging content.

These applications enable users to create, manipulate, and share multimedia content for various purposes including entertainment, education, communication, and artistic expression.

Here are some common types of multimedia applications

1 Video Editing Software

2 Audio Editing Software

3 Image Editing Software

4 Presentation Software

5 Gaming Applications

6 Web and Mobile Apps

7 Virtual Reality (VR) and Augmented Reality

8 Video Conferencing and Communication Apps

Security, Monitoring & Control

Security, monitoring, and control are critical aspects of information technology and network management. These components help organizations protect their data, assets, and systems, detect and respond to security threats, and maintain the overall health and performance of their IT infrastructure.

Here's an overview of each of these areas

Security:

- Security involves protecting systems, data, and resources from unauthorized access, attacks, and potential threats.
- It encompasses various measures and practices aimed at preventing, detecting, and responding to security breaches.

- Security measures can include encryption, access controls, firewalls, intrusion detection systems, antivirus software, and more.

Cybersecurity: Cybersecurity encompasses a wide range of practices and technologies aimed at safeguarding computer systems, networks, and data from unauthorized access, attacks, and breaches. This includes firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), antivirus software, and encryption.

Access Control: Access control mechanisms ensure that only authorized users can access specific resources or areas of a network. This involves user authentication (e.g., passwords, multi-factor authentication), authorization, and user privilege management.

Data Protection: Data security involves measures to protect sensitive data from theft, corruption, or unauthorized disclosure. Techniques include data encryption, data masking, and regular data backups.

Security Policies and Compliance: Establishing and enforcing security policies is crucial for maintaining a secure IT environment. Compliance with industry regulations (e.g., GDPR, HIPAA) and standards (e.g., ISO 27001) is also vital for organizations.

Incident Response: Organizations need plans and procedures for responding to security incidents. This includes identifying, mitigating, and recovering from security breaches or vulnerabilities.

Monitoring

- Monitoring involves observing and tracking the behaviour, performance, and activities of systems, networks, processes, or environments.
- It is essential for identifying anomalies, diagnosing issues, and ensuring that everything is functioning as expected.
- Monitoring tools and techniques can include logging, real-time alerts, dashboards, performance metrics, and more.
- In IT environments, monitoring helps maintain uptime, optimize resource utilization, and provide insights for making informed decisions.

Network Monitoring: Network monitoring tools continuously track the performance and availability of network devices, servers, and services. They provide real-time data and alerts to help IT teams detect and resolve issues promptly.

Security Monitoring: Security monitoring involves the continuous monitoring of network traffic and system logs to identify potential security threats and anomalies. Security Information and Event Management (SIEM) systems are often used for this purpose.

Application Monitoring: Monitoring the performance and availability of applications is crucial for ensuring a positive user experience. Application monitoring tools track metrics like response times, error rates, and resource utilization.

Performance Monitoring: Performance monitoring tools help IT teams optimize the performance of their infrastructure by collecting and analyzing data on system resource usage, latency, and throughput.

Compliance Monitoring: Organizations monitor their systems to ensure they comply with internal policies, industry regulations, and legal requirements.

Control

- Control refers to the ability to influence, manage, and regulate systems or processes in order to achieve desired outcomes.
- In the context of security and monitoring, control mechanisms are used to implement changes, enforce policies, and respond to incidents.
- Controls can be automated or manual and can range from simple actions like user access management to complex processes like disaster recovery planning.
- Effective controls help maintain system integrity, enforce compliance, and mitigate risk.

These three concepts are closely interconnected

- Security and Control: Security measures include controls that are designed to safeguard systems and data. Access controls, authentication mechanisms, encryption, and authorization processes are examples of security controls.
- Security and Monitoring: Monitoring is essential to detect security breaches or unusual activities. Intrusion detection systems, security information and event management (SIEM) systems, and network traffic analysis tools are used to monitor and identify potential security threats.
- Monitoring and Control: Monitoring provides real-time data and insights that are used to implement control measures. For instance, if a server's performance metrics indicate high resource utilization, a control action might involve reallocating resources to maintain optimal performance.

SNMP V2 and V3, RMON, RMON2

SNMP, which stands for Simple Network Management Protocol, was developed in 1988 by a consortium of university researchers. Its primary purpose was to offer monitoring capabilities for devices connected across TCP/IP-based networks. Just two years later, in 1990, SNMP earned recognition as an internet standard from the Internet Architecture Board (IAB).

The SNMPv2 protocol standards introduced several endeavors to tackle the security concerns inherent in SNMPv1. These efforts included the introduction of various security models like the party-based SNMPv2p, user-based SNMPv2u, and the community-based SNMPv2c.

Despite these initiatives not completely rectifying the critical security issues, SNMPv2 did bring about several enhancements over SNMPv1. Notably, it improved data retrieval capabilities through the inclusion of SNMP GETBULK operations. Moreover, SNMPv2 retained the community-based security approach established by SNMP.

SNMP V3

In the late 1990s, SNMPv3 was conceived, and by December 2002, it was ratified as a standard.

This version is delineated across RFCs 3410 to 3415. While SNMPv3 retains the fundamental SNMP management system and operations from SNMPv1 and SNMPv2, it introduces a comprehensive security architecture.

This architecture is designed in a modular fashion, allowing specific components to be enhanced without necessitating a complete overhaul.

SNMPv3's framework encompasses several models:

1 Message Processing Model (SNMPv3)

2 User-Based Security Model

3 View-Based Access Control Model

This framework is structured to support multiple models concurrently and to facilitate gradual replacements over time. For instance, although SNMPv3 introduces a new message format, it still supports messages created in SNMPv1 and SNMPv2c formats. Similarly, the user-based security model can coexist with the previously used community-based models. Additionally, SNMPv3 incorporates significant protocol updates

1 Enhanced Notification Support: SNMPv3 introduces a new notification type called INFORM. This type resembles a TRAP but requires acknowledgment. If acknowledgment is absent, the INFORM is retransmitted.

2 Trap Filtering: SNMPv3 allows filtering of TRAPs at the sender's end.

3 Dynamic Configuration: SNMP agents in SNMPv3 can be dynamically configured using MIB modules defined in RFC 3584 and RFCs 3411 through 3415

SNMP utilizes port numbers 161 and 162 for transmitting instructions and messages. Specifically, the SNMP agent employs port 161, while the SNMP manager operates through port 162.

RMON

RMON1, or Remote Network Monitoring Version 1, is an initial. version of the Remote Network Monitoring (RMON) standard. It was designed to facilitate remote monitoring and analysis of network traffic and performance on specific network segments. RMON1 focuses on providing essential statistics and information relating to network traffic and errors, primarily at the physical and data link layers of the OSI model.

Key features of RMON1 include:

1 **Packet and Byte Counts:** RMON1 allows administrators to gather information on the number of packets and bytes transmitted and received on a network segment. This data helps in understanding network utilization.

2 **Error Statistics:** RMON1 provides insights into various types of errors occurring on the network, such as CRC errors, collision counts, and other anomalies.

3 **Utilization Metrics:** Administrators can monitor the utilization of network resources, which helps in identifying congestion and potential performance issues.

4 **Promiscuous Mode:** RMON1 enables network devices to capture packets in promiscuous mode, allowing administrators to analyses all traffic passing through a specific segment.

5 **Historical Data:** RMON1 supports historical data collection, allowing administrators to track network trends over time.

6 **Alarms and Events:** RMON1 can generate alarms or events based on specified thresholds, notifying administrators when specific conditions are met (e.g., excessive errors).

7 **Protocol Distribution:** This feature provides statistics about the distribution of different network protocols, helping administrators understand the composition of network traffic.

The wireless channel - Link level design - Channel Access Network design - Standards

It seems like you're looking for information on wireless communication, link-level design, channel access methods, and standards. I'll provide a brief overview of each topic:

1 Wireless Channel: The wireless channel refers to the medium through which wireless signals propagate between devices. It's influenced by factors such as distance, obstacles, interference, and environmental conditions. To design an effective wireless communication system, understanding the characteristics of the wireless channel is crucial. Different wireless technologies (e.g., Wi-Fi, cellular, Bluetooth) use various frequency bands and modulation schemes to mitigate channel effects and improve signal reliability.

2 Link-Level Design: Link-level design focuses on optimizing the communication link between a transmitter and a receiver. This involves choosing modulation schemes, coding techniques, and error correction mechanisms to maximize data throughput while maintaining a reliable connection. The design also considers signal-to-noise ratio (SNR), bit error rate (BER), and other performance metrics to ensure efficient data transmission.

3 Channel Access Methods: In wireless networks, multiple devices share the same channel, which can lead to collisions and reduced efficiency. Channel access methods determine how devices access and use the shared channel. Common methods include:

- Frequency Division Multiple Access (FDMA): Divides the channel into frequency bands, with each device allocated a specific band.
- Time Division Multiple Access (TDMA): Divides the channel into time slots, allowing different devices to transmit at different times.
- Code Division Multiple Access (CDMA): Uses unique codes to differentiate between devices, allowing multiple devices to transmit simultaneously.
- Carrier Sense Multiple Access (CSMA): Devices listen for a clear channel before transmitting to avoid collisions. Variants include CSMA/CA (Collision Avoidance) used in Wi-Fi and CSMA/CD (Collision Detection) used in Ethernet.

4. Standards: Various organizations develop and maintain standards for wireless communication to ensure interoperability and widespread adoption. Some notable wireless standards include:
   - Wi-Fi (IEEE 802.11): Standard for wireless local area networks (WLANs). It defines different generations (802.11a/b/g/n/ac/ax) with varying data rates and features.
   - Cellular Networks (e.g., 4G LTE, 5G): Standards developed by organizations like 3GPP for mobile communication, offering high-speed data, low latency, and seamless mobility.
   - Bluetooth (IEEE 802.15.1): Standard for short-range wireless communication between devices, commonly used for connecting peripherals.
   - Zigbee (IEEE 802.15.4): Standard for low-power, short-range communication often used in applications like home automation and sensor networks.
   - NFC (Near Field Communication): Standard for short-range communication used for contactless payments and data exchange.

These standards help ensure that devices from different manufacturers can communicate effectively and that users can seamlessly switch between different networks or technologies.

Optical Networks - Cross connects - LANS

Optical Networks

- Optical networks are telecommunication networks that use optical fibers to transmit information in the form of light signals.
- These networks leverage the properties of light to transmit data over long distances with high speed and minimal signal loss.
- Optical networks are widely used for various communication applications due to their numerous advantages over traditional copper-based networks.

In an optical network, information is carried by modulating light signals with data. The light signals travel through optical fibers, which are thin strands made of glass or plastic designed to guide the light along their length through multiple internal reflections. These fibers have a core, where the light travels, surrounded by a cladding that reflects the light back into the core to prevent signal loss.

Key Characteristics and Components of Optical Networks:

1 **High Data Rates:** Optical networks can achieve extremely high data rates, ranging from gigabits per second (Gbps) to terabits per second (Tbps). This makes them well-suited for transmitting large volumes of data quickly.

2 **Large Bandwidth:** Optical fibers have a broad bandwidth capacity, allowing multiple signals to be transmitted simultaneously on different wavelengths using techniques like Wavelength Division Multiplexing (WDM).

3 **Low Signal Loss:** Optical signals can travel over long distances without significant signal degradation or loss of quality, making optical networks suitable for long-haul communication.

4 **Low Interference:** Optical signals are less susceptible to electromagnetic interference compared to electrical signals on copper cables.

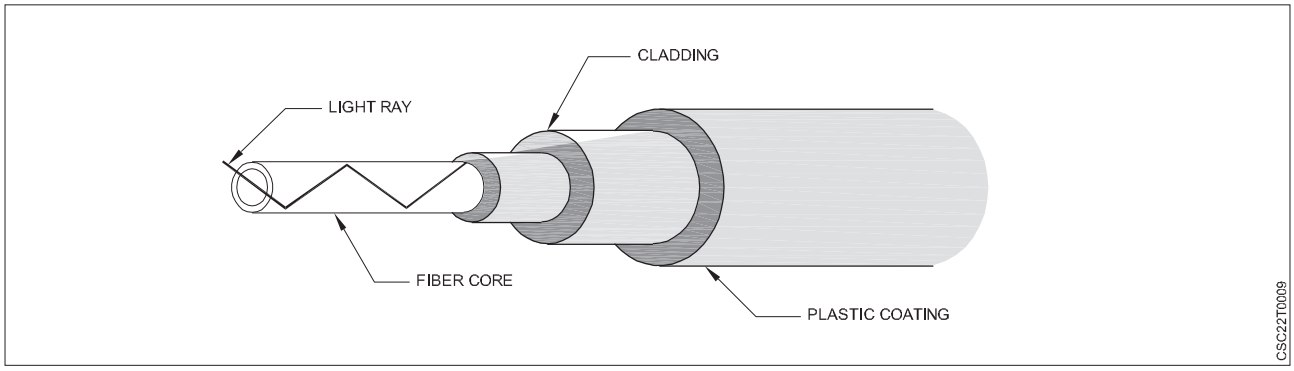
Applications of Optical Networks:

1 **Telecommunications:** Optical networks are used in telecommunication systems to transmit voice, data, and video signals over long distances. They form the backbone of modern telecommunications infrastructure.

2 **Data Centers:** Within data centers, optical networks provide high-speed connections between servers, storage systems, and networking equipment. These connections enable rapid data exchange and efficient data center operation.

3 **Internet Backbone:** Optical networks form the core of the global internet, interconnecting major data centers and network nodes worldwide to facilitate data transmission across regions.

4 **Cable Television (CATV):** Optical networks are used for distributing cable television signals to homes, allowing for high-quality video and audio delivery.



Cross Connects: A cross connect is a physical or logical connection between two or more communication channels within a network.

The purpose of a cross connect is to facilitate the routing of data between these channels without the need to go through intermediate points.

Cross connects are commonly used in data centers and telecommunications facilities to create efficient and flexible network architectures.

There are two main types of cross connects:

- Physical Cross Connect: In a physical cross connect, cables are physically connected to routing or switching equipment to establish a direct link between two network interfaces. This is often done using patch panels, where different cables can be connected or disconnected manually.
- Logical Cross Connect: A logical cross connect is a virtual or software-based connection that is established within a network device, such as a router or switch. It involves configuring routing tables or software-defined networking (SDN) controllers to direct traffic between specified endpoints.

LANs (Local Area Networks)

LANs (Local Area Networks): LANs are networks that cover a limited geographic area, such as a single building, campus, or small group of buildings. LANs are used to connect devices like computers, printers, and servers within a relatively close proximity. Ethernet is the most common technology used for LANs, and it provides high-speed data transmission over twisted-pair copper cables or optical fibers.

In the context of optical networks, cross connects can be used to link different LAN segments within a larger network infrastructure. This helps manage traffic efficiently and allows for scalability as the network grows.

Voice Over IP - Multimedia Networks

Voice Over IP (VoIP):

Voice over Internet Protocol (VoIP) is a technology that allows voice communication and multimedia sessions to be transmitted over the Internet or other IP-based networks.

Instead of using traditional circuit-switched networks, VoIP converts voice signals into digital data packets and transmits them over data networks.

This technology has revolutionized communication by enabling cost-effective and feature-rich voice communication, as well as integration with other forms of multimedia content.

Key Features of VoIP:

1 **Cost Savings:** VoIP generally offers lower costs for long-distance and international calls compared to traditional telephone services.

2 **Rich Features:** VoIP systems often include features such as call forwarding, voicemail, caller ID, conference calling, and more.

3 **Integration:** VoIP can easily integrate with other applications and services, such as video conferencing, instant messaging, and email.

4 **Scalability:** VoIP systems can be easily scaled to accommodate a growing number of users or devices.

5 **Flexibility:** Users can make calls from computers, VoIP phones, mobile devices, and other compatible devices.

6 **Unified Communications:** VoIP enables the integration of voice, video, and data communication, promoting unified communication experiences.

7 **Advanced Services:** VoIP supports advanced services like virtual phone numbers, call routing, and interactive voice response (IVR) systems.

Multimedia Networks:

- Multimedia networks are communication networks designed to handle various types of multimedia data, including voice, video, text, and images.
- These networks provide the infrastructure necessary to transmit, receive, and manage different types of media content. Multimedia networks are crucial for applications such as video conferencing, streaming services, online gaming, and more.

Key Aspects of Multimedia Networks:

1 **QoS (Quality of Service):** Multimedia networks prioritize certain types of traffic, like voice and video, to ensure consistent quality and low latency.

2 **Bandwidth Management:** Managing bandwidth effectively is essential for delivering high-quality multimedia content without bottlenecks.

3 **Real-Time Communication:** Multimedia networks are optimized for real-time communication, ensuring minimal delays in transmitting data.

4 **Media Compression:** To optimize data transmission, multimedia content is often compressed using codecs.

5 **Security:** Secure transmission of multimedia content is critical to protect sensitive information and maintain privacy.

6 **Content Delivery:** Multimedia networks often involve content delivery networks (CDNs) to efficiently distribute large files or streaming content.

7 **Protocols:** Various protocols, such as Real-Time Transport Protocol (RTP) for real-time media, are used in multimedia networks.

Integration of VoIP and Multimedia Networks:

VoIP is a crucial component of multimedia networks as it provides the means for transmitting real-time voice communication. In a multimedia network, VoIP technology can work alongside video streaming, text chat, and other multimedia services to offer comprehensive communication experiences. Integration of VoIP with multimedia networks allows users to engage in voice calls, video conferencing, instant messaging, and other multimedia interactions seamlessly over a single network infrastructure.

A Virtual Private Network (VPN) is a technology that creates a secure and encrypted connection between your device and a remote server.

Introduction to VPN and DHCP

This connection allows you to access the internet or other network resources while maintaining privacy, security, and anonymity.

VPNs are used to enhance online security, protect sensitive data, and enable remote access to private networks.

Key features and uses of VPNs include:

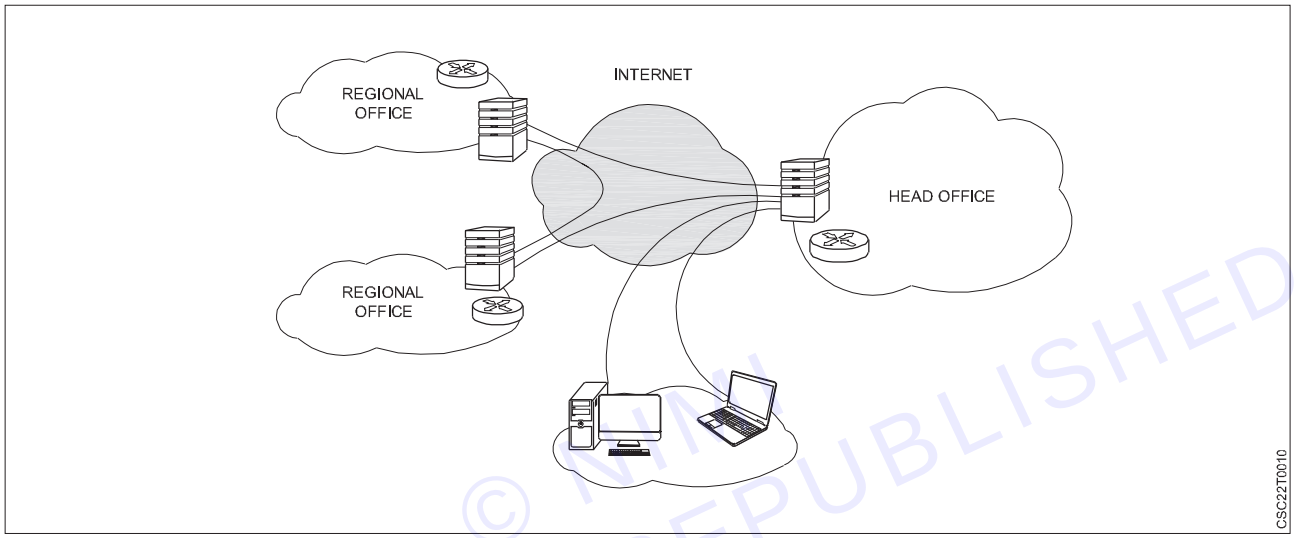
1 **Privacy and Anonymity:** VPNs mask your IP address and encrypt your internet traffic, making it difficult for third parties, such as websites and hackers, to track your online activities. This helps maintain your anonymity and privacy.

2 **Security:** The encryption used in VPNs ensures that your data remains confidential and secure while being transmitted over potentially insecure networks, like public Wi-Fi hotspots.

3 **Bypass Geographic Restrictions:** VPNs enable you to access content that might be restricted or blocked in your region. By connecting to a server in a different location, you can appear as if you're browsing from that region and access region-specific content.

4 **Remote Access:** Businesses use VPNs to provide secure remote access for employees who need to connect to the company's internal network from outside locations. This is particularly useful for remote work or traveling employees.

5 **Types of VPNs:** There are various types of VPNs, including remote-access VPNs and site-to-site VPNs. Remote-access VPNs are used by individuals to connect to a private network over the internet. Site-to-site VPNs are used to connect multiple networks in different locations.



Introduction to DHCP

Dynamic Host Configuration Protocol (DHCP) is a network protocol used to automatically assign IP addresses and other network configuration settings to devices on a network. It simplifies the process of connecting devices to a network by eliminating the need for manual IP configuration.

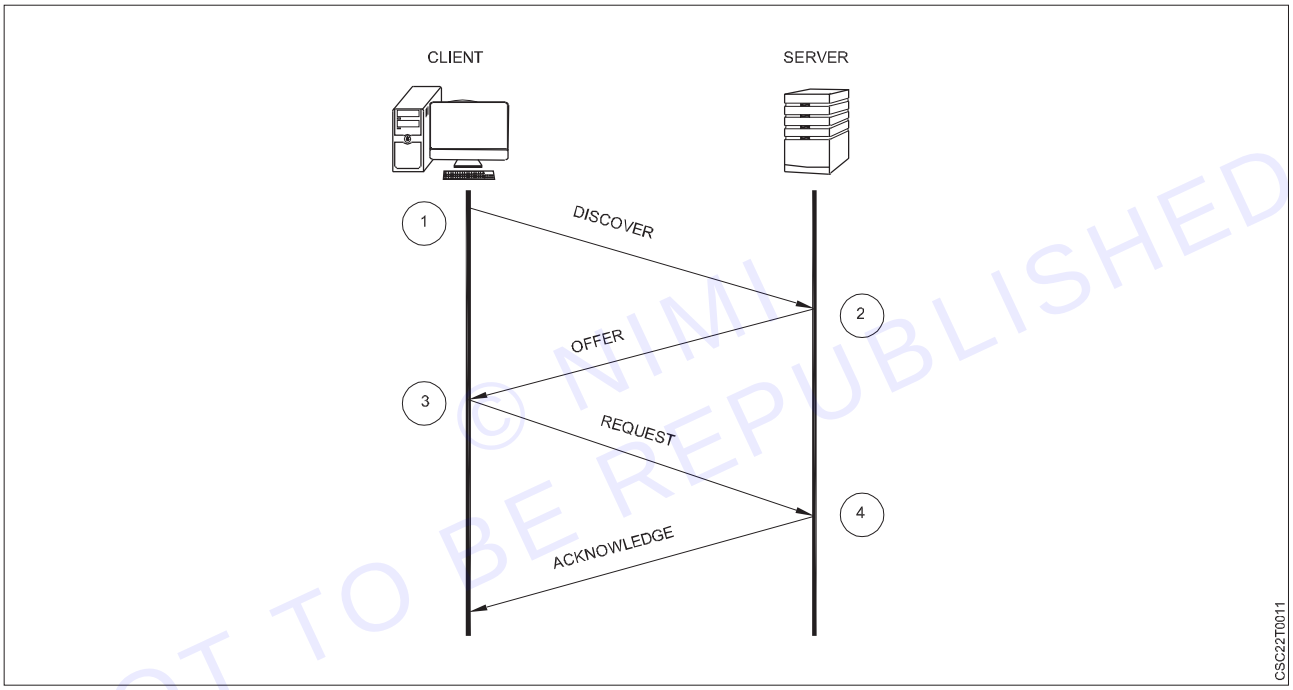
Key features and uses of DHCP include:

- **Automatic IP Address Assignment:** DHCP servers allocate IP addresses dynamically to devices as they connect to the network. This automation reduces the risk of IP address conflicts and simplifies network management.
- **Efficient Resource Allocation:** DHCP optimizes the allocation of IP addresses by releasing them when they are no longer in use. This prevents address wastage and ensures efficient use of available addresses.
- **Centralized Configuration:** DHCP enables centralized management of network configuration settings. It can provide additional information to devices, such as DNS server addresses, default gateways, and subnet masks.
- **Scalability:** DHCP is highly scalable and suitable for both small and large networks. It streamlines the process of adding new devices to the network without manual configuration.
- **Lease Management:** DHCP leases are time-limited assignments of IP addresses. This allows network administrators to control how long devices can retain an IP address, ensuring that addresses are periodically released for other devices to use.

DORA Process

The "DORA" process is an acronym that stands for "Discover, Offer, Request, Acknowledge." It refers to the sequence of steps in the Dynamic Host Configuration Protocol (DHCP) through which a client device obtains an IP address and other network configuration information from a DHCP server. Here's a breakdown of each step in the DORA process:

- **Discover:** In this initial step, the client device (often a computer, smartphone, or any device seeking network connectivity) sends out a DHCP "Discover" message as a broadcast signal on the local network. This message indicates that the client is in need of an IP address and other configuration parameters.
- **Offer:** When a DHCP server receives the "Discover" message, it responds with a DHCP "Offer" message. This message is a broadcast sent by the DHCP server to the client, containing a proposed IP address, subnet mask, lease duration, and other network configuration information. The server temporarily reserves the offered IP address for the client.
- **Request:** Upon receiving one or more "Offer" messages, the client evaluates the offers and selects one of the proposed IP addresses. The client then sends a DHCP "Request" message to the chosen server, requesting the use of the offered IP address and confirming its acceptance of the configuration parameters.
- **Acknowledge:** Once the DHCP server receives the "Request" message from the client, it sends a DHCP "Acknowledge" (or "ACK") message. This message confirms that the client has been assigned the requested IP address and provides the client with the approved network configuration details. The client device then configures its network settings based on the provided information.



Attacks, Services and Mechanisms, Security Attacks, Security Services, Integrity check, Digital Signatures, Authentication

Network security is a critical aspect of modern computing and technology that involves the protection of a computer network infrastructure from various threats and unauthorized access. It encompasses a range of practices, technologies, and policies designed to ensure the confidentiality, integrity, and availability of network resources and data.

Numerous individuals depend on the Internet for a wide array of personal, social, and professional tasks. However, there exists a faction that seeks to harm our internet-linked computers, infringe upon our privacy, and disrupt internet services, rendering them useless.

Network attack

Network attacks are malicious activities or actions that target vulnerabilities in computer networks with the intent to compromise their confidentiality, integrity, or availability. These attacks can vary in sophistication and impact, ranging from simple exploits to complex, coordinated efforts.

There are two main types of network attacks

1 Active Attacks

2 Passive Attacks

Active Attacks

An active attack is a type of malicious activity in which an unauthorized party takes deliberate action to breach the security of a computer system, network, or device. Unlike passive attacks, which involve eavesdropping or monitoring without altering data, active attacks involve direct interference with the target to gain unauthorized access, disrupt services, or manipulate data.

Here are some common types of active attacks

1 **Spoofing:** Attackers manipulate network protocols, IP addresses, or other identification information to impersonate a trusted entity, gain unauthorized access, or deceive users

2 **Denial of Service (DoS) Attack:** As previously mentioned, this attack floods a network, server, or service with excessive traffic to make it unavailable to legitimate users.

- **DoS:** Overwhelming a single system with a flood of traffic to make it unavailable.
- **DDoS:** Coordinating multiple systems to flood a target with traffic, amplifying the impact.

3 **Brute Force Attack:** Attackers attempt to guess passwords or encryption keys by systematically trying all possible combinations until they find the correct one.

4 **Password Attacks:** This includes various methods like dictionary attacks, where attackers try common passwords, or credential stuffing, where stolen usernames and passwords from one site are used on other sites.

5 **SQL Injection:** Attackers manipulate input fields on a website to inject malicious SQL code into a database, potentially allowing unauthorized access or data retrieval.

6 **Malware Attacks:** These involve deploying malicious software onto a system to compromise its security, steal data, or perform other malicious actions.

- **Viruses:** Malicious programs that attach themselves to legitimate files and replicate when the infected file is executed.
- **Worms:** Self-replicating programs that spread across networks and systems without human intervention.
- **Trojans:** Malware disguised as legitimate software, often used to gain unauthorized access to systems.

7 Spoofing A specific type of malware that encrypts a user's files and demands a ransom for decryption.

8 **Phishing:** While primarily a form of social engineering, phishing emails may also lead to active attacks, such as directing users to malicious websites that download malware onto their systems.Phishing: Deceptive emails or messages aimed at tricking recipients into revealing sensitive information, such as passwords or credit card details.

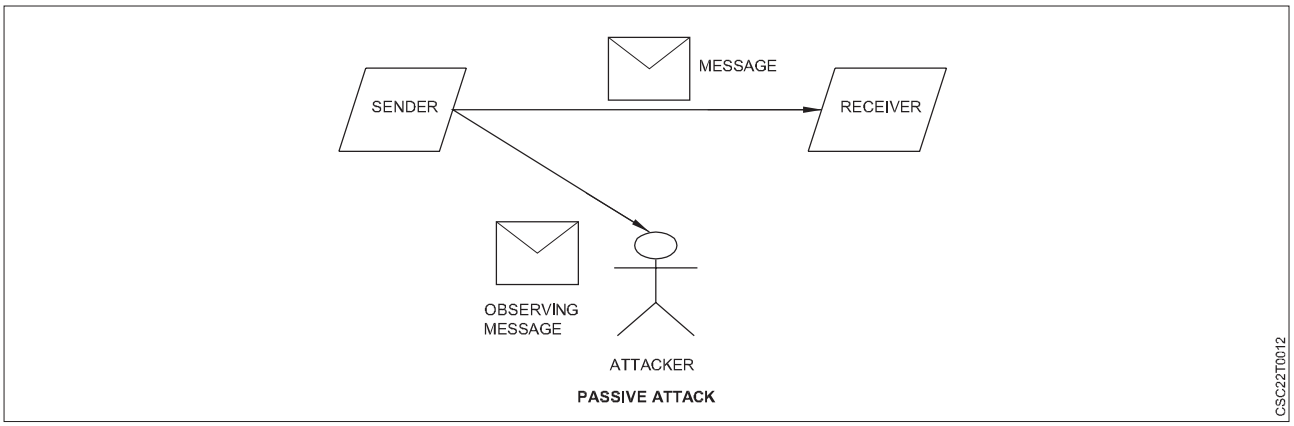
- **Spear Phishing:** Targeted phishing attacks aimed at specific individuals or organizations.
- **Whaling:** Similar to spear phishing, but targeting high-profile individuals, executives, or celebrities.

Passive Attacks

Passive attacks are a type of cybersecurity attack that focuses on intercepting and gathering information from a targeted system or network without altering the data or causing any noticeable disruption. Unlike active attacks that involve modifying or damaging data, passive attacks are primarily concerned with unauthorized access to sensitive information, such as confidential data, credentials, or communication content. These attacks are often difficult to detect because they don't involve direct manipulation of data, making them a significant concern for maintaining data privacy and security.

There are two main categories of passive attacks:

1 **Eavesdropping:** Eavesdropping attacks involve an unauthorized individual or entity intercepting and monitoring data transmissions between legitimate users. This can happen on both wired and wireless networks.



Attackers might use techniques like packet sniffing to capture data packets as they travel across the network. The intercepted data might contain sensitive information, such as passwords, financial details, or confidential messages.

2 **Traffic Analysis:** Traffic analysis attacks focus on observing patterns in communication, even without directly accessing the content of the messages. Attackers analyze factors like message frequency, size, timing, and the parties involved to deduce information about the communication. For example, an attacker might infer the relationship between two individuals by analyzing the frequency and timing of their communication.

Difference Between Active & Passive Attack

Active Attack	Passive Attack
In an active attack, Modification in information takes place.	While in a passive attack, Modification in the information does not take place.
Active attack is a danger to integrity as well as availability.	Passive attack is a danger to Confidentiality.
In an active attack, attention is on prevention.	While in passive attack attention is on detection.
Due to active attacks, the execution system is always damaged.	While due to passive attack, there is no harm to the system.

Services- In the context of computer networks, services refer to functions or capabilities provided by networked systems to users or other systems. These services facilitate communication, resource sharing, and other network activities. Examples of network services include:

- **File Sharing:** Allowing users to access and share files on a network.
- **Email:** Sending and receiving electronic messages.
- **Web Hosting:** Hosting websites accessible over the internet.
- **Domain Name System (DNS):** Resolving domain names to IP addresses.
- **Remote Access:** Accessing a computer or network from a remote location.
- **Directory Services:** Managing and organizing information about resources in a network.
- **Authentication and Authorization:** Verifying user identities and controlling access to resources.

Mechanisms: Mechanisms in cybersecurity refer to the tools, technologies, and practices used to protect systems and networks from attacks and maintain their security. Some common security mechanisms include:

- **Firewalls:** Hardware or software devices that monitor and control incoming and outgoing network traffic based on predetermined security rules.
- **Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS):** Monitoring and responding to suspicious network activities.
- **Encryption:** Transforming data into a secure format to prevent unauthorized access during transmission or storage.

- **Access Control:** Regulating who can access what resources based on user identities and permissions.
- **Multi-factor Authentication (MFA):** Requiring multiple forms of verification for user authentication.
- **Vulnerability Assessment:** Identifying and assessing vulnerabilities in systems and networks.
- **Penetration Testing:** Simulating attacks to identify vulnerabilities and weaknesses in security defenses.
- **Security Information and Event Management (SIEM):** Collecting and analyzing security data to detect and respond to threats.

These components - attacks, services, and mechanisms—are integral to the field of cybersecurity, helping organizations protect their systems, data, and networks from a wide range of threats.

Security Services: Security services refer to various measures and mechanisms put in place to ensure the protection of information and resources in a computer system or network.

These services are designed to maintain the confidentiality, integrity, availability, and authenticity of data. Some common security services include access control, encryption, authentication, and auditing.

Integrity Check: Integrity refers to the accuracy and reliability of data. An integrity check is a process or mechanism used to verify that data has not been tampered with or altered in an unauthorized manner.

This can involve various techniques such as checksums, hash functions, and digital signatures to detect any unauthorized modifications to data.

Digital Signatures

A digital signature is a cryptographic technique that provides authentication, data integrity, and non-repudiation for digital documents or messages. It's a way to ensure that the sender of a message is verified, that the message hasn't been altered in transit, and that the sender cannot later deny having sent the message.

Here's how a digital signature works:

1 Message Digest Generation:

The sender creates a unique hash value (also known as a message digest) of the content they want to sign. This is typically done using a hash function like SHA-256. The hash value is a fixed-size string of characters that is unique to the content of the message.

2 Signing:

The sender uses their private key to encrypt the hash value of the message. This encrypted hash value is the digital signature. The private key is a secret and should only be known to the sender.

3 Sending:

The original message, along with the digital signature, is sent to the recipient.

4 Verification:

The recipient uses the sender's public key (which is available to everyone) to decrypt the digital signature. This produces the original hash value.

5 Message Digest Calculation:

The recipient independently calculates the hash value of the received message using the same hash function.

6 Comparison:

The recipient compares the calculated hash value to the decrypted hash value (original hash value from the sender). If they match, it means the message hasn't been altered in transit and that the signature is valid.

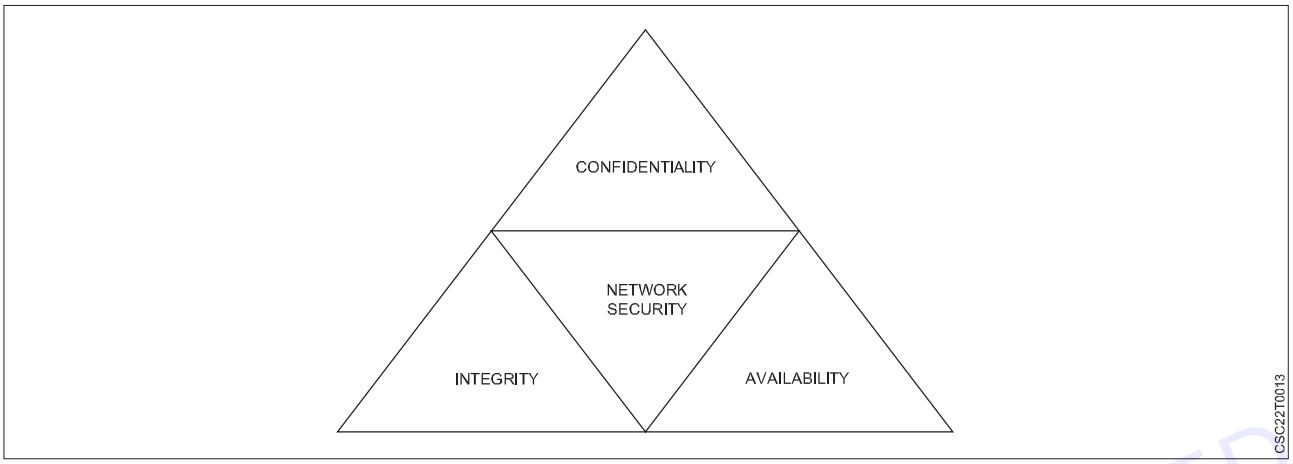
The digital signature ensures the following:

- **Authentication:** The recipient can verify the identity of the sender because only the sender's private key could have produced the correct digital signature.
- **Data Integrity:** Any modification of the original message, even a minor one, will result in a completely different hash value. This means that the recipient can detect if the message has been tampered with.
- **Non-Repudiation:** Since the digital signature is tied to the sender's private key, the sender cannot deny sending the message later on.

Digital signatures are widely used for various purposes, such as signing contracts electronically, securing email communications, validating software updates, and more. They play a crucial role in ensuring the authenticity and integrity of digital transactions and communications

CIA Tried

The CIA triad is a widely recognized model for information security. It stands for Confidentiality, Integrity, and Availability, which are three essential concepts that help to ensure the security of sensitive information.



Confidentiality

This refers to the protection of information from unauthorized access or disclosure. Confidentiality ensures that sensitive information is only accessible to authorized individuals or systems. This can be achieved through methods such as encryption, access controls, and secure communications.

Tools for Confidentiality

- **Encryption**

Encryption involves converting information into an unintelligible form to prevent unauthorized individuals from comprehending it. This is achieved through the utilization of algorithms, with the transformation of data being facilitated by a confidential encryption key. Consequently, only those in possession of the corresponding decryption key can revert the transformed data back into a readable format. By employing encryption, confidential data such as credit card details can be safeguarded as it is converted into an indecipherable ciphertext. The sole method to access this encrypted data is by employing decryption. The two main categories of encryption are asymmetric-key and symmetric-key encryption.

- **Access control**

Access control establishes regulations and guidelines for restricting entry to a system, as well as to tangible or digital assets. It constitutes a procedure through which users receive permission to access systems, assets, or information along with specific entitlements. Access control mechanisms necessitate users to furnish authentication details prior to obtaining entry, which can encompass individual names or device identifiers. In instances of tangible setups, these validation elements can assume diverse formats, although non-transferable credentials offer the highest degree of security.

- **Authentication**

Authentication is a procedure that verifies and affirms an individual's identity or authorized role. It encompasses various methods, often relying on a combination of the following factors:

- Something the individual possesses (such as a smart card or a radio key containing confidential keys).
- Something the individual knows (like a password).
- Something intrinsic to the individual (such as a fingerprint).

Authentication is indispensable for organizations as it empowers them to ensure the security of their networks by granting access solely to authenticated users for their safeguarded assets. These assets might span computer systems, networks, databases, websites, as well as other web-based applications or services.

- **Authorization**

Authorization serves as a security protocol that confers the right to perform certain actions or possess specific privileges. Its purpose lies in establishing whether an individual or system possesses the entitlement to access resources, following an access control framework. These resources encompass an array of elements such as computer software, files, services, data, and attributes of applications. Normally, authorization follows the preliminary step of authentication, which validates the identity of the user. System administrators often hold designated permission levels that encompass both system-wide and user-specific resources. In the process of authorization, a system validates the access regulations of an authenticated user, subsequently permitting or denying access to the designated resources

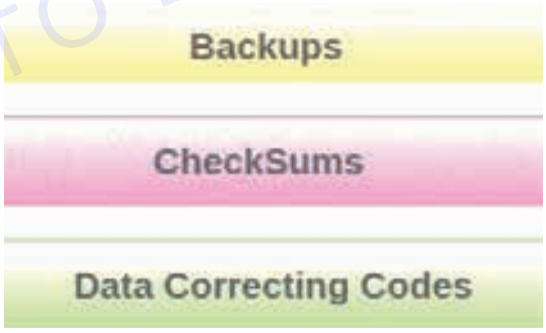
- **Physical security**

Physical security encompasses strategies implemented to prevent unauthorized entry to IT assets, such as facilities, equipment, personnel, resources, and other valuable properties, with the aim of averting damage. Its primary role is safeguarding these assets against tangible hazards, which encompass risks like theft, vandalism, fires, and natural catastrophes.

Integrity

This refers to the protection of information from unauthorized modification, deletion, or corruption. Integrity ensures that information is accurate and trustworthy. Methods to ensure integrity include data validation checks, digital signatures, and access controls.

Tools for Integrity



Backups

Backup involves creating regular copies of data or files. This is done to have duplicates available in case the original data is lost or damaged. Additionally, backups can serve historical purposes like long-term studies, statistics, or meeting data retention policies. In various systems, including Windows, applications often generate backup files with the ".BAK" extension.

Checksum

A checksum is a numeric value utilized to validate the accuracy of a file or data transfer. It's essentially a calculation that transforms the contents of a file into a numerical value. Its main purpose is to compare two sets of data and confirm their equivalence. The calculation of a checksum takes into account the complete content of a file. The design of a checksum function ensures that even a minor alteration in the input file, like a single bit being flipped, is highly likely to produce a distinct output value.

Data Correcting Codes

It's a technique for encoding data in a manner that enables effortless detection and automatic correction of minor alterations.

Availability

This refers to the ability of authorized individuals or systems to access information when needed. Availability ensures that information is accessible and usable. Methods to ensure availability include redundancy, backup and recovery, and disaster recovery planning.

Tools for Availability

- Physical safeguards and computational redundancies.

Physical safeguards

Physical security involves maintaining the accessibility of information despite physical obstacles. This entails securing sensitive data and essential information technology within protected environments.

Computational redundancies

It's employed to enhance resilience against unintended errors. This safeguards computers and storage units that act as backups in the event of malfunctions.

Concept of Cryptography

Cryptography is the art of securing information and communication by employing codes, ensuring that only intended recipients can comprehend and handle the data.

This safeguards against unauthorized access.

The term "crypt" signifies "hidden," and "graphy" denotes "writing." In cryptography, methods derived from mathematical principles and algorithms—sets of rule-based calculations—are used to transform messages in manners that hinder easy decoding.

These algorithms are applied in various tasks like generating cryptographic keys, digital signatures, and verification.

They serve to uphold data privacy, enable secure internet browsing, and safeguard sensitive transactions like credit and debit card dealings.

Contemporary cryptography revolves around four primary goals:

1 **Confidentiality:** Ensuring that information remains incomprehensible to anyone other than its intended recipients.

2 **Integrity:** Guaranteeing that information cannot be modified during storage or transmission without being noticed by the intended receiver.

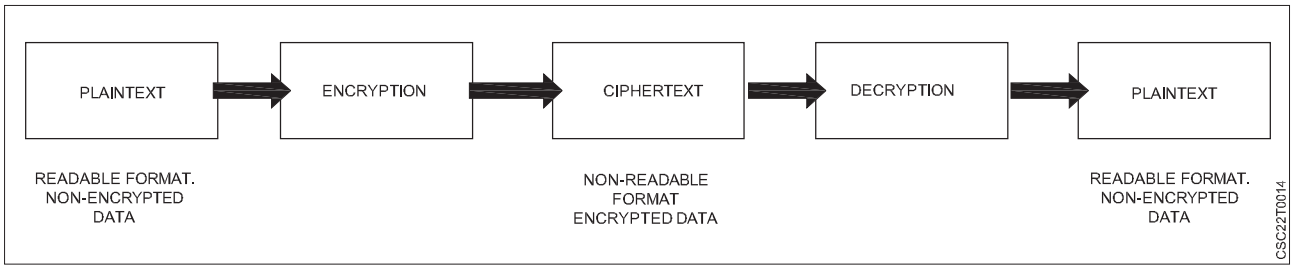
3 **Non-repudiation:** Preventing the originator/sender of information from disowning their involvement in creating or sending the information at a later point.

4 **Authentication:** Enabling both the sender and receiver to verify each other's identities and the source/ destination of the information.

At its core, cryptography comprises two essential phases:

Encryption and Decryption.

During Encryption, a cipher is applied to the plaintext, converting it into ciphertext. Decryption, on the other hand, involves using the same cipher to reverse the process, converting the ciphertext back into plaintext.



he primary application of cryptography in electronic data transmission is the encryption and decryption of emails and other plaintext messages. The most straightforward technique is the “secret key” or symmetric approach.

In this method, a secret key is employed to encrypt the data, and upon decryption, the secret key and the encoded message are shared with the recipient. However, a significant problem arises from this process. If intercepted, a third party could use the shared key to decipher and analyze the message.

To address this issue, cryptographers developed the asymmetric or “public key” approach. In this scheme, each user possesses two keys: a private key and a public key. Before sending a message, the sender obtains the recipient’s public key and uses it to encrypt the message. Since only the recipient has access to their corresponding private key, they can decrypt the message.

This asymmetric approach resolves the security vulnerability of the secret key approach, ensuring that even if the communication is intercepted, only the intended recipient possessing the private key can decipher the message.

Plain Text: Plain text is the original, unencrypted message or data that you want to protect or transmit securely. It’s the human-readable form of the information that you can easily understand. For example, if you have a message like “Hello, this is a secret message,” that would be the plain text.

Ciphertext: Ciphertext refers to the transformed and encrypted form of data that has undergone encryption using cryptographic techniques. It is the result of applying an encryption algorithm to plaintext (original, readable data) in order to secure it during transmission or storage. Ciphertext appears as a seemingly random and unreadable sequence of characters, making it unintelligible without the appropriate decryption key or algorithm. The primary purpose of ciphertext is to protect sensitive information from unauthorized access, ensuring its confidentiality and integrity.

For instance, let’s employ the Caesar Cipher to encrypt a sentence. With a key of 7, the letter ‘a’ shifts to ‘h’.

Original Sentence: This is a plaintext.

Encrypted Sentence(Ciphertext): Aopz pz h wshpualea.

Purpose of cryptography

- The purpose of cryptography is to secure communication and data by converting information into a format that is unintelligible to unauthorized individuals.
- This safeguards sensitive information during transmission and storage, preventing unauthorized access, eavesdropping, and tampering.
- Cryptography also enables the verification of data authenticity and the authentication of users or entities in digital transactions.
- Overall, cryptography plays a vital role in ensuring privacy, data integrity, and secure interactions in various digital environments.

Cryptographic algorithms

Cryptographic algorithms, also referred to as ciphers, are essential components of cryptosystems that ensure secure communication between computer systems, devices, and applications.

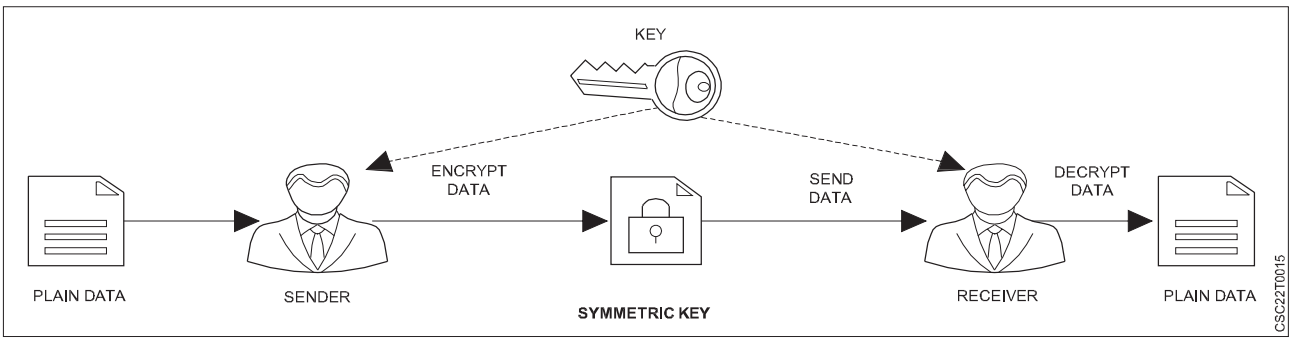
A cipher suite encompasses various algorithms: one for encryption, another for message authentication, and yet another for key exchange. These processes are integrated into protocols and implemented through software operating on operating systems and interconnected computer networks. This involves:

- Generating public and private keys for encrypting and decrypting data.
- Performing digital signatures and verification for authenticating messages.
- Executing key exchange mechanisms to establish secure communication channels.

Types of Cryptography

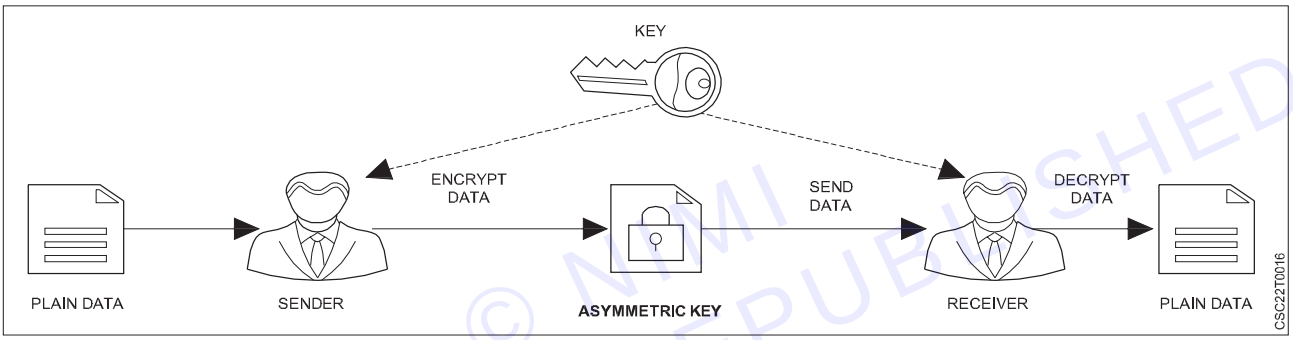
Symmetric key

Symmetric key cryptography involves a method where both the sender and recipient utilize a common shared key for both encrypting and decrypting messages.



Asymmetric key

Asymmetric key cryptography, also known as public-key cryptography, employs a pair of keys: a public key and a private key. The sender uses the recipient's public key to encrypt the message, and the recipient utilizes their private key to decrypt it. Conversely, the sender can sign a message with their private key, and the recipient can verify the signature using the sender's public key. This approach eliminates the need for a shared secret key and simplifies the key exchange process. Asymmetric key cryptography provides enhanced security but is generally slower than symmetric key cryptography.

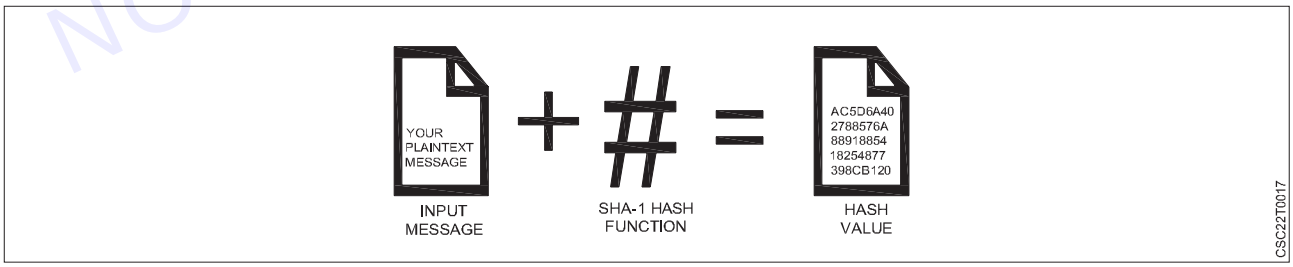


Hash Function

A hash function is a mathematical algorithm that takes an input (or "message") and produces a fixed-size string of characters, which is usually a hexadecimal number.

The output, often referred to as the "hash value" or "hash code," is unique to the specific input data.

Hash functions are designed to be fast to compute and irreversible, meaning it's practically impossible to go from the hash value back to the original input data



Some of the most famous hashing algorithms are-

Several well-known hashing algorithms are widely used for various cryptographic and data integrity purposes. Some of the most famous ones include:

- **MD5 (Message Digest Algorithm 5):** MD5 produces a 128-bit hash value. However, it is considered weak and insecure due to vulnerabilities that allow collision attacks.
- **SHA-1 (Secure Hash Algorithm 1):** SHA-1 produces a 160-bit hash value. Like MD5, it's considered weak due to vulnerabilities. It's no longer recommended for security-sensitive applications.
- **SHA-256 (Secure Hash Algorithm 256):** A member of the SHA-2 family, SHA-256 produces a 256-bit hash value. It's widely used for digital signatures and certificates and is considered secure.

- **SHA-3 (Secure Hash Algorithm 3):** The latest member of the SHA family, SHA-3 was designed to provide a new level of security and resistance to various attacks.
- **Blake2:** A high-speed cryptographic hash function that's an improvement over SHA-3 in terms of speed and security.
- **Whirlpool:** A cryptographic hash function that produces a 512-bit hash value. It's used in various security applications and is known for its strong security properties.
- **RIPEMD (RACE Integrity Primitives Evaluation Message Digest):** RIPEMD comes in several versions, including RIPEMD-160. It was designed as an alternative to MD5 and SHA-1.
- **HMAC (Hash-based Message Authentication Code):** While not a hash function itself, HMAC uses a cryptographic hash function (like SHA-256) along with a secret key to create a message authentication code. It's used to verify the integrity and authenticity of messages.

SSL Protocol

SSL (Secure Sockets Layer) protocol

The SSL (Secure Sockets Layer) protocol is a cryptographic protocol designed to provide secure communication over a computer network, typically the internet. It ensures that the data transmitted between a client (such as a web browser) and a server is encrypted and protected from eavesdropping, tampering, and forgery.

SSL was developed by Netscape Communications in the 1990s, and its successor is TLS (Transport Layer Security). TLS continues to be used widely today for securing online transactions, sensitive data transmission, and various forms of communication.

The SSL/TLS protocol operates by establishing a secure communication channel between the client and server using a combination of encryption, authentication, and data integrity mechanisms.

Secure Socket Layer Protocols:

- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol

SSL record protocol

The SSL Record Protocol is responsible for dividing the application data into manageable chunks, adding encryption and integrity protection, and then transmitting these chunks as records over the network. It ensures confidentiality, integrity, and authenticity of the data being exchanged between the client and server. The protocol uses cryptographic algorithms to achieve these goals.

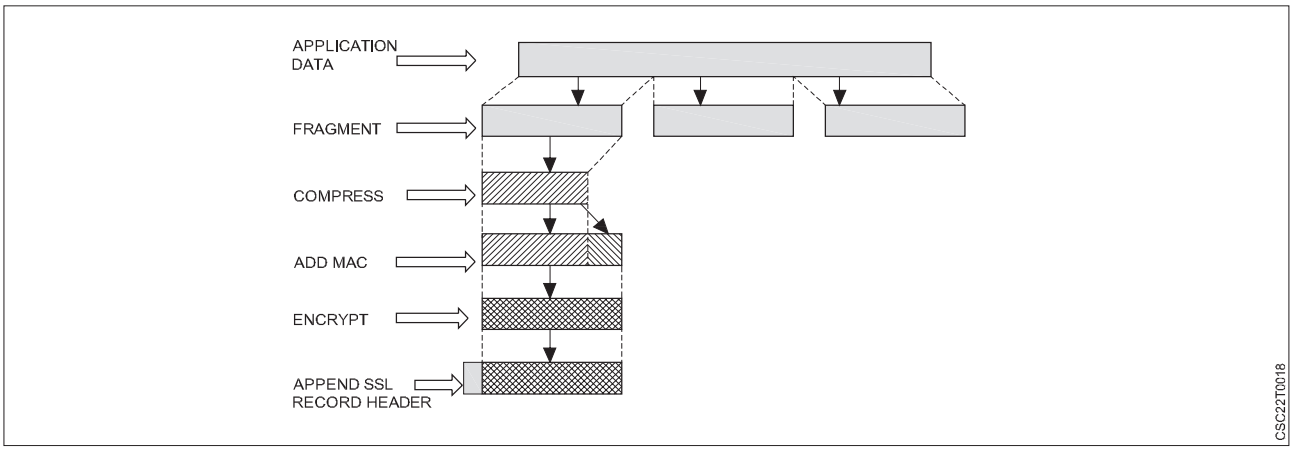
Certainly, the SSL Record Protocol provides two fundamental services to the SSL/TLS connection:

1 **Confidentiality:** The SSL Record Protocol ensures the confidentiality of data by encrypting the application data before transmission. This means that any data sent between the client and server is scrambled using encryption algorithms, making it unreadable to anyone who intercepts the communication without the appropriate decryption key. This service prevents eavesdropping and unauthorized access to sensitive information.

2 **Integrity and Authenticity:** The SSL Record Protocol also ensures the integrity and authenticity of the data being transmitted. It achieves this by adding a cryptographic hash (HMAC) to the data before encryption. This hash allows the recipient to verify that the data hasn't been tampered with during transmission. Additionally, SSL/TLS uses digital certificates to authenticate the identities of the communicating parties, ensuring that you are indeed connecting to the intended server and not a malicious imposter.

Handshake Protocol

The Handshake Protocol is instrumental in establishing secure sessions between a client and a server. It allows both parties to mutually authenticate each other through a series of message exchanges. The protocol progresses through four distinct phases:



Phase 1: During this initial phase, both the client and the server send "hello" packets to each other. These packets contain essential information such as the IP session details, chosen cipher suite, and protocol version. This exchange is crucial for setting up the foundation of security.

Phase 2: In the second phase, the server takes the lead by transmitting its certificate and its key exchange information. The server's role in this phase concludes with the dispatch of a "Server-hello-end" packet, marking the end of its contribution.

Phase 3: The third phase involves the client's response to the server. The client forwards its certificate and key exchange details to the server during this stage.

Phase 4: The final phase encompasses the execution of the "Change Cipher Suite" procedure. This pivotal step signifies the transition to an encrypted communication state. Following this phase, the Handshake Protocol concludes, paving the way for secure data transmission.

Change-cipher Protocol

The Change Cipher Spec Protocol is closely integrated with the SSL record protocol and plays a crucial role in the SSL/TLS connection setup. Until the Handshake Protocol concludes, the SSL record output remains in a "pending" state. Once the Handshake Protocol is successfully completed, this "pending" state transitions into the "current" state.

The Change Cipher Spec Protocol is simple in nature, consisting of a single message that is just one byte in length. This message can have only one possible value. The primary purpose of this protocol is to trigger the transfer of the data in the "pending" state to become the new "current" state.

In essence, the Change Cipher Spec Protocol serves as a catalyst for moving the SSL/TLS connection from the negotiation phase (Handshake Protocol) to the encrypted data exchange phase (current state), ensuring that the encryption settings agreed upon during the handshake are applied to subsequent communication.

Alert Protocol

The Alert Protocol is an integral part of the SSL/TLS protocol suite, designed to enhance the reliability and communication between a client and a server.

This protocol is responsible for transmitting alert messages between the two parties, conveying crucial information about the status and health of the SSL/TLS connection.

Alert messages generated by the Alert Protocol can encompass a range of situations, including errors, warnings, or notifications. These messages play a vital role in ensuring that both parties are informed about any anomalies that might arise during the course of the communication.

Alert messages serve various purposes, such as signaling issues related to the SSL/TLS connection's security, such as certificate problems or unexpected closures. They also assist in diagnosing and troubleshooting any potential problems that might arise during the communication process.

By employing the Alert Protocol, SSL/TLS connections become more robust and responsive, as both parties are promptly made aware of any potential issues that might impact the integrity, confidentiality, or authenticity of the exchanged data. This ultimately contributes to a safer and more secure communication environment.

Intrusions and Viruses, Firewalls, Intrusion Detection

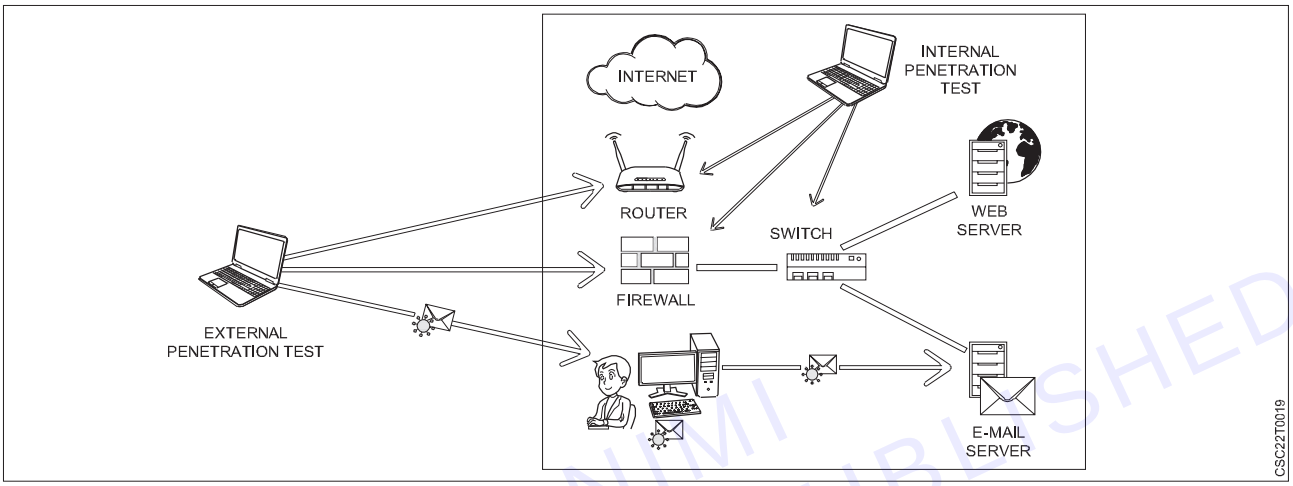
What are Intrusions?

Intrusions, also known as security breaches or cyberattacks, occur when unauthorized individuals or entities gain access to computer systems, networks, or data without permission. These intruders may have malicious intent, such as stealing sensitive information, disrupting services, or causing damage. Detecting and preventing intrusions is a critical aspect of maintaining the security and integrity of digital systems.

Types of Intrusions

1 External Intrusions

2 Internal Intrusions



1 **External Intrusions:** External intrusions, also known as external cyberattacks or external security breaches, refer to unauthorized access and malicious activities initiated by attackers from outside an organization's network or systems. These intrusions can target a wide range of entities, including businesses, government agencies, and individuals. The goal of external intrusions is often to compromise data, steal sensitive information, disrupt services, or cause damage to the targeted organization.

- **Brute Force Attacks:** A brute force attack is a cybersecurity attack method in which an attacker attempts to gain access to a system, network, or account by systematically trying all possible combinations of passwords or encryption keys until the correct one is found. This method relies on the attacker's ability to automate the process of trying numerous combinations quickly and efficiently.
- **Denial of Service (DoS) Attacks:** Attackers overwhelm a system with excessive traffic or requests, causing it to become unavailable.
- **Phishing:** Attackers use deceptive emails or websites to trick users into revealing sensitive information, such as login credentials.

2 **Internal Intrusions:** Internal intrusions, also known as insider threats, occur when individuals with authorized access to an organization's systems, networks, or data misuse their privileges for malicious purposes. Unlike external intrusions, which involve attackers from outside the organization, internal intrusions involve individuals who are already part of the organization. These individuals could be employees, contractors, partners, or anyone with legitimate access to the organization's resources.

Internal intrusions can be particularly damaging due to the insider's familiarity with the organization's systems, processes, and sensitive information. There are two main categories of insider threats:

- **Malicious Insiders:** These are individuals who intentionally misuse their access for personal gain, harm the organization, or engage in activities that are against the organization's interests. Motivations for malicious insiders can include financial gain, revenge, ideology, or a desire to sell sensitive information.
- **Negligent Insiders:** Negligent insiders are individuals who unintentionally cause security breaches due to carelessness, lack of awareness, or inadequate training. They might inadvertently share sensitive information, click on phishing emails, or mishandle data.

Examples of Internal Intrusions:

1 **Data Theft:** An employee with access to sensitive customer information steals this data to sell or use for personal gain.

2 **Sabotage:** A disgruntled employee intentionally disrupts critical systems or services to cause harm to the organization.

3 **Unauthorized Access:** An insider uses their privileges to access information or systems beyond their job responsibilities.

4 **Unintentional Data Exposure:** An employee inadvertently sends sensitive information to the wrong recipients or leaves confidential documents in a public area.

5 **Insider Trading:** In the context of financial markets, employees or individuals with access to confidential financial information trade securities based on that information before it becomes public.

Mitigating Internal Intrusions:

To address internal intrusions, organizations can implement the following measures:

1 **Access Controls:** Implement the principle of least privilege, where individuals are given the minimum access required to perform their job tasks.

2 **User Monitoring:** Implement monitoring systems that track and log user activities to detect unusual or unauthorized behavior.

3 **User Behavior Analytics:** Use advanced analytics to detect anomalies in user behavior that might indicate malicious intent.

4 **Regular Training:** Provide cybersecurity awareness training to employees to educate them about security best practices and the potential risks of insider threats.

5 **Whistleblower Programs:** Establish mechanisms for employees to report suspicious activities without fear of retaliation.

6 **Separation of Duties:** Divide tasks and responsibilities among multiple individuals to prevent a single individual from having excessive control.

7 **Data Loss Prevention (DLP):** Implement DLP tools to monitor and control the movement of sensitive data within and outside the organization.

8 **Incident Response Plan:** Develop a plan to respond to insider threats, including protocols for investigating and addressing incidents.

By combining technical controls, policies, user education, and monitoring, organizations can reduce the risk of internal intrusions and effectively manage insider threats to their systems, data, and operations.

Preventing and Responding to Intrusions:

- **Security Measures:** Implement a robust set of security measures, including firewalls, intrusion detection/prevention systems, access controls, and encryption.
- **Regular Updates:** Keep all software, operating systems, and applications up-to-date with the latest security patches.
- **User Training:** Educate users about security best practices, such as recognizing phishing emails and avoiding suspicious downloads.
- **Multi-Factor Authentication (MFA):** Require multiple forms of verification for accessing sensitive systems or data.
- **Incident Response Plan:** Develop a well-defined plan to respond to security incidents effectively. This includes isolating affected systems, analyzing the extent of the breach, and notifying relevant parties.
- **Monitoring and Logging:** Regularly monitor network and system logs to detect unusual activities. Timely detection can help mitigate potential damage.
- **Vulnerability Management:** Regularly assess and address vulnerabilities within the organization's infrastructure.

- **Security Audits:** Conduct regular security assessments and penetration testing to identify weaknesses before attackers do.

Intrusion Detection

Intrusion Detection refers to the process of monitoring computer networks or systems to detect unauthorized access, malicious activities, and potential security breaches. It involves analyzing network and system data in real-time to identify suspicious or anomalous behavior that could indicate a cyberattack or unauthorized activity. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) are used to implement these detection mechanisms. Here's an overview of intrusion detection:

1 **Types of Intrusion Detection Systems:**

- **Host-Based IDS (HIDS):** Monitors activities on a single host or device, analyzing system logs, file changes, and other host-specific information.
- **Network-Based IDS (NIDS):** Monitors network traffic, analyzing packets and network data to identify patterns of behavior that match known attack signatures or abnormal activities.
- **Anomaly-Based IDS:** Creates a baseline of normal behavior and then identifies deviations from this baseline, alerting when behavior falls outside established norms.
- **Signature-Based IDS:** Compares observed data against a database of known attack patterns or signatures to identify and alert about specific threats.

2 **Detection Techniques:**

- **Signature-Based Detection:** Matches patterns or signatures of known attacks to identify threats.
- **Anomaly-Based Detection:** Identifies deviations from established normal behavior patterns.
- **Heuristic Detection:** Employs rules and algorithms to identify potentially malicious activities.
- **Behavioral Detection:** Observes user and system behaviors to detect suspicious actions.

3 **Alerts and Responses:**

- When an intrusion is detected, the IDS generates alerts or notifications to inform system administrators or security personnel.
- Intrusion Prevention Systems (IPS) can take automated actions to block or mitigate detected threats, such as blocking network traffic from suspicious IP addresses.

4 **Benefits:**

- **Early Detection:** Allows for prompt response to potential security breaches, minimizing the impact of attacks.
- **Real-Time Monitoring:** Provides continuous monitoring of network and system activities.
- **Reduced Downtime:** Enables rapid identification and containment of threats, reducing downtime and system disruptions.

5 **Challenges:**

- **False Positives:** IDS may generate alerts for legitimate activities that resemble attack patterns.
- **False Negatives:** Some sophisticated attacks may evade detection by known signatures or patterns.
- **Complexity:** Configuring and maintaining IDS systems can be complex and resource-intensive.
- **Performance Impact:** Intensive monitoring and analysis can impact system performance.

Intrusion Detection plays a critical role in enhancing cybersecurity by identifying potential threats and facilitating timely responses to mitigate risks. It is an integral part of a comprehensive cybersecurity strategy aimed at safeguarding digital assets, data, and systems from unauthorized access and attacks.

Virus (Vital Information Resources Under Seize)

Essential Information Compromised: A computer virus is a piece of code or software that can have detrimental effects on your computer data by either corrupting or completely destroying it. These viruses can rapidly create copies of themselves and distribute them throughout various folders, resulting in harm to your computer's data. In

reality, a computer virus is a form of malicious software or “malware” that, upon infecting your system, duplicates itself by altering other computer programs and implanting its own code.

The methods through which viruses impact computers and devices include:

- **Downloading Files Online:** When files are downloaded from the internet, viruses can infiltrate and infect the system.
- **Media or Drive Removal:** Viruses can spread when removable media or drives are connected to infected systems and then introduced to other devices.
- **Pen Drives:** Infections can occur through the use of infected pen drives or USB devices, which carry the virus from one system to another.
- **Email Attachments:** Viruses often arrive as attachments in emails, allowing them to enter systems once the attachments are opened.
- **Unpatched Software & Services:** Vulnerabilities in software and services that haven’t been updated with the latest patches can be exploited by viruses.
- **Weak Administrator Passwords:** Viruses can take advantage of weak or unprotected administrator passwords to gain unauthorized access and spread.

The effects of a virus on a computer system include:

- **Disruption of Normal Functionality:** Viruses can interfere with the regular operations of the targeted computer system.
- **Disruption of Network Usage:** The presence of a virus can disrupt the system’s ability to access and use network resources.
- **Alteration of Configuration Settings:** Viruses can modify the settings and configurations of the system, potentially leading to instability and unexpected behavior.
- **Data Destruction:** Viruses can destroy or corrupt data stored on the infected computer, causing irreversible loss.
- **Disruption of Network Resources:** The virus’s impact can extend to disrupting resources shared across a computer network.
- **Confidential Data Destruction:** Viruses can target and destroy sensitive and confidential data, jeopardizing privacy and security.

A computer virus attack can manifest through several noticeable signs. Here are a few examples:

- **Increased Pop-Up Windows:** You might experience a surge in pop-up windows appearing on your screen. These pop-ups could urge you to visit unfamiliar websites or prompt you to download software, potentially malicious.
- **Homepage Alteration:** Your usual homepage could be replaced by a different website without your consent. Moreover, you might find it challenging to restore your original homepage settings.
- **Unauthorized Email Activity:** Your email account might exhibit abnormal behavior, such as sending out a large number of emails without your knowledge. Criminals could gain control over your account or manipulate it to send emails from another compromised computer.
- **Frequent System Crashes:** A virus can cause significant harm to your hard drive, resulting in device freezes or crashes. In severe cases, your device might not restart at all.
- **Unusual Sluggishness:** If your computer’s processing speed suddenly decreases, it could indicate the presence of a virus affecting its performance.
- **Unrecognized Startup Programs:** You might notice unfamiliar programs launching when you start your computer. This anomaly could become evident as you power up your device or by reviewing the list of active applications.
- **Unexpected Activities like Password Changes:** Unauthorized actions, such as changes to your passwords, can occur due to a virus attack. This may lead to difficulties in accessing your computer.

Firewalls

A firewall is a network security device or software application that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Its main purpose is to establish a barrier between a trusted internal network and untrusted external networks, such as the internet, to prevent unauthorized access and protect sensitive data.

Firewalls work by examining network packets and applying rules to determine whether to allow or block the traffic. There are several types of firewalls, each with its own approach to filtering traffic:

1 **Packet Filtering Firewall:** This type of firewall examines packets of data and compares their attributes, such as source and destination IP addresses, port numbers, and protocol types, against a set of predefined rules. It then decides whether to allow or deny the packet based on these rules.

2 **Stateful Inspection Firewall:** Also known as dynamic packet filtering, this firewall not only considers individual packets but also keeps track of the state of active connections. It monitors the state of connections and ensures that only legitimate traffic associated with an established connection is allowed through.

3 **Proxy Firewall:** A proxy firewall acts as an intermediary between internal and external networks. It receives and forwards traffic on behalf of the internal network, effectively hiding internal network details. This adds an extra layer of security by preventing direct connections between external entities and the internal network.

4 **Application-layer Firewall:** This type of firewall operates at the application layer of the OSI model. It can understand specific application protocols and make decisions based on the actual content of the traffic. This allows for more granular control and the ability to block or allow specific application functions or commands.

5 **Next-Generation Firewall (NGFW):** NGFWs combine traditional firewall functionality with additional features such as intrusion detection and prevention, deep packet inspection, and application awareness. They aim to provide more advanced threat detection and prevention capabilities.

6 **Unified Threat Management (UTM):** UTM appliances integrate multiple security features into a single device. These features can include firewalling, antivirus, intrusion detection/prevention, content filtering, and more.

Firewalls can be deployed at various points within a network architecture, including:

- **Perimeter/Front-end Firewalls:** These protect the network from external threats, typically placed at the boundary between an internal network and the internet.
- **Internal Firewalls:** Placed within the internal network, these segment different parts of the network to contain potential breaches and limit the spread of threats.
- **Host-based Firewalls:** Installed on individual devices (such as computers or servers), these firewalls control traffic at the device level and can be customized for specific security needs.

The classification of a firewall as either hardware or software can be a source of confusion. As previously mentioned, firewalls exist in both forms: as network security devices and as software applications on computers. Thus, the distinction between the two isn't absolute, and having both can be beneficial.

While hardware and software firewalls share the same goal, they function differently due to their respective formats. A hardware firewall is a tangible device situated between a computer network and a gateway, like a broadband router. Conversely, a software firewall is a program installed on a computer, operating through port numbers and interactions with installed software.

Additionally, there are cloud-based firewalls often referred to as Firewall-as-a-Service (FaaS). One key advantage of these cloud-based solutions is their centralized management. Similar to hardware firewalls, cloud-based options excel at delivering perimeter security.

In essence, the distinction between hardware and software firewalls isn't always clear-cut, as both forms contribute to network security, albeit through varying mechanisms.

Cyber security systems & cyber laws

Cybersecurity systems refer to the technologies, processes, and practices implemented to protect computer systems, networks, and data from various forms of cyber threats. These threats can include unauthorized access, data breaches, malware infections, phishing attacks, and more. Cybersecurity systems play a critical role in maintaining the confidentiality, integrity, and availability of digital assets and information.

Types of Cyber Security

Cybersecurity encompasses a wide range of practices, technologies, and measures designed to protect computer systems, networks, and data from cyber threats and attacks. There are various types of cybersecurity that focus on different aspects of protection. Here are some of the main types:

1 Network Security: Network security focuses on protecting the integrity, confidentiality, and availability of a network and its data. This involves measures like firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), virtual private networks (VPNs), and network segmentation.

2 Endpoint Security: Endpoint security involves securing individual devices (endpoints) like computers, smartphones, and tablets. This is achieved through antivirus software, anti-malware solutions, and other tools to prevent, detect, and respond to threats on these devices.

3 Application Security: Application security focuses on securing software applications and the code they are built upon. This includes identifying and addressing vulnerabilities in software to prevent exploitation by attackers.

4 Cloud Security: As more data and services move to the cloud, cloud security becomes crucial. It involves securing data, applications, and infrastructure hosted in cloud environments, and ensuring proper access controls and encryption.

5 Data Security: Data security involves protecting sensitive data from unauthorized access, theft, or breaches. This can include encryption, access controls, data masking, and data loss prevention (DLP) solutions.

6 Identity and Access Management (IAM): IAM is about ensuring that only authorized individuals have access to the appropriate resources. It includes techniques like multi-factor authentication (MFA), single sign-on (SSO), and user access management.

7 Incident Response: Incident response is the process of managing and mitigating the consequences of a cybersecurity incident. It involves identifying, containing, eradicating, and recovering from attacks to minimize damage and restore normal operations.

8 Security Operations Center (SOC): A SOC is a centralized unit that monitors and responds to security threats in real-time. It uses advanced tools and technologies to detect, analyze, and respond to incidents.

9 Vulnerability Management: This involves identifying and addressing vulnerabilities in software and systems before they can be exploited by attackers. Regular vulnerability assessments and patch management are key components.

10 Penetration Testing: Also known as ethical hacking, penetration testing involves simulating cyberattacks to identify vulnerabilities and weaknesses in systems and networks. This helps organizations proactively address these issues.

11 Physical Security: Physical security is about protecting the physical assets of an organization, such as data centers and hardware, from unauthorized access, theft, and damage.

12 Mobile Security: As mobile devices become more prevalent, mobile security focuses on protecting smartphones, tablets, and other mobile devices from malware, data theft, and unauthorized access. These are just some of the many facets of cybersecurity. Organizations often adopt a multi-layered approach, combining various types of cybersecurity measures to create a comprehensive security strategy that addresses a wide range of potential threats.

Why is cybersecurity important?

In today's interconnected world, advanced cyberdefense programs bring benefits to all. On an individual level, a cybersecurity attack can lead to severe consequences ranging from identity theft to extortion attempts and even the loss of precious data such as family photographs. The reliance on critical infrastructure, including power plants, hospitals, and financial service firms, is universal. Securing these vital entities is paramount to maintaining the functioning of our society.

Moreover, the efforts of cyberthreat researchers play a crucial role in benefiting everyone. For instance, the team of 250 threat researchers at Talos engages in the investigation of emerging threats and strategies for cyberattacks. Their work includes identifying new vulnerabilities, enlightening the public about the significance of cybersecurity, and fortifying open source tools. The impact of their endeavors extends to making the internet a safer space for all users.

Types of Cyber Security Threats

A cybersecurity threat refers to the malevolent actions undertaken by individuals or groups with the intent to compromise, steal, or manipulate data, breach network security, or cause disruption within the digital realm. Contemporary cybersecurity experts identify the following prevailing threats.

Cybersecurity Threat	Description
Malware	Malware includes various malicious software types such as viruses, worms, Trojans, and ransomware, designed to infect, damage, or gain unauthorized access.
Phishing	Phishing involves tricking users into revealing sensitive information or clicking on malicious links through deceptive emails or messages.
Denial of Service (DoS)	DoS attacks overwhelm a system, network, or website with excessive traffic, causing it to become unavailable to legitimate users.
Distributed DoS (DDoS)	DDoS attacks involve multiple systems flooding a target with traffic, making it even more difficult to mitigate and recover from the attack.
Man-in-the-Middle (MitM)	In MitM attacks, attackers intercept and possibly alter communications between two parties without their knowledge, compromising data integrity.
SQL Injection	SQL injection attacks exploit vulnerabilities in web applications by injecting malicious SQL code, potentially allowing unauthorized database access.
Zero-Day Exploits	Zero-day exploits target unpatched vulnerabilities in software before a fix is available, giving attackers the advantage of exploiting unknown weaknesses.
Ransomware	Ransomware encrypts a victim's data and demands a ransom for decryption. Paying the ransom is discouraged as it may not guarantee data recovery.
Social Engineering	Social engineering manipulates individuals into revealing confidential information, often relying on psychological tactics and deception.
Insider Threats	Insider threats come from individuals within an organization who misuse their access to steal data, commit fraud, or intentionally cause harm.
Malvertising	Malvertising delivers malicious code through legitimate-looking online advertisements, potentially infecting users who interact with the ads.
IoT Vulnerabilities	Internet of Things (IoT) devices with inadequate security can be compromised, leading to breaches or even the takeover of connected devices.
Data Breaches	Data breaches involve unauthorized access to sensitive information, resulting in the exposure or theft of personal or proprietary data.
Drive-By Downloads	Drive-by downloads occur when malware is downloaded and installed on a user's system without their consent, often through malicious websites.
Cryptojacking	Cryptojacking involves the unauthorized use of a victim's computer to mine cryptocurrencies, slowing down the system and using up resources.
Botnets	Botnets are networks of compromised devices controlled by attackers, often used to carry out coordinated attacks or distribute malware.
Eavesdropping	Eavesdropping attackers intercept and listen in on communication between parties, potentially gaining access to sensitive information.
Brute Force Attacks	Brute force attacks use trial-and-error to guess passwords or encryption keys, exploiting weak credentials to gain unauthorized access.
Keyloggers	Keyloggers record keystrokes on compromised systems, capturing sensitive information such as login credentials and credit card numbers.
Drive Encryption Exploits	Attackers target vulnerabilities in drive encryption mechanisms to gain unauthorized access to encrypted data on compromised devices.

Cyber Laws (IT Law)

Cyber laws, also known as IT laws or information technology laws, in India pertain to the legal framework governing various aspects of electronic communication, online transactions, digital signatures, cybersecurity, and other technology-related matters. The primary legislation that addresses cyber laws in India is the Information Technology Act, 2000, along with its subsequent amendment

Advantages of Cyber Law(IT-Law)

1 **Facilitating E-Commerce:** The legal framework provided by cyber law enables organizations to conduct e-commerce activities with legal certainty and protection.

2 **Validity of Digital Signatures:** The Act confers legal recognition and validity to digital signatures, enhancing the credibility and authenticity of electronic transactions.

3 **Entry of Corporate Certifying Authorities:** Corporate entities are empowered to become Certifying Authorities, which can issue Digital Signature Certificates, promoting a broader range of options for digital identity verification.

4 **Promoting E-Governance:** The Act allows governmental bodies to issue notifications online, fostering e-governance practices and streamlining administrative processes.

5 **Streamlined Documentation:** Organizations are authorized to submit various documents and applications electronically to government offices or agencies, using prescribed e-forms, thus simplifying administrative procedures.

6 **Addressing Security Concerns:** The Act addresses critical security issues relevant to electronic transactions, enhancing the overall trust and reliability of online interactions.

7 **Comprehensive Security Measures:** Cyber law encompasses both hardware and software security measures, ensuring a holistic approach to safeguarding digital transactions and communications.

Threat	ACT
Primary Legislation	Information Technology Act,2000 (IT Act) and its subsequent amendments
Digital Signatures	Legal recognition of digital signatures for authentication of electronic transactions
Data Protection and privacy	Guidelines for the protection of personal data and privacy
Cybercrimes	Definition of cybercrimes and penalities for unauthorized access, hacking, etc.
Cybersecurity	Encouragement of Cybersecurity practices and mechanisms
Cyber Appellate Tribunal	Establishments for appeals against decisions related to cyber matters
Intermediary Liability	Responsibilities and liabilities of intermediaries for content hosting and transmission
Electronic Contracts	Recognition of validity and enforcement of electronic contracts
CERT-in	National agency for coordinating responses to cybersecurity incidents

MODULE 2 : Database Concepts

LESSON 18 - 36 : Database concepts

Objectives

At the end of this lesson, you will be able to:

- understand basic concepts & applications of datebase system
- create & mange database file using MYSQL
- handle, store and organise facilitating information access and analysis.

Concept of DBMS, RDBMS

What is Database?

A database is a compilation of interconnected data that enables efficient operations such as data retrieval, insertion, and deletion.

It serves as a means to systematically arrange information, often in the format of tables, schemas, views, and reports.

For instance, consider a college database. This database arranges information concerning administrators, staff members, students, and faculty in a coherent manner. By utilizing this database, you can seamlessly retrieve, add, and remove information as needed.

Concept of DBMS and RDBMS

Database Management System

A Database Management System (DBMS) is software designed for the administration of databases. Widely used commercial databases like MySQL and Oracle serve various applications. The DBMS offers an interface to execute diverse tasks, such as crafting databases, storing data, data updates, and table creation.

This system ensures database security and protection, along with maintaining data consistency when multiple users are involved.

The DBMS empowers users to undertake the following tasks:

1 **Database Creation**: DBMS enables the creation of new databases to organize and store data efficiently.

2 **Data Storage**: It provides a platform for storing a wide range of data types, ensuring data integrity and accessibility.

3 **Data Retrieval:** Users can effortlessly retrieve specific information from the database using queries and search operations.

4 **Data Update and Modification**: DBMS facilitates the modification and updating of existing data, ensuring accuracy and relevance.

5 **Table Creation**: Users can define and create tables with specific structures to store data in a structured manner.

6 **Data Integrity**: It enforces rules and constraints to maintain the integrity and consistency of the data.

7 **Data Security**:** DBMS offers security features to control access to the data, protecting it from unauthorized users.

8 **Data Backup and Recovery:** Users can perform data backups regularly and recover data in case of system failures or errors.

9 **Indexing**: DBMS allows the creation of indexes, enhancing data retrieval performance.

10 **Query Optimization**: It optimizes queries to improve the efficiency of data retrieval operations.

11 **Concurrency Control**: In multi-user environments, DBMS ensures that multiple users can access and modify data simultaneously without conflicts.

12 **Transaction Management**: DBMS supports transactions, ensuring that a series of operations are completed successfully or not at all.

13 **Data Reporting**: Users can generate reports and analyze data to extract valuable insights.

14 **User Management**: DBMS enables the management of user roles, permissions, and access levels to maintain data security.

15 **Data Relationships**: It allows users to establish relationships between different tables, facilitating complex data retrieval and analysis.

16 **Data Validation**: DBMS enforces data validation rules to ensure that entered data meets specified criteria.

17 **Data Sharing**: It enables data sharing across different applications and users while maintaining data integrity.

18 **Data Migration**: Users can transfer data between different databases or systems using DBMS tools.

19 **Data Archiving**: Older or less frequently used data can be archived to free up space while retaining accessibility.

20 **Data Auditing:** DBMS tracks changes and activities related to the database for auditing and compliance purposes.

Characteristics of DBMS

A Database Management System (DBMS) is a software application that facilitates the creation, maintenance, and manipulation of databases. It acts as an intermediary between users or applications and the actual physical database, providing an organized and controlled environment for storing and retrieving data. Here are some key characteristics of a DBMS:

1 **Data Abstraction**: A DBMS provides a level of abstraction that hides the complex underlying details of how data is stored and managed. This allows users and applications to interact with the database without needing to understand its internal workings.

2 **Data Integrity**: DBMS systems enforce data integrity rules to ensure that the data stored in the database remains accurate and consistent. This involves maintaining constraints, such as uniqueness, referential integrity, and data validation, to prevent incorrect or conflicting data from being stored.

3 **Data Security**: DBMS systems offer various security mechanisms to control access to the database. This includes user authentication, authorization, and access control, ensuring that only authorized users can perform specific actions on the data.

4 **Data Independence**: DBMS systems provide a separation between the logical structure of the database and its physical storage. This means that changes to the physical storage (such as moving to a different storage device) can be made without affecting the way users and applications access the data.

5 **Data Consistency**: A DBMS ensures that data remains consistent even when multiple users or applications are accessing and modifying it simultaneously. Techniques like transactions and locking mechanisms are used to maintain data consistency.

6 **Query Language**: DBMS systems provide a structured query language (SQL) that allows users to interact with the database using standardized commands. SQL enables users to retrieve, manipulate, and manage data stored in the database.

7 **Data Redundancy Elimination**: DBMS systems help in reducing data redundancy by providing features like normalization. This minimizes data duplication and improves data consistency.

8 **Concurrent Access and Transaction Management:** DBMS systems handle concurrent access to the database by multiple users or applications. Transaction management ensures that a series of database operations are treated as a single, indivisible unit, ensuring data integrity and consistency.

9 **Backup and Recovery**: DBMS systems offer mechanisms for data backup and recovery. Regular backups help prevent data loss in case of hardware failures, software errors, or other unexpected events.

10 **Scalability**: A well-designed DBMS can be scaled to accommodate increasing amounts of data and growing user loads. This can involve adding more hardware resources or optimizing the database structure.

11 **Data Relationships**: DBMS systems support the establishment and management of relationships between different sets of data, allowing for the creation of complex data structures and efficient retrieval of related information.

12 **Performance Optimization**: DBMS systems often include query optimization techniques to improve the performance of database operations. This involves choosing the most efficient execution plans for complex queries.

13 **Data Dictionary**: A data dictionary is a part of the DBMS that stores metadata, which includes information about the structure of the database, data types, relationships, and constraints. This helps users and applications understand the database's schema.

Advantages of DBMS

Manages Data Redundancy: By centralizing data within a single database file, DBMS effectively controls redundancy in recorded information.

Facilitates Data Sharing: Authorized users within an organization can seamlessly share data among multiple individuals through the DBMS.

Simplified Maintenance: The centralized nature of the database system makes maintenance notably simpler and more manageable.

Time Efficiency: DBMS expedites development processes and diminishes maintenance demands, leading to time savings.

Enables Backup: Incorporating backup and recovery subsystems, DBMS automatically backs up data in the event of hardware or software failures and facilitates data restoration as needed.

Supports Multiple User Interfaces: DBMS offers various user interfaces, including graphical user interfaces and application program interfaces, catering to diverse user preferences.

Disadvantages of DBMS

Expense for Hardware and Software: Running DBMS software demands robust data processing capabilities and substantial memory capacity, incurring costs for hardware upgrades.

Space Consumption: DBMS occupies considerable disk space and memory to ensure efficient operations.

Increased Complexity: Implementing a database system introduces added complexity and prerequisites.

Heightened Vulnerability to Failure: Database failures have a substantial impact, particularly in organizations where all data resides within a single database. Incidents like power outages or database corruption could lead to permanent data loss.

Relational Database Management System

RDBMS stands for Relational Database Management System.

RDBMS is an abbreviation for Relational Database Management Systems.

It serves as software that enables the creation, modification, and maintenance of relational databases.

A relational database is a type of system for storing and retrieving data presented in a structured table layout composed of rows and columns.

It's a specific component of Database Management Systems (DBMS) conceptualized by E.F. Codd in the 1970s. The fundamental principles of relational DBMS form the basis for prominent database systems like SQL, MySQL, and Oracle.

How it works

- Information is portrayed in the form of rows known as tuples within RDBMS.
- The prevailing choice for databases is the relational type. It comprises numerous tables, each endowed with its unique primary key.
- The structured assembly of tables facilitates seamless data retrieval within RDBMS.

table/Relation

All contents within a relational database are organized into relations. RDBMS databases utilize tables for data storage. A table constitutes an assembly of interconnected data elements, arranged in rows and columns. These tables symbolize real-world entities, like individuals, locations, or occurrences, for which data is accumulated. The systematic arrangement of data within a relational table embodies the conceptual portrayal of the database.

Properties of a Relation

- Every relation within the database possesses an exclusive name for distinct identification.
- Within a relation, duplication of tuples is prohibited.
- Tuples within a relation are unordered.
- All attributes contained in a relation are indivisible; each cell comprises a singular value.
- A table serves as a fundamental illustration of data storage within RDBMS

Benefits(Advantages)

1 **Ease of Management**: Independent manipulation of tables simplifies database management without impacting others.

2 **Enhanced Security**: Multiple layers of security ensure controlled data access and sharing.

3 **Flexibility**: Centralized data updating prevents the need for modifications across various files. Expanding the database to include more records is straightforward, ensuring scalability. SQL queries can be applied easily.

4 **User Support**: RDBMS accommodates multiple users through a client-side architecture.

5 **Efficient Data Handling**

- Rapid data retrieval due to the relational design.
- Keys, indexes, and normalization minimize data redundancy.
- ACID properties ensure data consistency during transactions.

6 **Large Data Storage and Retrieval**: RDBMS facilitates handling vast data volumes.

7 **Effortless Data Handling**

- Swifter data fetching resulting from relational structure.
- Keys, indexes, and normalization principles avert data redundancy.
- Data consistency maintained through ACID properties in transactions.

8 **Fault Tolerance**: Database replication permits simultaneous access and aids system recovery during crises such as power outages or abrupt shutdowns.

Drawbacks(Disadvantages)

1 **High Costs and Infrastructure**: RDBMS demand substantial investments in terms of both expenses and infrastructure to establish and maintain their operations.

2 **Scalability Challenges**: Expanding data necessitates additional servers, increased power, and memory resources, which can be complex and costly.

3 **Complexity**: Large datasets can result in intricate relationships that might impede comprehension and even decrease performance.

4 **Structured Limits**: Relational databases have predefined limits on fields or columns, which could lead to data truncation or loss.

Data Models, Concept of DBA, Database Users

Data Types

Data types define the kind of data that can be stored in a column of a database table. Different database management systems (DBMS) might support slightly different sets of data types, but here are some common categories:

1 **Numeric Types**

- **Integer**: Whole numbers (e.g., 1, -5, 100).
- **Float/Double**: Approximate decimal numbers with a specified precision (e.g., 3.14, -0.001).

2 **Character Strings**

- **Char**: Fixed-length character string (e.g., 'Hello').
- **Varchar**: Variable-length character string (e.g., 'StudentName').

3 **Date and Time Types**

- **Date:** Represents a date (e.g., '2023-08-31').
- **Time**: Represents a time (e.g., '15:30:00').
- **Timestamp**: Represents a date and time (e.g., '2023-08-31 15:30:00').

4 **Boolean Type**

- **Boolean**: Represents true or false values.

5 **Binary Types**

- **Blob (Binary Large Object)**: Stores binary data, such as images or files.

6 **Enumeration Types**:

- **Enum**: Represents a predefined set of values (e.g., 'red', 'green', 'blue').

7 **Composite Types**

- **JSON or XML**: Stores structured data in JSON or XML format.

Data Models

In the realm of Database Management Systems (DBMS), a data model refers to a set of tools designed to condense the database's description. Data models offer a clear representation of data, aiding in the creation of an effective database. They guide us from conceptualizing data design to its accurate implementation.

Types of Relational Models

1 Conceptual Data Model

2 Representational Data Model

3 Physical Data Model

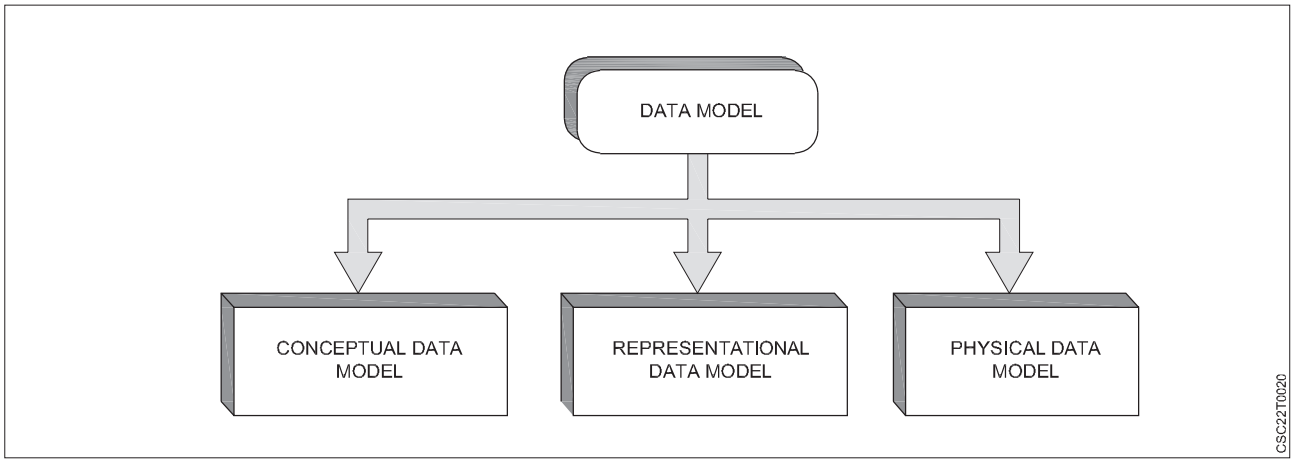
1 **Conceptual Data Model**

The conceptual data model presents a broad overview of the database, focusing on high-level aspects. It serves to comprehend the database's necessities and demands. This model is pivotal during requirement gathering, preceding the actual database design by Database Designers. A prominent example of this approach is the entity/relationship (ER) model. The ER model delves into entities, relationships, and attributes, providing a foundation for database designers. Moreover, this concept facilitates discussions with non-technical stakeholders, aiding in understanding and addressing their requirements.

1 **Entity-Relationship Model (ER Model)**: The ER Model stands as a top-tier data model employed to outline data and their interconnections. Essentially, it serves as a conceptual blueprint for databases, enabling an uncomplicated depiction of data perspectives.

Components of ER Model

1 **Entity**: An entity represents a real-world object, whether it's a person, place, concept, or item. In an ER Diagram, entities are depicted using rectangles.



2 **Attributes**: Attributes provide descriptions or characteristics of an entity. These can include properties like name, age, roll number, or any other relevant information. In an ER Diagram, attributes are symbolized using ovals or ellipses.

3 **Relationship**: Relationships define connections between various entities. They represent how different entities are related to each other in the real world. In an ER Diagram, relationships are indicated using diamonds or rhombuses.

2 **Representational Data Model**

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is a Relational model. The relational Model consists of Relational Algebra and Relational Calculus. In the Relational Model, we basically use tables to represent our data and the relationships between them.

3 **Physical Data Model**

The Physical Data Model serves the practical implementation of the Relational Data Model. In the end, all database data is physically stored on secondary storage devices like disks and tapes. This storage occurs in the form of files, records, and other specific data structures. This model encompasses details about file formats, database structure, external data structure presence, and their interrelationships.

Advantages of Data Models

1 **Accurate Data Representation**: Data models ensure precise and structured representation of data, enhancing its clarity and organization.

2 **Data Integrity and Minimized Redundancy**: Data models aid in identifying missing data and reducing redundancy by maintaining consistent and non-repetitive data entries.

3 **Enhanced Data Security**: Data models contribute to improved data security measures, safeguarding information against unauthorized access or tampering.

4 **Effective Physical Database Creation**: A well-designed data model should offer sufficient detail to serve as a foundation for constructing the physical database, ensuring alignment between the conceptual and physical levels.

5 **Relationship Definition**: Data models support the delineation of relationships between tables, as well as the establishment of primary and foreign keys. This is crucial for maintaining data consistency and integrity.

6 **Stored Procedure Definition**: The information encapsulated in a data model can be utilized to define stored procedures, enabling efficient data manipulation and retrieval.

Disadvantages of Data Models

1 **Complexity with Large Databases**: In the context of extensive databases, comprehending the intricacies of the data model can become challenging and overwhelming.

2 **SQL Proficiency Required**: Proper knowledge of SQL is essential to effectively utilize and manipulate physical models. This requirement can be a barrier for users lacking SQL expertise.

3 **Impact of Structural Changes**: Even minor alterations to the data model's structure can necessitate significant modifications throughout the associated application, potentially causing disruptions and increased maintenance efforts.

4 **Lack of Standard Data Manipulation Language**: Unlike SQL, which is a standardized language for database manipulation, data models don't inherently provide a universally accepted data manipulation language.

5 **Requirement for Knowledge of Physical Data Storage**: Creating a data model necessitates a comprehensive understanding of the characteristics of how data is physically stored, which can present a learning curve and barrier for some users.

DBA(Database Administrator)

A Database Administrator (DBA) is an individual responsible for overseeing, maintaining, coordinating, and operating a database management system. Their primary role involves managing, securing, and ensuring the smooth functioning of database systems. They hold the authority to grant database access, coordinate tasks, plan for capacity, install and monitor software, and procure hardware resources as required. Their responsibilities encompass a wide range of tasks, including configuration, database design, migration, security implementation, troubleshooting, backup management, and data recovery. Database administration plays a vital and central role within any organization relying on one or multiple databases. DBAs serve as the overall leaders and supervisors of the database system.

Types of Database Administrator (DBA)

There are several types of Database Administrators (DBAs), each specializing in different aspects of managing and maintaining databases. Here are some common types:

1 **Administrative DBA**: Responsible for overall management of the database server, including tasks such as backup and recovery, security management, performance monitoring, and system maintenance.

2 **Development DBA**: Focuses on designing, developing, and implementing the database structure based on the requirements of applications. They create database objects, optimize queries, and ensure efficient data access.

3 **Data Warehouse DBA**: Specializes in managing data warehouses and data marts. They design and implement large-scale data repositories used for business intelligence and reporting.

4 **Cloud DBA**: Manages databases hosted on cloud platforms. They ensure data availability, performance, and security in cloud environments.

5 **Application DBA**: Works closely with application developers to optimize database performance for specific applications. They handle tasks like query tuning, indexing, and database design for application requirements.

6 **Backup and Recovery DBA**: Focuses on creating and implementing strategies for data backup, recovery, and disaster planning to ensure data availability and minimize downtime.

7 **Performance Tuning DBA**: Specializes in optimizing database performance. They monitor and analyze performance metrics, identifying and resolving performance bottlenecks.

8 **Security DBA**: Concentrates on database security, implementing access controls, encryption, and auditing mechanisms to protect sensitive data from unauthorized access and breaches.

9 **Replication DBA**: Manages database replication processes to ensure data consistency and availability across multiple database instances.

10 **Database Architect**: Designs and plans the overall structure of the database system, including schema design, table relationships, and data integrity. They provide a blueprint for the database implementation.

11 **Disaster Recovery DBA**: Focuses on creating and testing disaster recovery plans to ensure business continuity in case of data loss or system failures.

12 **Migration DBA**: Specializes in migrating data from one database platform to another. They ensure data accuracy, consistency, and minimal downtime during migrations.

13 **Database Compliance DBA**: Ensures that the database system adheres to industry regulations and compliance standards, such as GDPR or HIPAA.

14 **Big Data DBA**: Specializes in managing and optimizing large-scale databases used for big data analytics. They handle distributed databases, NoSQL databases, and data processing frameworks.

15 **NoSQL DBA**: Manages NoSQL databases, which are used for handling unstructured or semi-structured data. They specialize in platforms like MongoDB, Cassandra, and Redis.

Different types of Database Users

Database Administrator (DBA)

- Defines database schema and controls the three levels of the database.
- Creates new user accounts and manages access.
- Ensures database security and authorizes user access.
- Monitors performance, recovery, backup, and provides technical support.
- Responsible for resolving security breaches and performance issues.
- Performs Data Control Language (DCL) operations.
- Has a system or super user account in the DBMS.
- Handles hardware and software failures and repairs damage.
- Manages privileges and access permissions.

Naive / Parametric End Users

- Unsophisticated users who frequently use database applications.
- Lack in-depth DBMS knowledge.
- Commonly interact with databases to perform specific tasks.
- Examples: Railway ticket booking users, bank clerks.

System Analyst

- Analyzes requirements of parametric end users.
- Ensures end users' needs are met.
- Acts as an intermediary between end users and the DBMS.

Sophisticated Users

- Familiar with databases.
- Can develop own database applications.
- Write SQL queries directly through the query processor.
- Often engineers, scientists, business analysts.

Database Designers

- Design database structures including tables, indexes, views, triggers, etc.
- Enforce constraints and relationships in the design.
- Understand requirements of different user groups.
- Create designs that satisfy diverse user needs.

Application Programmers

- Write code for application programs.
- Back-end programmers who develop software.
- Use programming languages like Visual Basic, C, etc.
- Design, debug, test, and maintain programs for users' interaction with databases.

Casual Users / Temporary Users

- Occasional database users seeking new information each time.
- Examples include middle or higher-level managers.

Specialized Users

- Sophisticated users who create specialized database applications.
- Applications might not fit traditional data-processing frameworks.
- Examples include computer-aided design applications.

ER Model & Diagram, Database Schema

The term “ER model” refers to the Entity-Relationship model, which serves as a data model at an elevated level. Its purpose is to establish the definition of data components and their interconnections within a designated system.

It generates a conceptual blueprint for the database, creating a straightforward and easily manageable representation of data.

In the practice of ER modeling, the arrangement of the database structure is visually represented through a diagram termed as an entity-relationship diagram.

Why Use ER Diagrams In DBMS?

ER diagrams play a pivotal role in portraying the E-R model within a database, simplifying the process of transforming them into relational structures (tables).

The significance of ER diagrams lies in their ability to model real-world objects, adding a layer of practicality to their utility.

With no demand for technical prowess or hardware assistance, ER diagrams offer a user-friendly approach.

Even for individuals unfamiliar with complex concepts, these diagrams are highly accessible and uncomplicated to generate.

ER diagrams offer a uniform approach to logically visualizing data, providing a standardized solution.

Symbols Used in ER Model

The ER Model is employed to conceptualize the system’s logical aspect from a data stance, encompassing these symbols

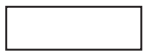
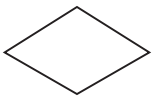
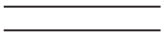
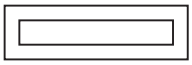
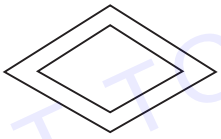
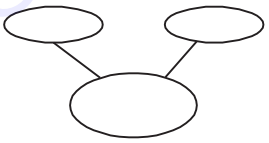
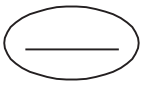
- **Rectangles**: These rectangles are utilized to symbolize Entities within the ER Model.
- **Ellipses**: Ellipses are employed to represent Attributes in the ER Model.
- **Diamonds**: The diamond shape is used to portray Relationships among different Entities.
- **Lines**: Lines are utilized to depict associations between attributes and entities, as well as entity sets with different types of relationships.
- **Double Ellipses**: Double ellipses are used to signify Multi-Valued Attributes.
- **Double Rectangles**: Double rectangles are employed to indicate a Weak Entity.

Components of ER Diagram

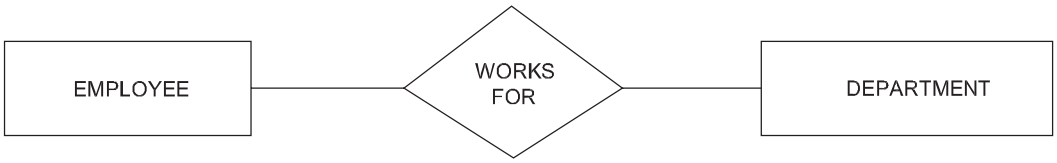
Entity

An entity can encompass various elements such as objects, classes, individuals, or locations. Within an ER diagram, entities are symbolized using rectangular shapes.

Take into account an instance of an organization - a manager, product, staff member, department, and so forth could be regarded as entities in this context.

	REPRESENTS ENTITY
	REPRESENTS ATTRIBUTE
	REPRESENTS RELATIONSHIP
	LINKS ATTRIBUTE (S) SET (S) or ENTITY SET (S) TO RELATIONSHIP
	REPRESENTS MULTIVALUED
	REPRESENTS DERIVED ATTRIBUTES
	REPRESENTS TOTAL PARTICIPATION OF ENTITY
	REPRESENTS WEAK ENTITY
	REPRESENTS WEAK RELATIONSHIPS
	REPRESENTS COMPOSITE ATTRIBUTES
	REPRESENTS KEY ATTRIBUTES / SINGLE VALUED ATTRIBUTES

CSC22T0021



CSC22T0022

1 Weak Entity

A weak entity is an entity that relies on another entity. Unlike a strong entity, a weak entity lacks its own key attribute. It is depicted using a double rectangle in diagrams.

Example: A corporation has the ability to retain data regarding the family members (parents, children, spouse) of an employee. However, these family members only hold significance due to their connection with the employee. Consequently, family members are categorized as a vulnerable entity type, while employees are recognized as the pivotal entity type for the family members, indicating that they are..

2 Strong Entity

A Strong Entity refers to an entity possessing a primary Attribute. It maintains independence from other entities within the Schema, boasting a primary key that facilitates its distinct identification. This is symbolized by a rectangle and is termed as a Strong Entity Type.

Attributes

Attributes are the characteristics that establish the nature of an entity type. For instance, attributes like Roll Number, Name, Date of Birth, Age, Address, and Mobile Number serve to characterize the entity type of Student. Within an Entity-Relationship (ER) diagram, attributes are depicted using oval shapes.

1 Key Attribute

The key attribute is the distinctive characteristic that identifies each individual entity within the set of entities. For instance, the Roll_No serves as a unique identifier for every student. In an Entity-Relationship (ER) diagram, this key attribute is depicted as an oval shape containing underlying lines.

2 Composite Attribute

A composite attribute is formed by combining multiple individual attributes. An illustration of this is the student Entity type's Address attribute, which encompasses Street, City, State, and Country. Within an ER diagram, a composite attribute is symbolized by an oval containing smaller ovals

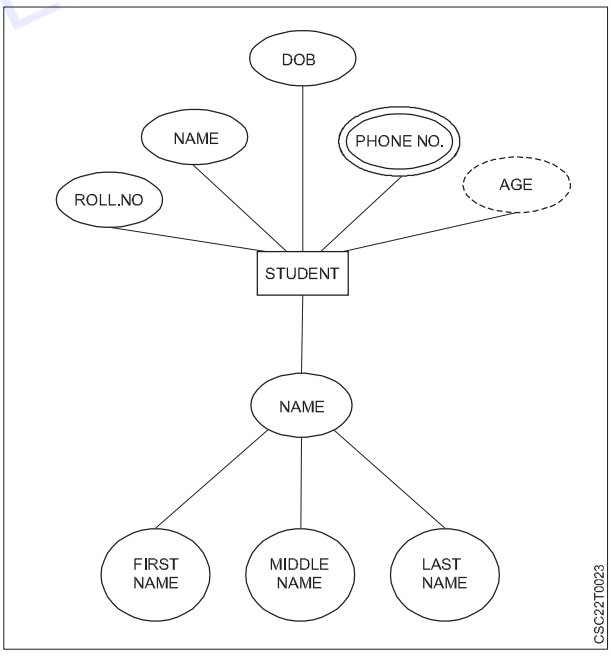
3 Multivalued Attribute

A characteristic that holds multiple values for a specific entity. For instance, Phone_Number (which can have multiple entries for a single student). In an Entity-Relationship (ER) diagram, a multivalued attribute is depicted using a pair of connected ovals.

4 Derived Attribute

An attribute that is obtained based on other attributes within the entity type is referred to as a derived attribute. For example, Age (calculated from Date of Birth). In an ER diagram, a derived attribute is depicted using a dashed oval.

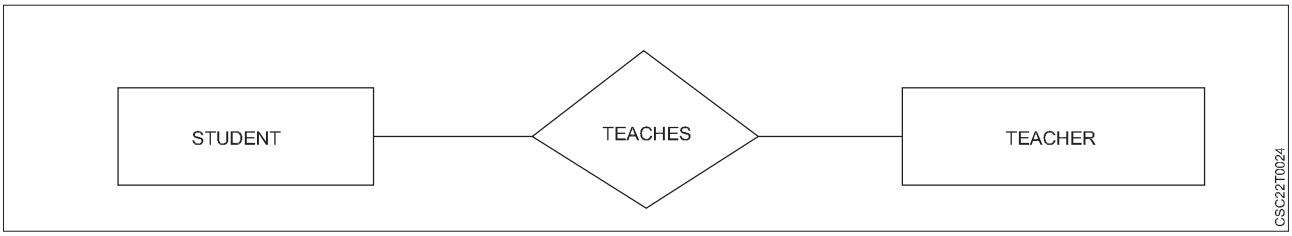
The Student entity type along with its attributes can be illustrated as follows:



Relationship

A relationship is utilized to define the connection between entities. In graphical representations, such as in an ER diagram, this relationship is symbolized using a diamond or rhombus shape.

Types of relationship are as follows:

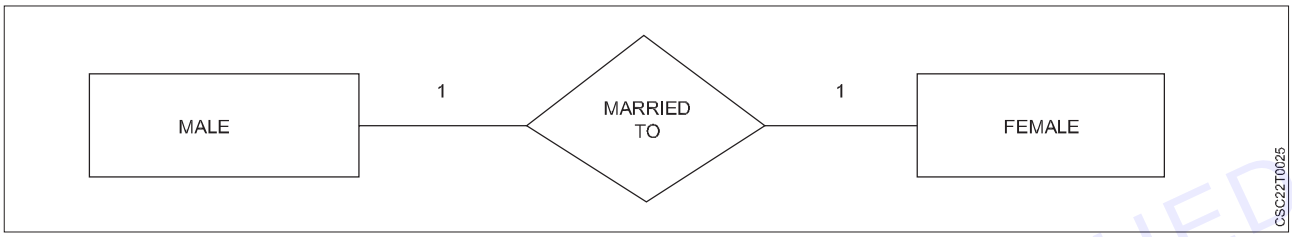


1 **One-to-One Relationship**

If a relationship is such that it links a singular instance of an entity, it is referred to as a one-to-one relationship.

Example :A woman can be married to a single man, and conversely, a man can be married to a single woman.

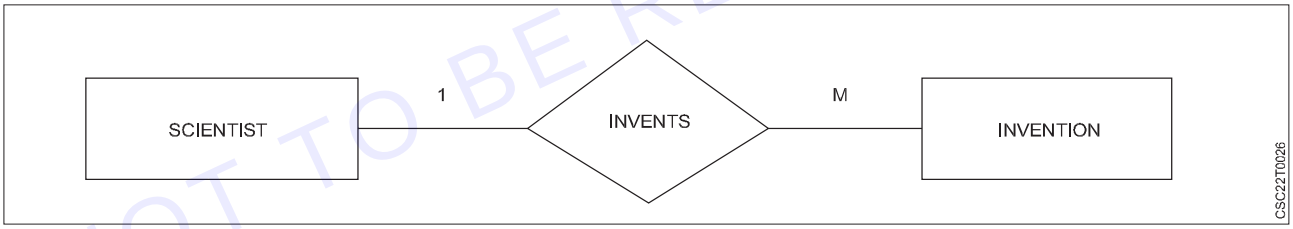
2 **One-to-many relationship**



When only one instance of the entity on the left is connected to multiple instances of the entity on the right within a relationship, it is referred to as a one-to-many relationship.

Example :Scientists have the capability to create numerous inventions, while the act of inventing a particular invention is attributed to a specific scientist.

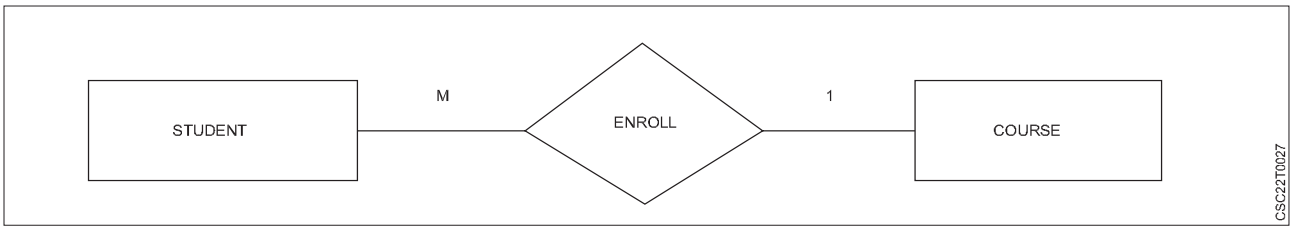
3 **Many-to-one relationship**



When more than one instances of the entity on the left are linked to a single instance of the entity on the right through a relationship, it is referred to as a many-to-one relationship.

Example :A student is enrolled in a single course, whereas a course can accommodate multiple students.

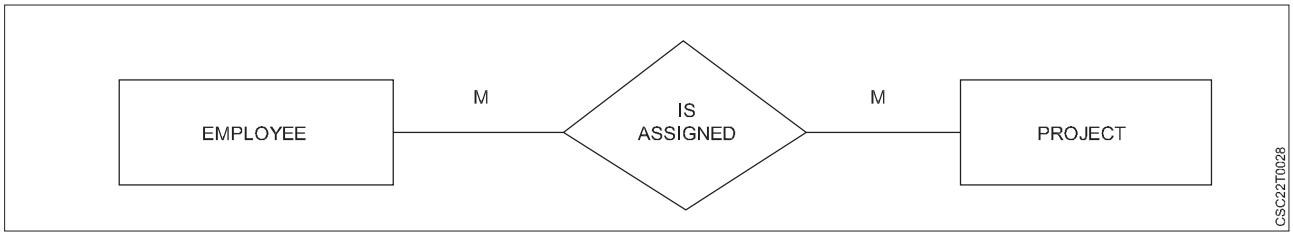
4 **Many-to-many relationship**



A many-to-many relationship arises when multiple instances of the entity on the left are connected with multiple instances of the entity on the right through the relationship.

Example :An employee can be assigned to multiple projects, and likewise, a project can involve multiple employees.

Advantages of ER Model



Simplicity: The conceptual ER Model construction is straightforward. If we grasp the connections between attributes and entities, developing the ER Diagram for the model becomes effortless.

Efficient Communication Tool: Database designers extensively rely on this model to effectively convey their concepts.

Smooth Transition to Various Models: This model harmonizes effectively with the relational model, seamlessly transforming the ER model into tables. Furthermore, it can be adapted into other models like the network model, hierarchical model, and more.

Disadvantages of ER Model

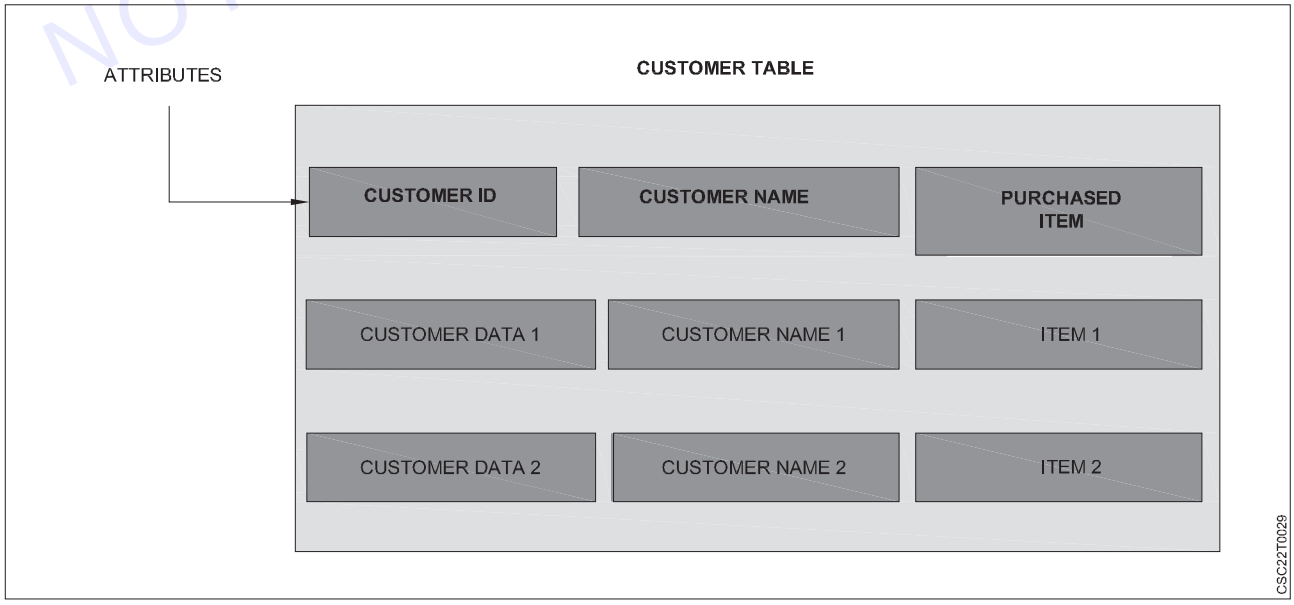
Lack of Notation Consistency: An established industry standard for creating an ER model is absent. Consequently, different developers might employ notations that are unclear to their peers.

Concealed Information: The ER model's high-level perspective can lead to the omission or concealment of certain details, potentially resulting in the loss of some information.

Schema

What is Schema?

- The framework of the database takes form through the attributes, and this framework is identified as the schema.
- A schema outlines the logical limitations, such as tables and primary keys, that define the structure of a database.
- The schema doesn't encompass the data attribute types



Database Schema

A database schema is a logical and structured representation of the organization, arrangement, and relationships among the data stored in a database. It defines the design, format, and constraints of the data stored in the database tables, along with the interconnections between these tables. In essence, a database schema outlines the blueprint for how data is organized, stored, and accessed within a database management system. It includes information about tables, fields, data types, relationships, constraints, and other elements that define the structure and integrity of the database.

Types of Database Schema

The database schema is categorized into three types, namely

1 Logical Schema

2 Physical Schema

3 View Schema

1 **Physical Database Schema**

- The physical schema outlines how data is physically stored within storage systems as files and indices. It involves the concrete code or syntax required to establish the database's structure. When crafting a database structure on the physical level, it is referred to as the physical schema.
- The choice of data storage locations and methods within various storage blocks is made by the Database Administrator.

2 **Logical Database Schema**

- The logical database schema encompasses all the rational restrictions to be enforced on the stored data, as well as outlines the tables, perspectives, entity connections, and integrity constraints.
- The logical schema elucidates the manner in which data is stored, comprising tables and their interconnected attributes.
- Through the utilization of ER modeling, the connections among data elements are upheld.
- Within the logical schema, diverse integrity constraints are outlined to ensure the accuracy of data insertion and updates.

3 **View Schema**

- This refers to a view-level design that outlines how interactions between end-users and the database are defined.
- Users can interact with the database through an interface without requiring extensive knowledge about the underlying data storage methods employed within the database.

Three Layer Schema Design

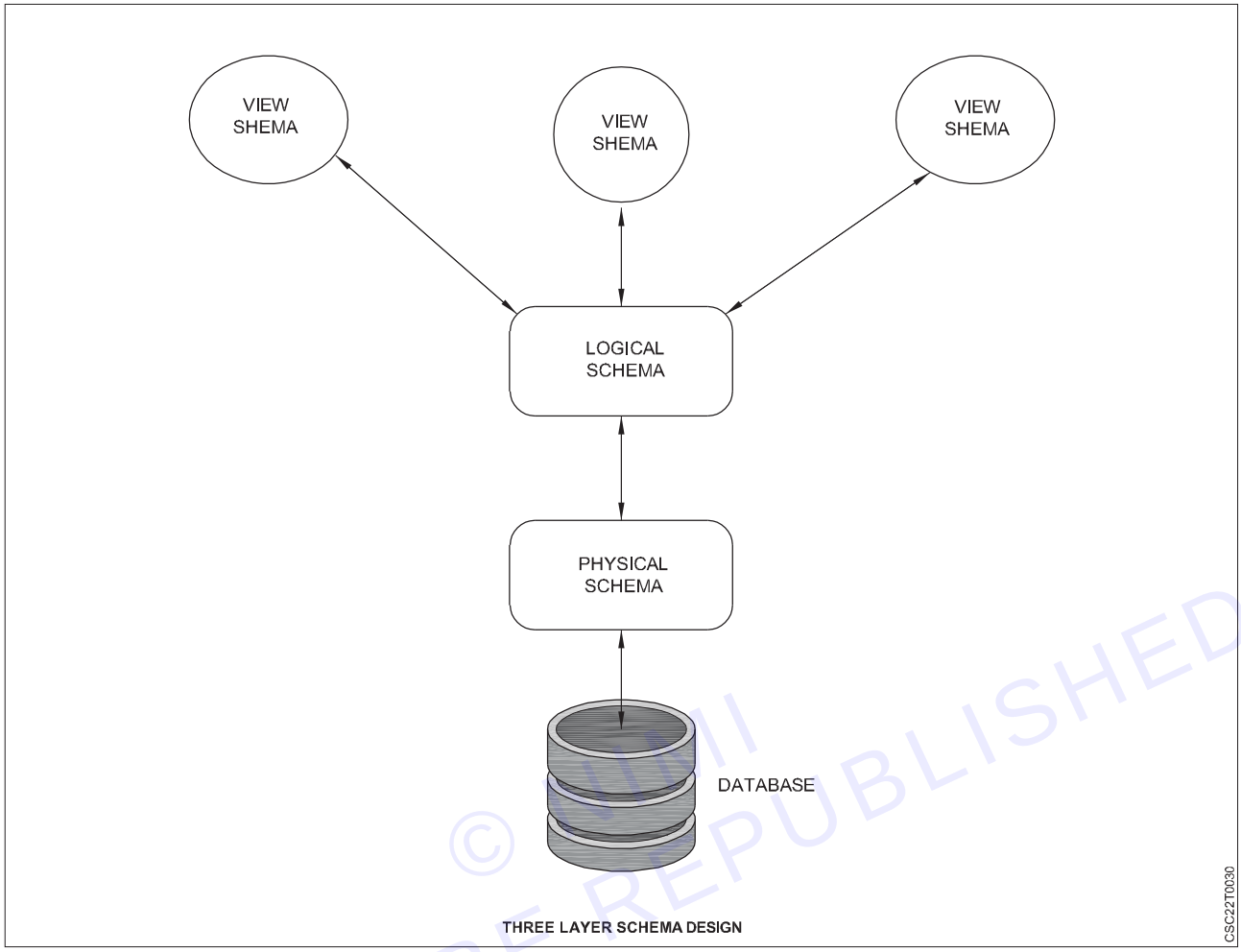
Creating Database Schema

To create a schema, the "CREATE SCHEMA" statement is employed in various databases. However, the interpretation of this statement can differ across different database systems. Let's explore some examples of statements used for creating a database schema in various database systems:

1 **MySQL**: In MySQL, the "CREATE SCHEMA" statement is utilized to create a database. This is because, in MySQL, both "CREATE SCHEMA" and "CREATE DATABASE" statements serve the same purpose.

2 **SQL Server**: Within SQL Server, the "CREATE SCHEMA" statement is employed to generate a new schema.

3 **Oracle Database**: In Oracle Database, the "CREATE USER" statement is used to create a new schema. In Oracle, a schema is automatically generated with every database user. The "CREATE SCHEMA" statement, however, doesn't create a new schema. Instead, it populates the existing schema with tables and views, facilitating access to these objects without necessitating multiple SQL statements for separate transactions.



THREE LAYER SCHEMA DESIGN

Database Schema Designs

Creating a schema design constitutes the initial phase in establishing a solid groundwork for data administration. Inefficient schema designs prove challenging to oversee and result in heightened memory consumption and resource utilization. The logical course of action is contingent upon the demands of the business. Opting for the appropriate database schema design is essential to simplify the project lifecycle. Presented herewith is an assortment of well-known database schema designs.

1 Flat Model

2 Hierarchical Model

3 Network Model

4 Relational Model

5 Star Schema

6 Snowflake Schema

1 **Flat Model**

A flat model schema embodies a 2-dimensional array where each column holds identical data types, and items within a row possess interrelatedness. It can be likened to a solitary spreadsheet or a database table devoid of interconnections. This schema structure proves optimal for modest applications devoid of intricate data structures.

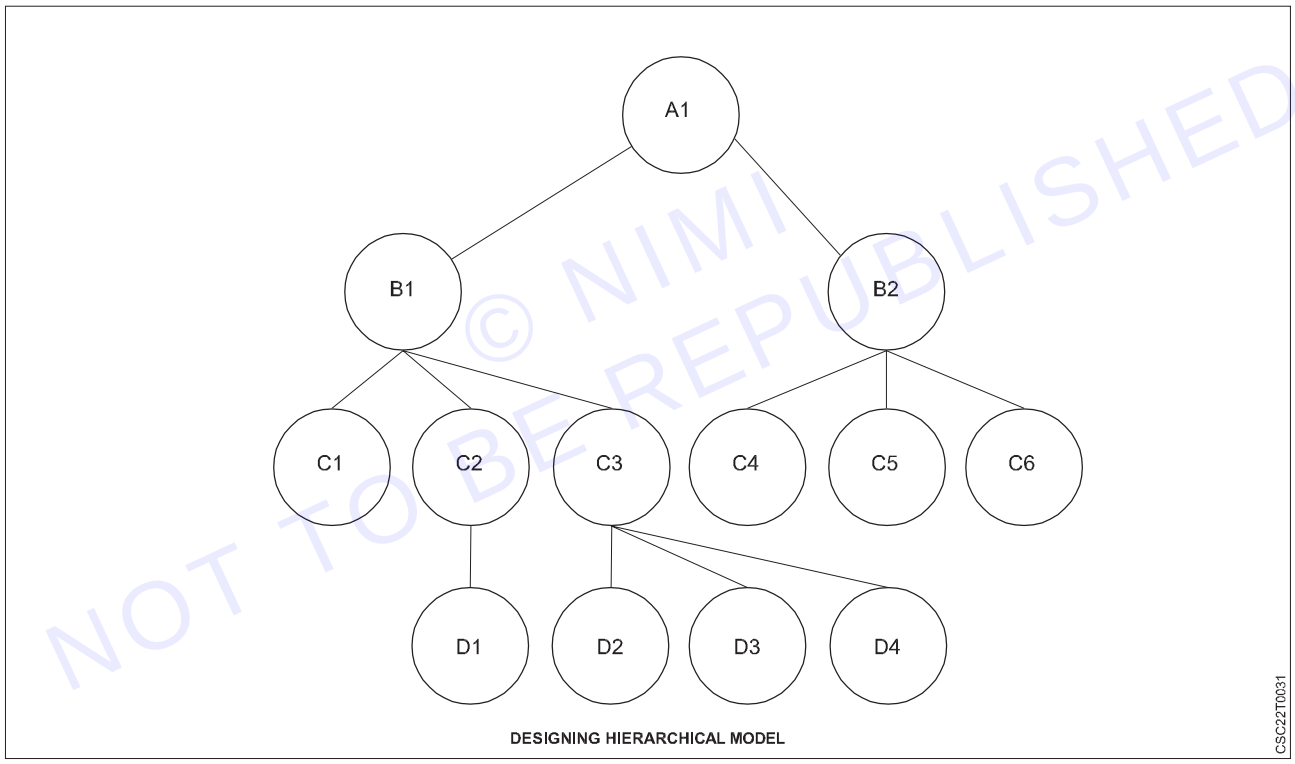
First_name	last_name	email	phone

Designing Flat model

2 **Hierarchical Model**

The Hierarchical model design features a configuration resembling a tree. Within this tree structure, there exists a root data node and subsequent child nodes. A one-to-many relationship is established between each parent node and its child node. These types of database schemas find representation in formats like XML or JSON files, which are capable of encompassing entities along with their respective sub-entities.

Hierarchical schema models excel in housing nested data, effectively representing instances like Hominoid classification.



DESIGNING HIERARCHICAL MODEL

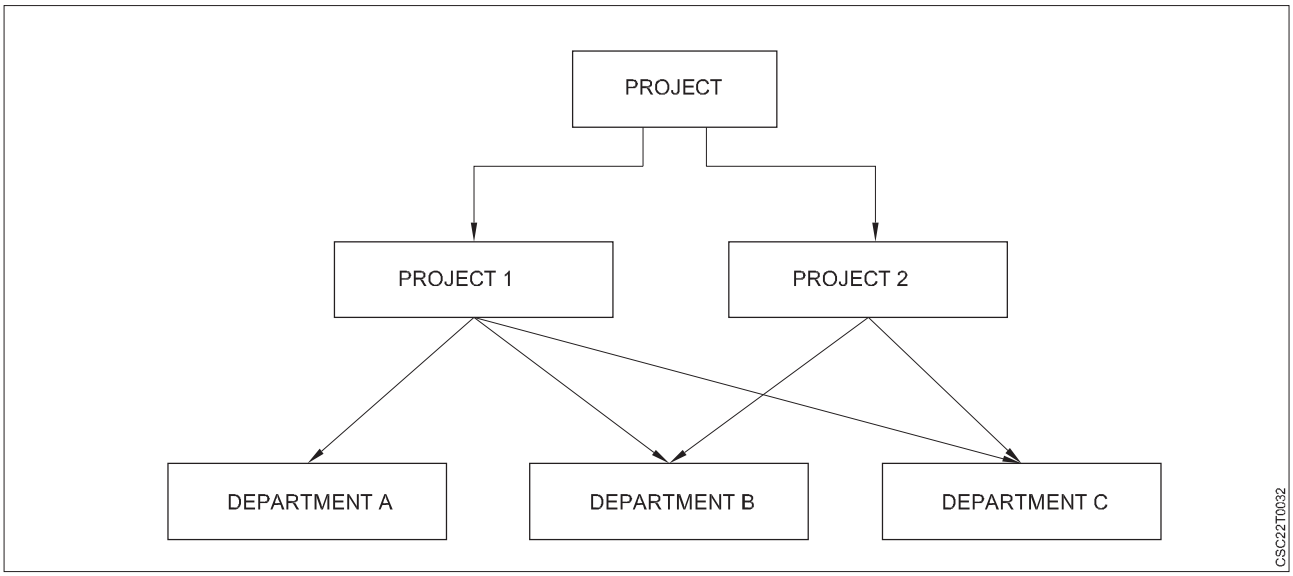
Designing Hierarchical Model

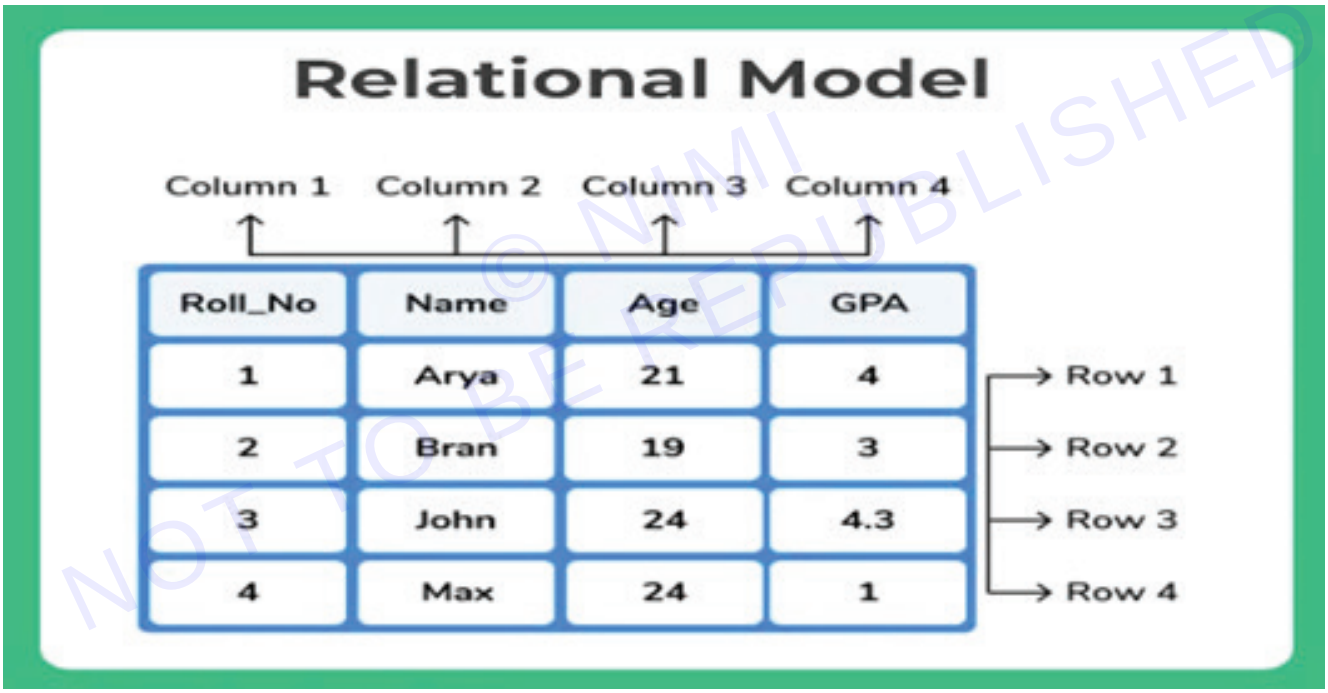
3 **Network Model**

The network model and the hierarchical model bear striking resemblances, but a significant disparity lies in their handling of data relationships. The network model permits the existence of many-to-many relationships, in contrast to the hierarchical models, which solely accommodate one-to-many relationships.

4 **Relational Model**

Relational models serve as the foundation for relational databases, where data is stored in the form of tables or relations. Within this context, relational operators are employed to manipulate and compute various values from the data.





5 **Star Schema**

The star schema represents an alternative schema design approach for data organization. It excels in managing and analyzing vast volumes of data and operates based on the concepts of “Facts” and “Dimensions”. In this context, a fact corresponds to a numerical data point that drives business processes, while a dimension provides context and description to the facts. The Star Schema is particularly effective for structuring data within Relational Database Management Systems (RDBMS).

6 **Snowflake Schema**

The snowflake schema is a modification of the star schema. In the star schema, there exists a central “Fact” table housing primary data points with references to its associated dimension tables. However, in the snowflake schema, dimension tables have the potential to branch out into their own additional dimension tables, creating a more intricate hierarchical structure.

Designing Database using Normalization Rules

Normalization

Normalization is a database design process that involves organizing and structuring a relational database to reduce data redundancy and dependency, thereby improving data integrity and overall efficiency. The primary goal of normalization is to eliminate anomalies that can occur when data is stored in a non-optimal structure. It helps ensure that the data is stored in a way that minimizes redundancy while preserving the relationships between different pieces of information.

The process of normalization is typically carried out through a series of steps or "normal forms," each addressing specific types of data anomalies. The most commonly discussed normal forms are the first normal form (1NF), second normal form (2NF), third normal form (3NF), and beyond.

Here's a brief overview of these normal forms

1 **First Normal Form (1NF)**

In 1NF, each column in a table must hold only atomic (indivisible) values, and each row should be uniquely identifiable. This eliminates the possibility of storing multiple values within a single cell and ensures that the data is organized in a tabular format.

EXAMPLE :Table 1's "STUDENT" relation violates the first normal form (1NF) due to the presence of the multi-valued attribute "STUD_PHONE". The decomposition of this relation into 1NF is demonstrated in table 2

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1

Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2

Example 2

ID Name Courses

---

1 A m1, m2

2 B m3

3 c m2, m3

- The previous table exhibits a multi-valued attribute "Course," which renders it non-compliant with the first normal form (1NF). The subsequent table, on the other hand, adheres to 1NF principles, as it lacks any multi-valued attributes.

ID	Name	Course
1	A	m1
1	A	m2
2	B	m3
3	C	m2
3	C	m3

2 **Second Normal Form (2NF)**

In 2NF, the table must be in 1NF, and all non-key attributes (attributes that are not part of the primary key) must be fully functionally dependent on the entire primary key. This eliminates partial dependencies, where non-key attributes are dependent on only a portion of the primary key.

Example

Product ID	product	Brand
1	Monitor	Dell
2	Monitor	Apple
3	Scanner	HP
4	Head phone	JBL

Product table following 2NF

Products Category table:

Product ID	product
1	Monitor
2	Monitor
3	Scanner
4	Head phone

Brand table:

Product ID	Brand
1	Dell
2	Apple
3	HP
4	JBL

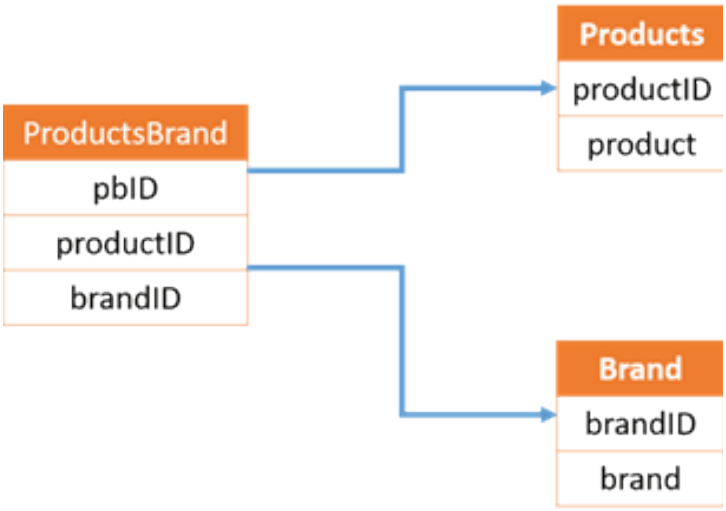
Products Brand table:

pb ID	product ID	brand ID
1	1	1
2	1	2
3	2	3
4	3	4

3 **Third Normal Form (3NF)**

In 3NF, the table must be in 2NF, and all non-key attributes must be directly dependent on the primary key. This eliminates transitive dependencies, where non-key attributes are dependent on other non-key attributes.

These three normal forms are the most commonly used and usually suffice for many database designs. However, there are additional normal forms like Boyce-Codd Normal Form (BCNF) and Fourth Normal Form (4NF) that address more complex scenarios of data dependencies and anomalies.



Normalization helps improve database design by:

- Reducing data redundancy, which in turn conserves storage space and ensures consistency.

- Preventing update anomalies (problems that arise when changes to data aren't properly reflected).
- Enhancing data integrity by maintaining consistent relationships between entities.

Benefits of Normalization

1 Reduces data duplication.

2 Enhances database organization.

3 Ensures uniform data across the database.

4 Enables increased flexibility in database design.

5 Upholds relational integrity principles.

Disadvantages of Normalization

1 Requires clear user requirements before database construction.

2 Performance deterioration occurs with higher-level normalization, like 4NF and 5NF.

3 Normalizing relations of higher degrees is time-consuming and complex.

4 Inadequate decomposition can lead to suboptimal database design and subsequent issues.

Various data types Data integrity, DDL DML and DCL statements

Data integrity refers to the accuracy, consistency, and reliability of data in any given context. It is a fundamental aspect of data management and is essential for ensuring that data remains intact and trustworthy throughout its lifecycle. Data integrity is particularly important in various domains, including databases, information systems, and data storage, as well as in compliance with regulations and data security.

1 **Accuracy**: Data should be free from errors and represent the true and correct values or information. Inaccurate data can lead to flawed decision-making and operational problems.

2 **Consistency**: Data should remain uniform and coherent across various data sources, databases, or data sets. Inconsistent data can lead to confusion and hinder data analysis.

3 **Completeness**: Data must be complete, meaning that all expected data elements or fields are present and appropriately filled in. Incomplete data can result in incomplete analysis or reports.

4 **Reliability**: Data should be reliable and available when needed. Unreliable data or data that is frequently unavailable can disrupt business processes.

5 **Security**: Data integrity also involves ensuring that data is protected from unauthorized access, tampering, or corruption. Security measures like encryption and access controls are essential to maintaining data integrity.

Methods and practices for ensuring data integrity include:

1 **Validation Rule**: Implement validation rules and constraints to ensure that data entered into a system adheres to predefined criteria, such as data types, ranges, and formats.

2 **Data Validation**: Employ data validation techniques to check data for accuracy and consistency. This may involve data cleansing, deduplication, and normalization.

3 **Backup and Recovery**: Regularly back up data and have robust disaster recovery plans in place to ensure data can be restored in case of corruption or loss.

4 **Access Control**: Implement access controls and authentication mechanisms to prevent unauthorized access and modification of data.

5 **Audit Trails**: Maintain audit trails to record and track changes made to data, including who made the changes and when they occurred. This helps identify and rectify unauthorized alterations.

6 **Hash Functions**: Use cryptographic hash functions to create checksums or hashes of data. Comparing these hashes can reveal any unauthorized changes to the data.

7 **Data Verification**: Periodically verify the accuracy and consistency of data through reconciliation and verification processes.

8 **Data Governance**: Establish data governance policies and procedures to enforce data integrity standards across the organization.

Types of data integrity

There are mainly four types of Data Integrity:

1 Domain Integrity

2 Entity Integrity

3 Referential Integrity

4 User-Defined Integrity

Domain Integrity

Domain, in this context, pertains to the acceptable values within a specified range. It denotes the scope of values that can be utilized and stored in a specific database column. The available data types primarily include integers, text, and dates, among others. It's important that any input entered into a column falls within the permissible range of the associated data type.

Example-To store employee salaries in the 'employee_table,' it's possible to implement constraints that permit only integer values. Any input that deviates from this requirement, like text or character-based data, would be declined, and the Database Management System (DBMS) would generate error messages to indicate the violation of the defined constraint. This helps ensure that only valid integer values are accepted for salary entries in the database, maintaining data consistency and accuracy.

Employee_id	Name	Salary	Age
1	Andrew	486522	25
2	Angel	978978	30
3	Anamika	697abc	35

This value is out of domain(not INTEGER)so it is not acceptable.

DOMAIN INTEGRITY

Entity Integrity

Every row representing an entity in a database table must have a distinct means of identification.

This is typically achieved using primary keys, which serve as unique identifiers for each record.

It's essential to enforce the entity constraint, which specifies that the primary key value must not be NULL.

This requirement ensures that every record in the database has a specific and non-null primary key value.

When the primary key value is not NULL, it becomes possible to distinguish records from one another, even if all other field values are identical. In essence, primary keys enable the unequivocal identification of each individual record in the database.

Example: In a customer database with a 'customer_table' containing attributes like age and name, it's crucial to ensure that each customer can be uniquely identified. Sometimes, there might be two customers with identical names and ages, leading to confusion when retrieving data. To address this challenge, primary keys are assigned

to each table entry. These primary keys serve the purpose of uniquely identifying each record in the table, even in cases where other attributes like name and age might not be sufficient for differentiation.

Primary Key

ID	Customer_Name	Age
1	Andrew	18
2	Angel	20
	Angel	20

This value cannot be NULL as we will not be able to identify customers uniquely

ENTITY INTEGRITY

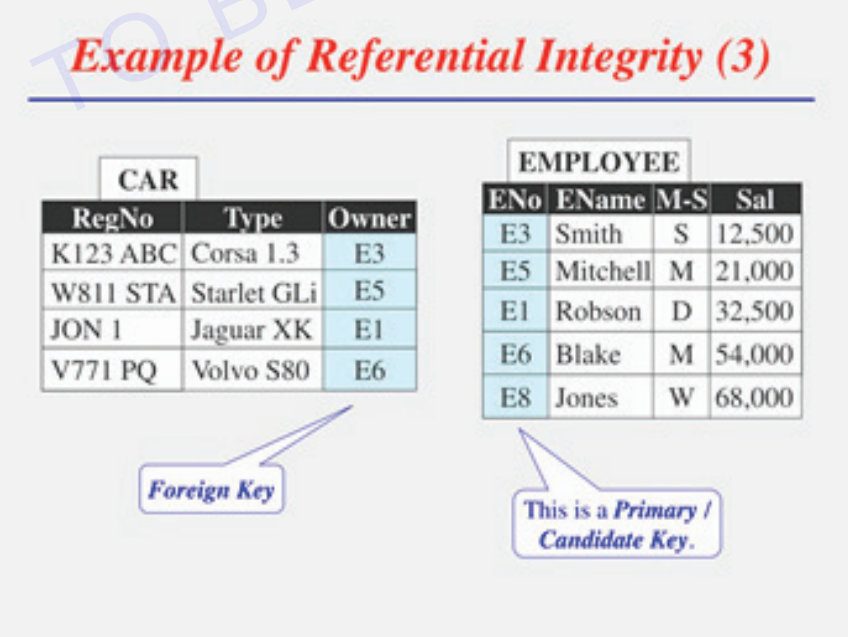
Referential Integrity

Referential Integrity is a crucial concept employed to uphold data consistency when managing two interconnected tables within a database. It involves establishing specific rules within the database structure to ensure that modifications, insertions, and deletions in the database do not compromise data integrity. These constraints for referential integrity dictate that when a foreign key in one table references the primary key of another table, every value of that foreign key in the first table must either be null or correspond to a valid entry in the second table.

Example: Imagine we have two tables: “table 1” (with columns student_id, name, age, and course_id) and “table 2” (with columns course_id, course_name, and duration).

In the context of referential integrity, it means that if any “course_id” exists in the “table 1” table, it must also exist in the “table 2” table; otherwise, this scenario is not permitted.

In other words, the “course_id” in the “table 1” table should either be null or, if a “course_id” is present, it must be a valid entry in the “table 2” table. This way, referential integrity is maintained to ensure the consistency and accuracy of data between these two tables.



User-Defined Integrity

On occasion, domain, referential, and entity integrity alone may fall short in preserving data integrity. In such cases, additional measures are often employed, typically involving the use of triggers and stored procedures.

Triggers are essentially sets of statements that automatically execute in response to predefined events, providing a means to enforce more intricate data integrity rules when necessary.

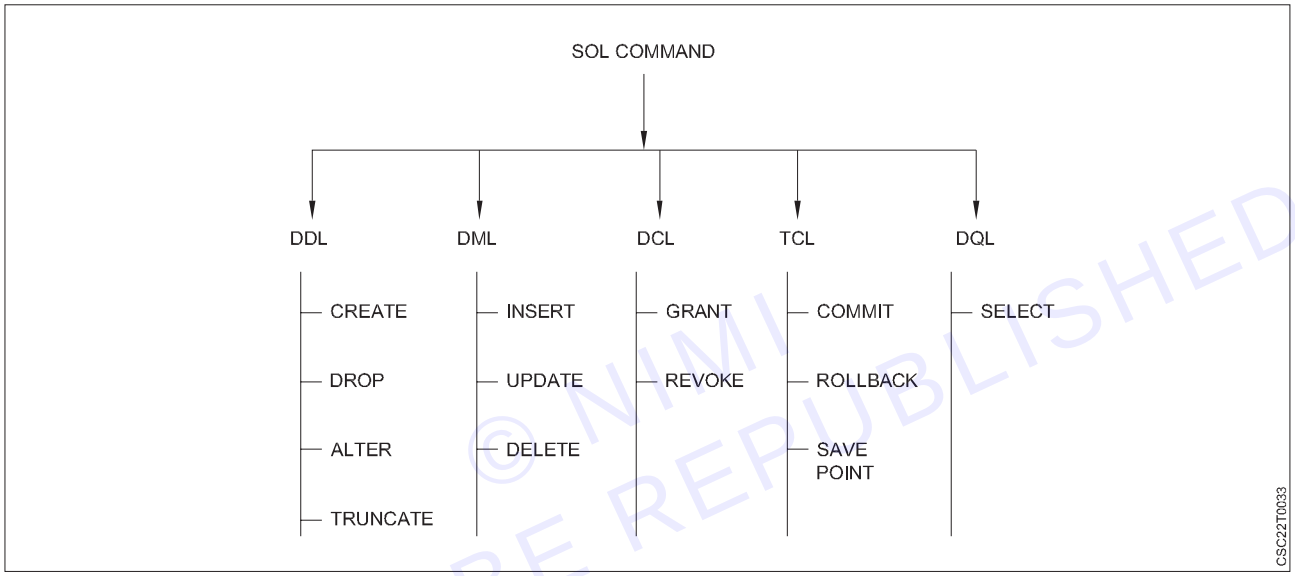
Example:-When a new row containing marks for various subjects of students is added to the student_table, an automatic calculation of the new average is performed and stored.

SQL commands

- SQL commands are directives employed to interact with a database, facilitating the execution of particular actions, operations, and inquiries on data within the database.
- SQL has the capability to execute a range of functions, such as generating tables, inserting data into tables, deleting tables, altering table structures, and defining user permissions.

Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



DDL (Data Definition Language)

DDL, which stands for Data Definition Language, is a subset of SQL (Structured Query Language) used for defining and managing the structure of a database. DDL commands allow you to create, modify, and delete database objects like tables, indexes, and constraints. DDL is primarily concerned with defining the schema or structure of the database. Common DDL commands include:

1 **CREATE**: Used to create new database objects such as tables, indexes, and views.

CREATE TABLE table_name (

column1 datatype,

2 **ALTER**: Used to modify the structure of an existing database object, like adding or dropping columns from a table.

ALTER TABLE table_name

ADD column_name datatype;

ALTER TABLE table_name

DROP COLUMN column_name;

3 **DROP**: Used to delete a database object like a table, index, or view.

DROP TABLE table_name;

4 **TRUNCATE**: Used to remove all rows from a table but keep the table structure intact.

TRUNCATE TABLE table_name;

5 **COMMENT**: Used to add comments or descriptions to database objects for documentation purposes.

ON TABLE table_name IS 'This is a table comment.';

6 **CREATE INDEX**: Used to create an index on one or more columns of a table for faster data retrieval.

CREATE INDEX index_name ON table_name (column1, column2);

7 **CREATE VIEW**: Used to create a virtual table based on the result of a query.

CREATE VIEW view_name AS

SELECT column1, column2 FROM table_name WHERE condition;

DDL commands are typically used by database administrators and developers to design and manage the database's structure. They are essential for defining how data should be organized and ensuring data integrity within a database system.

DML(Data Manipulation Language)

DML, which stands for Data Manipulation Language, is a subset of SQL (Structured Query Language) that is used for managing and manipulating data stored in a database. DML commands allow you to perform operations on the data itself, such as inserting, updating, and deleting records in database tables. The primary DML commands are:

1 **INSERT**: Used to add new rows (records) of data into a database table.

INSERT INTO table_name (column1, column2, ...)

VALUES (value1, value2, ...);

2 **UPDATE**: Used to modify existing data in a database table.

UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

3 **DELETE**: Used to remove rows (records) from a database table based on specified criteria.

DELETE FROM table_name

WHERE condition;

DML commands are essential for maintaining and changing the data within a database. These commands enable users and applications to insert new data, update existing data, and remove unwanted data, ensuring that the database remains accurate and up-to-date.

DCL (Data Control Language)

DCL, which stands for Data Control Language, is a subset of SQL (Structured Query Language) used for controlling and managing permissions and access rights within a database management system (DBMS). DCL commands are essential for ensuring data security and access control by specifying which users or roles have the authority to perform certain actions on database objects. The two primary DCL commands are:

1 **GRANT**: The GRANT command is used to give specific privileges or permissions to users or roles. These privileges can include the ability to perform actions like SELECT, INSERT, UPDATE, DELETE, or even the ability to create or modify database objects.

GRANT privilege_type

ON object_name

TO user_or_role;

For example, to grant SELECT permission on a table to a user:

GRANT SELECT ON table_name TO user_name;

2 **REVOKE**: The REVOKE command is used to remove previously granted privileges from users or roles.

REVOKE privilege_type ON object_name FROM user_or_role;

For example, to revoke SELECT permission on a table from a user:

REVOKE SELECT ON table_name FROM user_name;

DCL commands play a crucial role in controlling who can access and manipulate data within a database, ensuring data integrity and security. Database administrators use these commands to define and enforce access policies, restrict unauthorized access, and manage the permissions of users and roles in the database system. Properly configured DCL commands help protect sensitive data and maintain the integrity of the database.

TCL (Transaction Control Language)

TCL, which stands for Transaction Control Language, is a subset of SQL (Structured Query Language) used for managing database transactions. Transactions in a database are sequences of one or more SQL statements that are treated as a single, indivisible unit of work. TCL commands are used to control the beginning and ending of transactions, ensuring data consistency and integrity. The primary TCL commands are:

1 **COMMIT**: The COMMIT command is used to permanently save the changes made during the current transaction. Once a COMMIT is issued, all changes are made permanent and cannot be rolled back.

COMMIT;

2 **ROLLBACK**: The ROLLBACK command is used to undo changes made during the current transaction and restore the database to its previous state. It cancels all the changes made since the last COMMIT or SAVEPOINT.

ROLLBACK;

3 **SAVEPOINT**: The SAVEPOINT command is used to set a point within a transaction to which you can later roll back if needed. It allows you to create intermediate savepoints within a transaction.

SAVEPOINT savepoint_name;

TCL commands are critical for maintaining data consistency and ensuring that a series of related SQL statements are executed as a single, atomic operation. Transactions help protect data integrity and ensure that the database remains in a consistent state even in the presence of errors or interruptions. The use of COMMIT and ROLLBACK commands is essential for managing the success or failure of database operations within a transaction.

DQL(Data Query Language)

Data Query Language (DQL) is a subset of SQL (Structured Query Language) specifically designed for retrieving and querying data from a relational database. DQL commands are used to interact with the data stored within database tables, allowing users to retrieve, filter, and manipulate data to extract meaningful information. The primary DQL command is:

SELECT: The SELECT command is the core of DQL and is used to retrieve data from one or more tables in a database. It enables you to specify the columns you want to retrieve, apply filtering conditions, and sort the result set.

SELECT column1, column2

FROM table_name

WHERE condition ORDER BY column1;

In addition to SELECT, DQL may involve using clauses like WHERE to filter data, JOIN to combine data from multiple tables, GROUP BY for grouping data, HAVING for filtering grouped data, and more.

DQL is fundamental for extracting and presenting data in a structured and meaningful way, making it one of the most commonly used components of SQL, especially for reporting and analysis purposes. It allows users to interact with and retrieve data from a database, helping them make informed decisions based on the information stored in the database.

Enforcing Primary key and foreign key

What is the Key?

In the realm of database management systems, a key refers to a particular attribute or a group of attributes employed to distinguish each record, often called a tuple, within a table uniquely. Keys hold significant importance in the structure of a relational database, as they serve to organize data and create connections between tables. Within a database, various types of keys exist, each serving a distinct function. Keys are pivotal components in relational databases, as they ensure record uniqueness, facilitate table relationships, and enhance overall database performance.

1 **Enforcing a Primary Key**

A primary key is a unique identifier for each record in a table. To enforce a primary key constraint, follow these steps:

- Define a primary key when creating a table using the 'PRIMARY KEY' constraint.

  CREATE TABLE employees (employee_id INT PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50));

- If you're altering an existing table to add a primary key, use the ALTER TABLE statement.

  ALTER TABLE employees ADD PRIMARY KEY (employee_id);

  The primary key constraint enforces uniqueness, meaning that no two rows in the table can have the same value for the primary key column(s).

2 **Enforcing a Foreign Key**

A foreign key is a field in one table that refers to the primary key in another table, establishing a relationship between the tables. To enforce a foreign key constraint, follow these steps:

- Define a foreign key when creating a table using the 'FOREIGN KEY' constraint.

CREATE TABLE orders (

order_id INT PRIMARY KEY,

customer_id INT, order_date DATE,

FOREIGN KEY (customer_id) REFERENCES customers(customer_id)

);

- The 'REFERENCES' clause specifies the table and column that the foreign key references. In this example, it references the 'customer_id' column in the 'customers' table.
- Ensure that the referenced column in the referenced table (in this case, 'customers' in the 'customers' table) has a primary key constraint.

Foreign key constraints enforce referential integrity, ensuring that data in the referencing table (in this case, the 'orders' table) is consistent with data in the referenced table (in this case, the 'customers' table).

Enforcing these constraints helps maintain data quality and consistency in your database, preventing the insertion of invalid or inconsistent data and facilitating data relationships between tables.

Difference between Primary Key and Foreign Key

Primary keys and foreign keys are both fundamental components of relational database design, but they serve different purposes and have distinct characteristics. Here are the key differences between primary keys and foreign keys:

Primary Key

1 **Uniqueness**: A primary key is a column or a set of columns in a table that uniquely identifies each row. Every value in the primary key column(s) must be unique within the table.

2 **Required**: A primary key is required for every table. It ensures that each row in the table can be uniquely identified.

3 **Constraints**: Primary keys enforce data integrity by preventing duplicate and null values in the primary key column(s).

4 **Indexed**: By default, primary key columns are automatically indexed, which can improve query performance.

5 **One per Table**: Each table can have only one primary key.

6 **Used for Joins**: Primary keys are often used as reference points for creating relationships with other tables through foreign keys.

Foreign Key

1 **Relationships**: A foreign key is a column or a set of columns in one table that establishes a link or relationship between data in two tables. It references the primary key of another table.

2 **Referential Integrity**: Foreign keys enforce referential integrity by ensuring that data in the referencing table (child table) corresponds to data in the referenced table (parent table).

3 **Optional**: While foreign keys are commonly used to create relationships, they are not required in every table. Tables can exist without foreign keys.

4 **Values Must Match**: Values in the foreign key column(s) must match values in the primary key column(s) of the referenced table.

5 **Can Have Multiple**: A table can have multiple foreign keys, each referencing a different table and primary key.

6 **Used for Joins**: Foreign keys are used to join related tables, allowing you to retrieve data from multiple tables based on their relationships.

Adding Indices

Adding indices to databases is a fundamental part of database optimization. Indices improve query performance by allowing the database management system (DBMS) to quickly locate rows that meet specific criteria. Here's how you can add indices to a database, typically using SQL:

1 **Determine Which Columns to Index**

Start by identifying the columns that are frequently used in WHERE clauses of your SELECT queries. These are good candidates for indexing, as they can significantly speed up data retrieval. Consider indexing columns used in JOIN conditions as well.

2 **Choose the Appropriate Index Type**

There are different types of indexes, and the choice depends on your specific use case and DBMS:

- **Single-Column Index**: This indexes a single column.
- **Composite Index**: Indexes multiple columns together, useful for queries involving multiple criteria.
- **Unique Index**: Ensures uniqueness on indexed columns.
- **Clustered Index (in some DBMS)**: Determines the physical order of data rows in the table.
- **Non-Clustered Index**: Contains a copy of the indexed columns and a pointer to the actual data row.

3 **Create an Index**

To create an index, you use the 'CREATE INDEX' statement in SQL. Here's a basic syntax example for creating a single-column index:

CREATE INDEX index_name

ON table_name (column_name);

For example, to create an index called 'idx_last_name' on the 'last_name'column of a table named 'employees' :-

CREATE INDEX idx_last_name

ON employees (last_name);

If you want to create a composite index on multiple columns, you can specify them within the parentheses:

CREATE INDEX idx_name_age

ON employees (last_name, age);

4 **Monitor and Maintain Indexes**

After adding indexes, it's essential to monitor their performance over time. Regularly analyze query execution plans to ensure that the indexes are being used as expected. Additionally, you may need to perform index maintenance tasks like rebuilding or reorganizing indexes to optimize performance as data changes.

5 **Consider Index Size**

Keep in mind that indexes consume storage space. Too many indexes or indexing large columns can significantly increase storage requirements. Balance the benefits of improved query performance against the storage costs.

6 **Drop or Modify Indexes**

If you find that an index is not being used or is no longer necessary, you can drop it using the 'DROP INDEX' statement:

DROP INDEX index_name;

You can also modify existing indexes, such as adding or removing columns from composite indexes, depending on your DBMS.

7 **Test Query Performance**

Before and after adding indexes, test the performance of your queries to ensure that they have indeed improved. Profiling tools and query execution plans can help you evaluate the effectiveness of your indexing strategy.

Concepts of Transactions

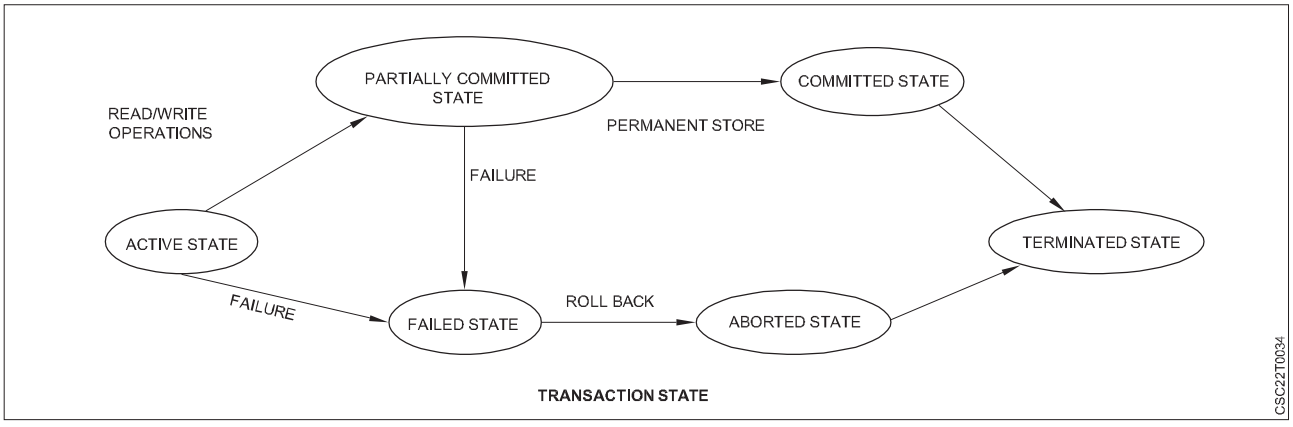
A transaction is a fundamental concept in DBMS that represents a unit of work involving one or more database operations (e.g., INSERT, UPDATE, DELETE, SELECT). Here are some key concepts associated with transactions:

Properties of Transaction (ACID Properties)

To carry out a transaction within a Database Management System (DBMS), it must exhibit a set of characteristics commonly referred to as the ACID properties.

- A – Atomicity
- C – Consistency
- I – Isolation
- D – Durability

Transaction States



Transactions in database management systems (DBMS) have several key properties that define their behavior and ensure data integrity. These properties are often referred to as the ACID properties:

1 **Atomicity (A)**: Atomicity ensures that a transaction is treated as a single, indivisible unit of work.

It follows the "all or nothing" principle, meaning that all the operations within a transaction are either fully completed or fully undone in case of an error or failure.

If any part of the transaction fails, the entire transaction is rolled back, and the database remains unchanged.

Atomicity ensures that the database remains in a consistent state.

2 **Consistency (C)**: Consistency guarantees that a transaction brings the database from one valid state to another valid state.

This means that a transaction must adhere to all integrity constraints, rules, and validations defined in the database schema.

If a transaction violates consistency rules, it is rolled back, and the database remains unchanged. Consistency ensures that data remains in a coherent state during and after the transaction.

3 **Isolation (I)**: Isolation defines the degree to which one transaction is isolated from the effects of other concurrent transactions.

It ensures that concurrent transactions do not interfere with each other and maintains data integrity.

Different isolation levels (e.g., Read Uncommitted, Read Committed, Repeatable Read, Serializable) specify the level of isolation between concurrent transactions.

Higher isolation levels provide stronger guarantees but may impact performance due to locking and resource contention.

4 **Durability (D)**: Durability guarantees that once a transaction is committed, its effects are permanent and will survive any subsequent system failures, such as power outages, crashes, or hardware failures.

This is typically achieved by writing transaction changes to non-volatile storage (e.g., disk) and maintaining a transaction log for recovery purposes. Durability ensures data persistence and reliability.

These ACID properties collectively provide a framework for ensuring the reliability, consistency, and integrity of database transactions, even in complex and concurrent environments. They are essential in applications where data accuracy is critical, such as financial systems, e-commerce platforms, and any scenario where data integrity is paramount. Transactions that adhere to the ACID properties can be trusted to maintain data integrity and consistency.

States of Transaction

Transactions in a database management system (DBMS) go through various states during their lifecycle. These states represent the progress and outcome of a transaction. The common states of a transaction include:

1 **Active (or Running)**: The transaction is in the active state when it is executing its operations. During this phase, the transaction is interacting with the data base, reading, and writing data. It may issue one or more SQL statements.

2 **Partially Committed (or Preparing)**: After the transaction has executed its operations successfully, it enters the partially committed state. In this state, the DBMS prepares to make the changes permanent but has not yet committed the transaction. The system ensures that all necessary conditions are met before moving to the committed state.

3 **Committed**: In the committed state, the transaction has been successfully completed, and all its changes have been made permanent in the database. Once a transaction is committed, its changes are considered permanent and cannot be undone.

4 **Aborted (or Rolled Back)**: If an error occurs during the execution of the transaction or if the transaction encounters a condition that causes it to fail, it may enter the aborted state. In this state, the transaction is rolled back, which means that any changes it made to the database are undone, and the database is restored to its previous state.

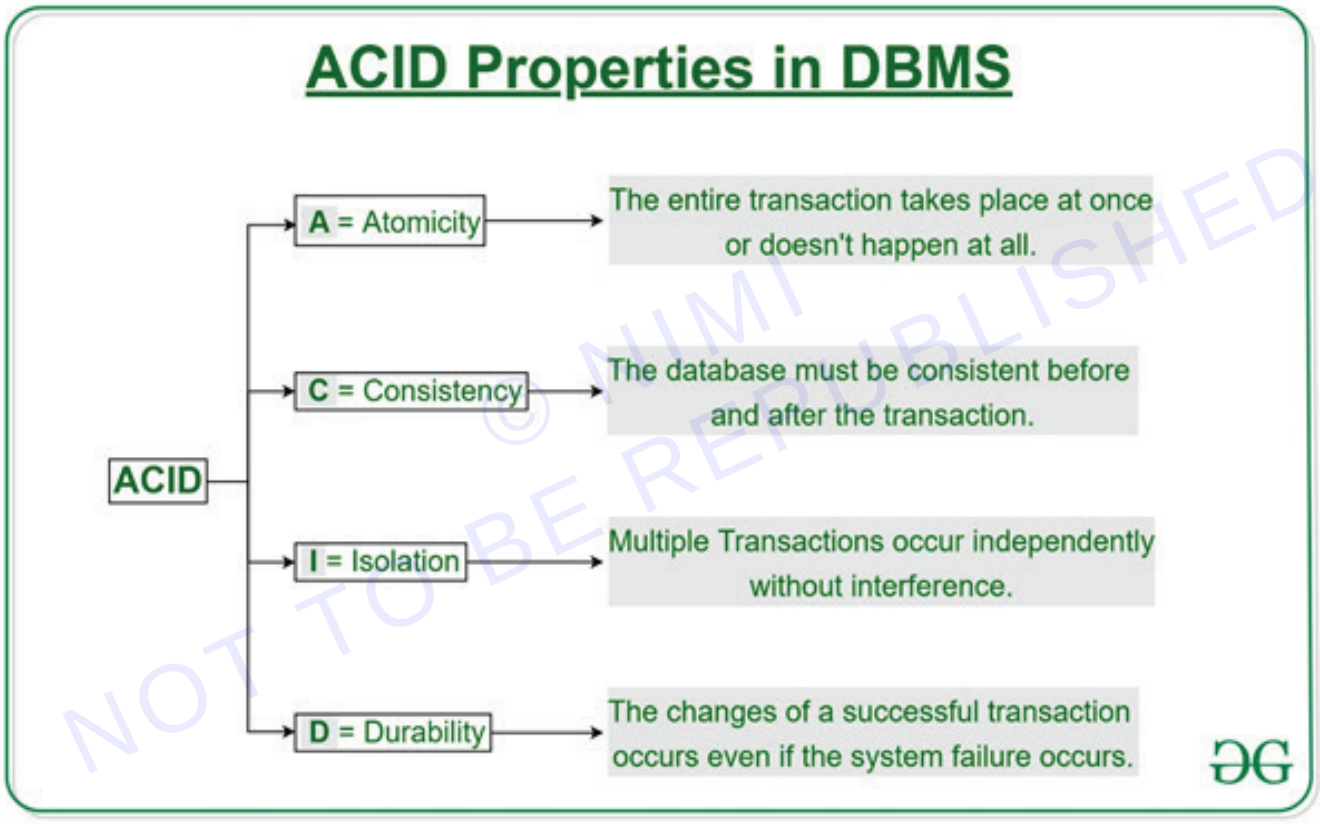
5 **Failed**: The failed state is different from the aborted state. A transaction enters the failed state when it encounters a critical error that prevents it from continuing. In this case, the DBMS automatically rolls back the transaction to maintain data integrity and consistency.

6 **Terminated (or Completed)**: After a transaction has reached a terminal state (committed, aborted, or failed), it is considered terminated. The transaction can no longer be modified or interact with the database. It may still be examined for auditing or debugging purposes.

7 **Pending**: In some systems, a transaction may enter a pending state temporarily. This typically occurs when a transaction is waiting for a resource or a lock that is held by another transaction. While in the pending state, the transaction is not actively executing but is waiting for its turn to proceed.

A transaction is a single logical unit of work that accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.

In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called ACID properties.



Atomicity

By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.

- **Abort**: If a transaction aborts, changes made to the database are not visible.
- **Commit**: If a transaction commits, changes made are visible.

Atomicity is also known as the 'All or nothing rule'.

Consider the following transaction T consisting of T1 and T2: Transfer of 100 from account X to account Y.

If the transaction fails after completion of T1 but before completion of T2. (say, after write(X) but before write(Y)), then the amount has been deducted from X but not added to Y. This results in an inconsistent database state. Therefore, the transaction must be executed in its entirety in order to ensure the correctness of the database state.

Before: X : 500	Y: 200
Transaction T	
T1	T2
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

Consistency

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database. Referring to the example above,

The total amount before and after the transaction must be maintained.

Total before T occurs = 500 + 200 = 700.

Total after T occurs = 400 + 300 = 700.

Therefore, the database is consistent. Inconsistency occurs in case T1 completes but T2 fails. As a result, T is incomplete.

Isolation

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of the database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

Let X= 500, Y = 500.

Consider two transactions T and T".

T	T"
Read (X) X: = X*100 Write (X) Read (Y) Y: = Y – 50 Write (Y)	Read (X) Read (Y) Z: = X + Y Write (Z)

Suppose T has been executed till Read (Y) and then T" starts. As a result, interleaving of operations takes place due to which T" reads the correct value of X but the incorrect value of Y and sum computed by

T": (X+Y = 50, 000+500=50, 500)

is thus not consistent with the sum at end of the transaction:

T: (X+Y = 50, 000 + 450 = 50, 450).

This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.

Durability

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

Some important points

Property	Responsibility for maintaining properties
Atomicity	Transaction Manager
Consistency	Application programmer
Isolation	Concurrency Control Manager
Durability	Recovery Manager

The ACID properties, in totality, provide a mechanism to ensure the correctness and consistency of a database in a way such that each transaction is a group of operations that acts as a single unit, produces consistent results, acts in isolation from other operations, and updates that it makes are durably stored.

ACID properties are the four key characteristics that define the reliability and consistency of a transaction in a Database Management System (DBMS). The acronym ACID stands for Atomicity, Consistency, Isolation, and Durability. Here is a brief description of each of these properties:

1 **Atomicity**: Atomicity ensures that a transaction is treated as a single, indivisible unit of work. Either all the operations within the transaction are completed successfully, or none of them are. If any part of the transaction fails, the entire transaction is rolled back to its original state, ensuring data consistency and integrity.

2 **Consistency**: Consistency ensures that a transaction takes the database from one consistent state to another consistent state. The database is in a consistent state both before and after the transaction is executed. Constraints, such as unique keys and foreign keys, must be maintained to ensure data consistency.

3 **Isolation**: Isolation ensures that multiple transactions can execute concurrently without interfering with each other. Each transaction must be isolated from other transactions until it is completed. This isolation prevents dirty reads, non-repeatable reads, and phantom reads.

4 **Durability**: Durability ensures that once a transaction is committed, its changes are permanent and will survive any subsequent system failures. The transaction's changes are saved to the database permanently, and even if the system crashes, the changes remain intact and can be recovered.

Overall, ACID properties provide a framework for ensuring data consistency, integrity, and reliability in DBMS. They ensure that transactions are executed in a reliable and consistent manner, even in the presence of system failures, network issues, or other problems. These properties make DBMS a reliable and efficient tool for managing data in modern organizations.

Advantages of ACID Properties in DBMS

1 **Data Consistency**: ACID properties ensure that the data remains consistent and accurate after any transaction execution.

2 **Data Integrity**: ACID properties maintain the integrity of the data by ensuring that any changes to the database are permanent and cannot be lost.

3 **Concurrency Control**: ACID properties help to manage multiple transactions occurring concurrently by preventing interference between them.

4 **Recovery**: ACID properties ensure that in case of any failure or crash, the system can recover the data up to the point of failure or crash.

Disadvantages of ACID Properties in DBMS

1 Performance: The ACID properties can cause a performance overhead in the system, as they **require** additional processing to ensure data consistency and integrity.

2 Scalability: The ACID properties may cause scalability issues in large distributed systems where multiple transactions occur concurrently.

3 Complexity: Implementing the ACID properties can increase the complexity of the system and require significant expertise and resources.

Overall, the advantages of ACID properties in DBMS outweigh the disadvantages. They provide a reliable and consistent approach to data

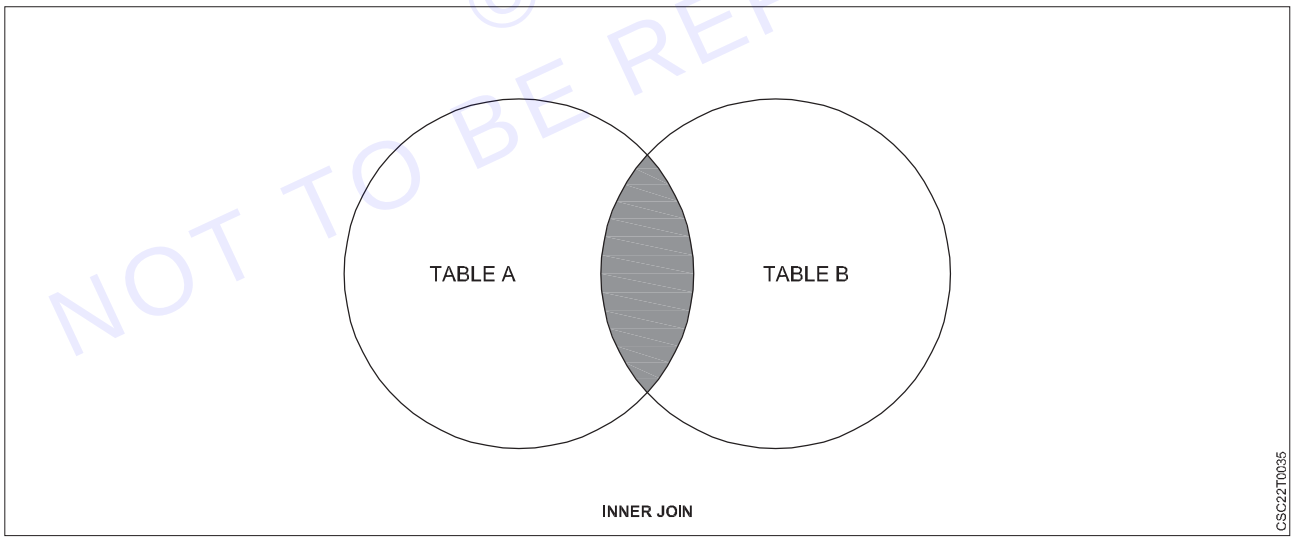
4 management, ensuring data integrity, accuracy, and reliability. However, in some cases, the overhead of implementing ACID properties can cause performance and scalability issues. Therefore, it's important to balance the benefits of ACID properties against the specific needs and requirements of the system

Joining of tables

Joining tables is a fundamental operation in relational databases that allows you to combine data from two or more tables based on a related column. This operation is crucial for querying and analyzing data stored in a database. There are several types of joins in SQL, the most common being INNER JOIN, LEFT JOIN (or LEFT OUTER JOIN), RIGHT JOIN (or RIGHT OUTER JOIN), and FULL JOIN (or FULL OUTER JOIN).

1 **INNER JOIN**

- Returns only the rows that have matching values in both tables.
- Rows that do not have a match in the other table are excluded from the result set.



Syntax

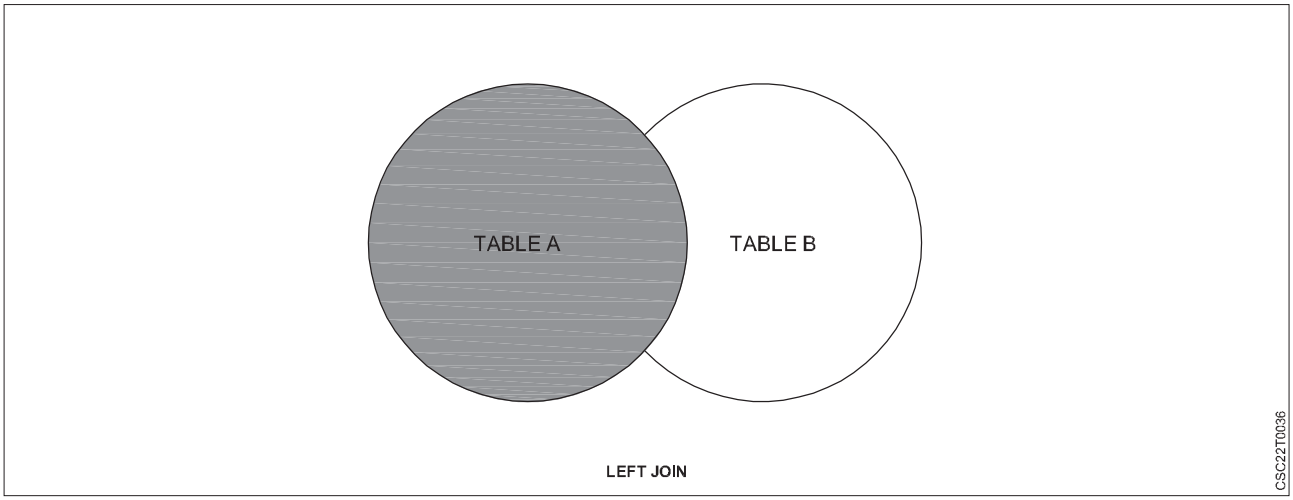
SELECT columns

FROM table1

INNER JOIN table2 ON table1.column = table2.colum

2 **LEFT JOIN (LEFT OUTER JOIN)**

- Returns all rows from the left table and the matched rows from the right table.
- If there is no match in the right table, NULL values are included for the columns from the right table.



Syntax

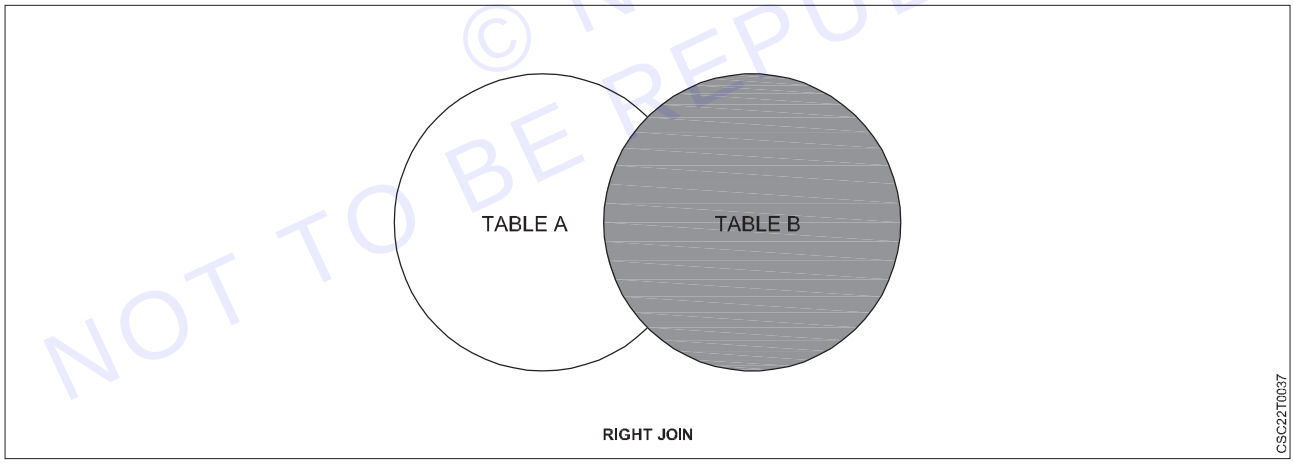
SELECT columns

FROM table1

LEFT JOIN table2 ON table1.column = table2.column;

3 **RIGHT JOIN (RIGHT OUTER JOIN)**

- Returns all rows from the right table and the matched rows from the left table.
- If there is no match in the left table, NULL values are included for the columns from the left table



Syntax

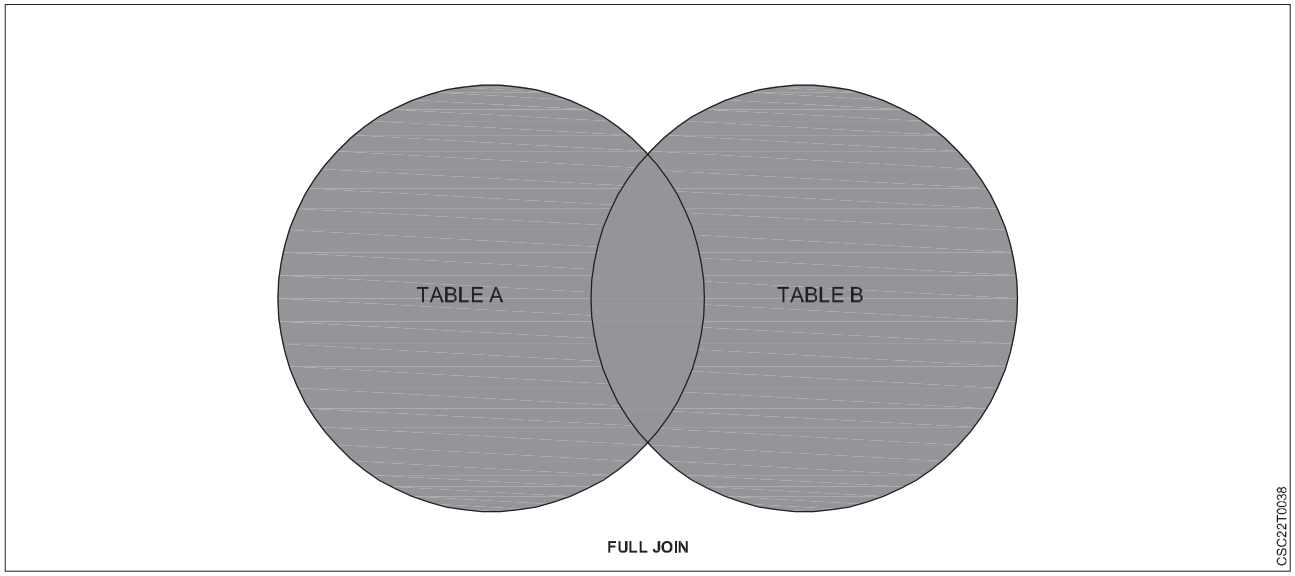
SELECT columns

FROM table1

RIGHT JOIN table2 ON table1.column = table2.column;

4 **FULL JOIN (FULL OUTER JOIN)**

- Returns all rows when there is a match in either the left or the right table.
- If there is no match in one of the tables, NULL values are included for the columns from the table without a match.



Syntax

SELECT columns

FROM table1

FULL JOIN table2 ON table1.column_name = table2.column_name;

When performing a join, it's essential to specify the columns on which you want to join the tables. These columns should have compatible data types or can be explicitly converted to compatible data types.

Here's a simple example using two hypothetical tables, "orders" and "customers," to demonstrate an INNER JOIN:

SELECT orders.order_id, customers.customer_name

FROM orders

INNER JOIN customers ON orders.customer_id = customers.customer_id;

This query retrieves the order IDs and customer names for orders that have matching customer IDs in both the "orders" and "customers" tables.

Sub Queries

A subquery in SQL is essentially a query nested within another query, commonly found within the WHERE clause of the main SQL query. Here are some fundamental rules and guidelines for using subqueries:

1 **Purpose of Subqueries**: Subqueries are employed to retrieve data that will be used by the main query for filtering, comparison, or as a part of calculations.

2 **Location**: Subqueries are typically placed within parentheses and inside the WHERE clause of the main query. They can also be used in other clauses like SELECT, FROM, or HAVING when necessary.

3 **Comparison Operators** : Subqueries often use comparison operators such as =, <, >, <=, >=, or IN to relate the results of the subquery with the main query.

4 **Single-Row or Multi-Row Results** : Subqueries can return either a single value or a set of values (single-row or multi-row results). The choice depends on the specific use case and the operator used.

5 **Subquery Types** : Subqueries can be categorized into various types, including scalar subqueries (returning a single value), single-row subqueries (returning one row), and multi-row subqueries (returning multiple rows).

6 **Alias and Correlation** : When using subqueries in the SELECT clause, it's often necessary to assign an alias to the subquery. Additionally, correlated subqueries refer to the main query's tables, allowing for more complex relationships between the subquery and the main query.

7 **Performance Considerations** : Be cautious when using subqueries, especially correlated subqueries, as they can impact query performance. In some cases, alternative methods like JOINs or CTEs (Common Table Expressions) might be more efficient.

8 **Subquery Limitations** : Some database systems have limitations on the complexity and nesting depth of subqueries. Always consult the documentation for your specific database system to understand its capabilities and limitations regarding subqueries.

Syntax

While there isn't a single, universal syntax for subqueries in SQL because their usage can vary depending on the specific query and database system, subqueries are commonly used in conjunction with the SELECT statement. Here's a general template for a subquery within a SELECT statement:

SELECT column1, column2, ...

FROM table1

WHERE columnN operator (SELECT columnX FROM tableY WHERE condition);

In this template:

- column1, column2, etc., represent the columns you want to retrieve in your main query.
- table1 is the main table you're querying.
- columnN is a column from table1 that you want to compare using an operator.
- operator is a comparison operator like =, <, >, IN, etc.
- The subquery within parentheses (SELECT columnX FROM tableY WHERE condition) is embedded in the WHERE clause and retrieves data from another table (tableY) based on a specific condition.

Subqueries with the INSERT Statement

Subqueries can also be used with the INSERT statement in SQL to insert data into a table based on the results of a subquery. This allows you to populate a table with data derived from another table or the result of a subquery. Here's a basic syntax for using subqueries with the INSERT statement:

INSERT INTO target_table (column1, column2, ...)

SELECT expression1, expression2, ...

FROM source_table

WHERE condition;

In this syntax:

- target_table is the table into which you want to insert data.
- column1, column2, etc., are the columns in target_table where you want to insert data.
- expression1, expression2, etc., are expressions or values that you want to insert into the corresponding columns in target_table.
- source_table is the table or subquery that provides the data you want to insert.
- condition is an optional condition that filters the data from the source_table before insertion.

Here's an example to illustrate how you might use a subquery with the INSERT statement:

Let's say you have two tables, employees and new_hires, and you want to insert data into the employees table from the new_hires table based on a certain condition:

INSERT INTO employees (employee_id, employee_name, salary)

SELECT new_id, new_name, new_salary

FROM new_hires

WHERE new_salary > 50000;

In this example, the INSERT statement retrieves data from the new_hires table, specifically the employee_id, employee_name, and salary columns, but only for records where the new_salary is greater than 50,000. It then inserts this data into the employees table

Subqueries with the SELECT Statement

Subqueries can be used with the SELECT statement in SQL to retrieve data from one or more tables based on the results of a nested query. This allows you to create more complex and dynamic queries by incorporating the results of one query into another. Here's the basic syntax for using subqueries with the SELECT statement:

SELECT column1, column2, ...

FROM table1

WHERE columnN operator (SELECT columnX FROM tableY WHERE condition);

In this syntax:

- column1, column2, etc., represent the columns you want to retrieve in your main query.
- table1 is the main table you're querying.
- columnN is a column from table1 that you want to compare using an operator.
- operator is a comparison operator like '=', '<',' >', 'IN', etc.
- The subquery within parentheses (SELECT columnX FROM tableY WHERE condition) is embedded in the WHERE clause and retrieves data from another table (tableY) based on a specific condition.

Here's an example to illustrate how you might use a subquery with the SELECT statement:

Suppose you have two tables, employees and salaries, and you want to retrieve the names of employees who earn a salary greater than the average salary in the salaries table:

ELECT employee_name

FROM employees

WHERE employee_id IN (SELECT employee_id FROM salaries WHERE salary > (SELECT AVG(salary) FROM salaries));

In this example, the main SELECT statement retrieves the employee_name from the employees table for employees whose employee_id matches the result of the subquery. The subquery calculates the average salary from the salaries table and then filters employees who earn more than the calculated average.

Subqueries with the UPDATE Statement

Subqueries can also be used with the UPDATE statement in SQL to modify data in a table based on the results of a subquery. This allows you to update records in one table using information from another table or using the results of a subquery. Here's a basic syntax for using subqueries with the UPDATE statement:

UPDATE target_table

SET column1 = value1, column2 = value2, ...

WHERE columnN operator (SELECT columnX FROM source_table WHERE condition);

In this syntax:

- target_table is the table you want to update.
- column1, column2, etc., are the columns in target_table that you want to update.
- value1, value2, etc., are the new values you want to set for the corresponding columns.
- columnN is a column in target_table that you want to compare using an operator.
- operator is a comparison operator like =, <, >, IN, etc.
- The subquery within parentheses (SELECT columnX FROM source_table WHERE condition) is embedded in the WHERE clause and retrieves data from another table (source_table) based on a specific condition.

Here's an example to illustrate how you might use a subquery with the UPDATE statement:

Suppose you have two tables, orders and customers, and you want to update the customer_id in the orders table based on a customer's name from the customers table

UPDATE orders

SET customer_id = (SELECT customer_id FROM customers WHERE customer_name = 'John Doe')

WHERE order_id = 123;

In this example, the UPDATE statement updates the customer_id column in the orders table with the result of the subquery. The subquery retrieves the customer_id from the customers table for the customer with the name 'John Doe,' and this value is used to update the specified order (order_id = 123) in the orders table.

Subqueries with the DELETE Statement

Subqueries can also be used with the DELETE statement in SQL to remove rows from a table based on the results of a subquery. This allows you to delete specific records in a table using information derived from another table or a subquery's results. Here's a basic syntax for using subqueries with the DELETE statement:

DELETE FROM target_table

WHERE columnN operator (SELECT columnX FROM source_table WHERE condition);

In this syntax:

- target_table is the table from which you want to delete rows.
- columnN is a column in target_table that you want to compare using an operator.
- operator is a comparison operator like =, <, >, IN, etc.
- The subquery within parentheses (SELECT columnX FROM source_table WHERE condition) is embedded in the WHERE clause and retrieves data from another table (source_table) based on a specific condition.
- condition is an optional condition that filters the data from the source_table before deletion.

Here's an example to illustrate how you might use a subquery with the DELETE statement:

Suppose you have two tables, employees and salaries, and you want to delete employees from the employees table whose salary exceeds the average salary in the salaries table:

DELETE FROM employees

WHERE employee_id IN (SELECT employee_id FROM salaries WHERE salary > (SELECT AVG(salary) FROM salaries));

In this example, the DELETE statement removes rows from the employees table where the employee_id matches the result of the subquery. The subquery calculates the average salary from the salaries table and filters employees with salaries greater than the calculated average.

Functions used in query like sum, average, max, min, count etc

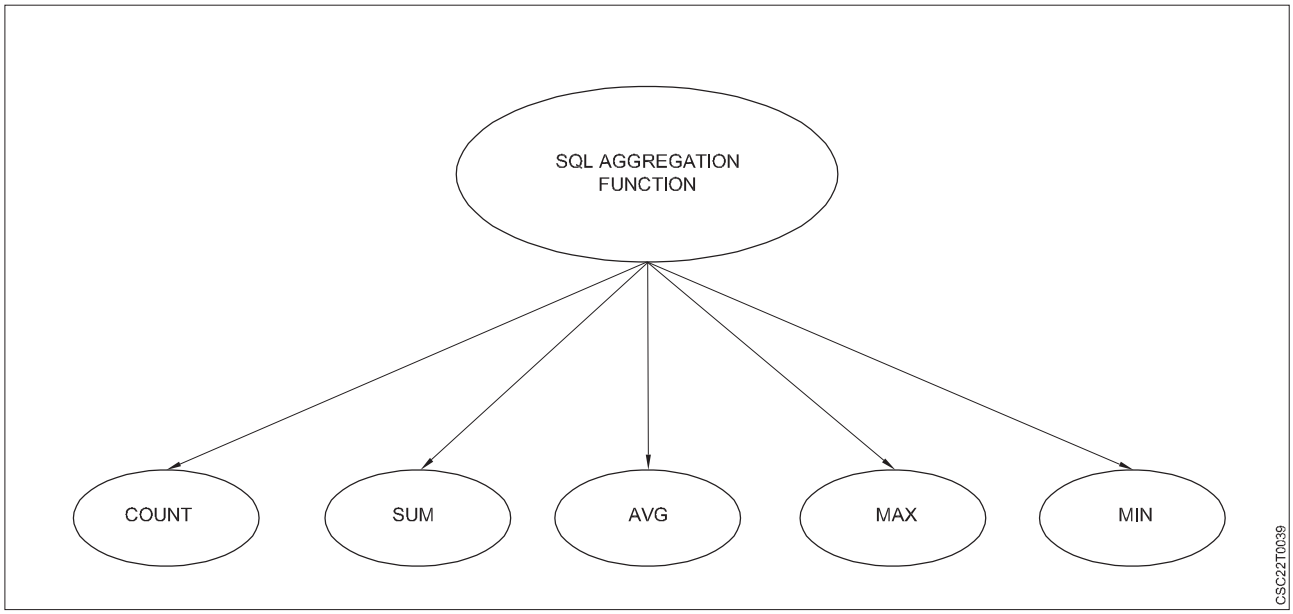
Functions like sum, average, max, min, count, and others are commonly used in various database query languages, such as SQL, to perform calculations and aggregations on data. Here's a brief overview of these functions:

1 **SUM**: The SUM function is used to calculate the total of a numeric column in a database table. For example, you can use it to find the total sales for a particular product.

SELECT SUM(sales_amount) FROM sales_data;

2 **AVERAGE (AVG)**: The AVERAGE function, often abbreviated as AVG, calculates the average value of a numeric column.

SELECT AVG(temperature) FROM weather_data



3 **MAX**: The MAX function returns the maximum value in a specified column. It is useful for finding the highest value in a dataset.

SELECT MAX(score) FROM exam_scores;

4 **MIN**: The MIN function returns the minimum value in a specified column. It is used to find the lowest value in a dataset.

SELECT MIN(price) FROM product_prices;

5 **COUNT**: The COUNT function counts the number of rows that meet certain criteria. It can be used to count all rows in a table or to count rows that meet specific conditions.

SELECT COUNT(*) FROM employees;-- Count all employees

SELECT COUNT(*) FROM order WHERE sus = 'Shipped'; -- Count shipped orders

Certainly! Here are some common SQL functions used in queries like SUM, AVERAGE, MAX, MIN, and COUNT, along with examples using a hypothetical table called sales:

Assume the sales table has the following structure:

order_id	product_id	quantity	price
1	101	5	10
2	102	3	15
3	101	2	10
4	103	4	12
5	102	6	15

1 **SUM**: Calculates the sum of values in a column.

Example: Calculate the total revenue.

SELECT SUM(quantity * price) AS total_revenue FROM sales;

Result:

total_revenue

180.0

2 **AVERAGE**: Calculates the average of values in a column.

Example: Calculate the average price of products.

SELECT AVG(price) AS avg_price FROM sales;

Result:

avg_price

12.4

3 **MAX**: Finds the maximum value in a column.

Example: Find the maximum quantity sold.

SELECT MAX(quantity) AS max_quantity FROM sales;

Result:

max_quantity

6

4 **MIN**: Finds the minimum value in a column.

10.0

5 **COUNT**: Counts the number of rows in a table or the number of non-null values in a column.

Example: Count the number of orders.

SELECT COUNT(*) AS order_count FROM sales

Result:

order_count

5

Example-2

PRODUCT_GOOD

PRODUCT	COMPANY	QTY	RATE	COST
Item1	ComA	3	100	200
Item2	ComB	4	250	750
Item3	ComA	5	300	600
Item4	ComC	6	100	500
Item5	ComB	3	200	400
Item6	ComA	4	250	750
Item7	ComA	6	300	2550
Item8	ComA	4	100	300
Item9	ComB	3	250	500
Item10	ComC	5	300	1250

Example: COUNT()

SELECT COUNT(*)

FROM PRODUCT_GOOD;

Output:

10

Example: COUNT with WHERE

SELECT COUNT(*)

FROM PRODUCT_GOOD;

WHERE RATE>=200;

Output:

7

Example: COUNT() with DISTINCT

SELECT COUNT(DISTINCT COMPANY)

FROM PRODUCT_GOOD;

Output:

3

Example: COUNT() with GROUP BY

SELECT COMPANY, COUNT(*)

FROM PRODUCT_GOOD

Output:

ComA 5

ComB 3

ComC 2

Example: COUNT() with HAVING

SELECT COMPANY, COUNT(*)

SELECT COMPANY, COUNT(*)

GRHAV2;

Output:

ComA 5

ComB 3

Indexing and Query Optimization

Absolutely, your analogy of indexing to the index in a book is quite apt. It provides a clear and intuitive understanding of how indexing works in databases.

In the context of databases, creating an index is akin to building a reference point for the data. Much like flipping through a book's index to quickly locate information, a database index allows for the rapid retrieval of specific data without scanning through the entire dataset. This indexing technique significantly enhances the speed and efficiency of queries, as the database can swiftly pinpoint the relevant information, leading to faster query performance.

The intricacies of how databases achieve this efficiency through indexing can be explored further in the upcoming sections of your article.

How to Create Indexes

Creating indexes in a relational database involves using the CREATE INDEX statement.

The syntax may vary slightly between different database management systems (DBMS)

Syntax:

CREATE INDEX [index_name]

ON [table_name] ([column_name]);

Query:

CREATE INDEX product_category_index

ON product (category);

When you run this query, it will experience a prolonged execution time compared to a standard query. This is because the database is scanning through a substantial 12 million rows and constructing a new 'category' index from the ground up, a process that takes approximately 4 minutes.

Now, let's assess how the performance of the original query improves after the implementation of indexing.

SELECT COUNT(*)

FROM product

WHERE category = 'electronics';

You'll observe a significant improvement in the query's speed this time. It is likely to complete in a much shorter timeframe, perhaps around 400 milliseconds.

Moreover, the positive impact of indexing on 'category' extends beyond just queries explicitly involving this condition. To illustrate, let's consider a scenario where queries involve additional conditions beyond 'category'—even these queries will experience enhanced performance due to the indexing on 'category'.

SELECT COUNT(*)

FROM product

WHERE category = 'electronics'

AND product_subcategory = 'headphone';

In this case, the query's execution time is expected to be reduced compared to its normal duration, perhaps completing in around 600 milliseconds. The database can efficiently locate all 'electronics' products using the index, resulting in a smaller set of records. Subsequently, it can then identify 'headphones' from this narrowed-down set in the usual manner.

Now, let's explore the impact of changing the order of conditions in the 'WHERE' clause.

SELECT COUNT(*)

FROM product

WHERE product_subcategory = 'headphone'

AND category = 'electronics';

Types of Indexing

Exploring the realm of database indexing involves delving into two primary types

1 **Clustered Index**

- A clustered index stands as the unique index for a table, employing the primary key to structure the data within that table. Unlike a non-clustered index, a clustered index doesn't require explicit declaration; instead, it is automatically generated when the primary key is defined. By default, the clustered index utilizes the ascending order of the primary key for organization.

These clustered indexes play a pivotal role in shaping the physical arrangement of data within the table, facilitating efficient retrieval and storage operations.

Let me demonstrate this with an easy example.

product_pkey

product_id
1
2
3
4

product

product_id	product_name	product_subcategory	brand	category	price
1	A	headphone	Sony	electronics	$280
2	B	sneaker	Nike	shoes	$70
3	C	shirt	Levi's	clothing	$50
4	D	baseball bat	Louisville Slugger	sports	$100

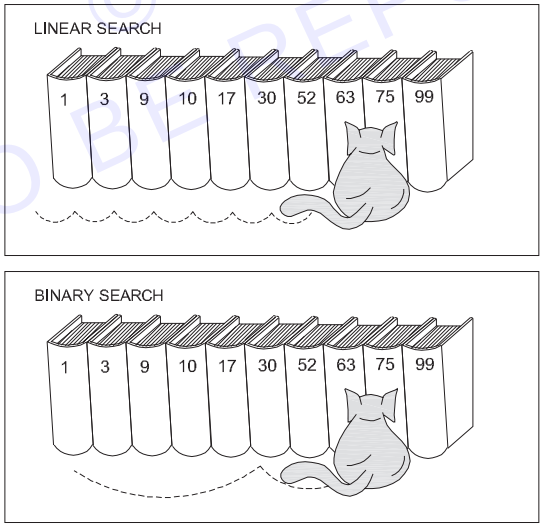
The 'product' table will automatically come with a clustered index named 'product_pkey,' and this index is structured around the primary key 'product_id.'

Now, when you run a query to search the table by 'product_id' (like in the query below), the clustered index will help the database to perform optimal searches, and return the result faster.

```
SELECT product_name, category, price
FROM product
WHERE product_id = 3;
```

You might be curious about how it accomplishes this. Indices employ an efficient search technique called binary search.

Finding Book #75



FINDING BOOK #75

CSC22T0040

Binary search stands out as a highly effective algorithm for locating an item within a sorted list. Its methodology involves iteratively dividing the data in half and assessing whether the target entry, sought through a query, is positioned before or after the middle entry in the dataset.

If the query value is less than the middle entry, the search narrows down to the lower half; otherwise, it focuses on the upper half.

This process continues until the desired value is located. The brilliance of binary search lies in its ability to minimize the number of searches required, resulting in faster query execution.

The following table helps to understand the impact of binary search in terms of number of searches:

Number of data entries	Maximum no. of searches to find target ($log_2 n$)
8	3
100	7
1,000	10
10,000	14
100,000	17
1,000,000	20

Likewise, in the case of our dataset containing 12 million rows, employing a binary search would necessitate a maximum of 24 searches, as opposed to the worst-case scenario of 12 million searches.

This underscores the formidable efficiency and power of indexes in optimizing data retrieval processes.

2 Non-clustered Index

Now, the challenge is to extend the benefits of indexing beyond the primary key, and the solution lies in non-clustered indexes.

All the queries we initially explored to enhance query performance relied on non-clustered indexes—indexes that need to be explicitly defined.

A non-clustered index is distinct in that it's stored separately from the actual data in the table. It operates much like the index page of a book, as mentioned earlier. The index page is situated in one location, while the contents of the book are in another. This design permits the inclusion of more than one non-clustered index per table, as we discussed earlier.

But how is this achieved?

Consider crafting a query that involves searching for an entry in a column for which you've already established a non-clustered index. This type of index inherently encompasses:

1 Column entries for which the index is created.

2 Addresses of the corresponding rows in the main table to which the column entries belong.

You can see this visually in the left mini-table in the figure:

Product_category_index

Category	Product_id
Clothing	3
electronics	1
sports	4
shoes	2

Product

Product_ id	product_ name	product_ subcategory	brand	category	price
1	A	head phone	sony	electronices	$280
2	B	sneaker	Nike	shoes	$70
3	C	Shirt	Levi's	dothing	$50
4	D	baseball bat	Louisville slugger	sports	$100

Let me explain this using a query.

CREATE INDEX product_category_index

ON product (category);

SELECT product_name, category, price

FROM product

WHERE category = ‘electronics’;

The database operates through three key steps:

1 **Firstly**: It navigates to the non-clustered index (in this case, ‘product_category_index’), pinpointing the column entry you searched for (e.g., category = ‘electronics’) using the efficient binary search method.

2 **Secondly**: It seeks the address of the corresponding row in the main table that corresponds to the identified column entry.

3 **Finally**: It accesses that specific row in the main table, retrieving additional column values as needed for your query (e.g., product_name, price).

It’s important to note that a non-clustered index involves an additional step compared to a clustered index—it requires finding the address and then going to the corresponding row in the main table. This additional step makes non-clustered indexes relatively slower than their clustered counterparts.

```
CREATE TABLE demo(
   id INT NOT NULL,
   first_name VARCHAR(25) NOT NULL,
   last_name VARCHAR(35) NOT NULL,
   age INT NOT NULL,
   PRIMARY KEY(id)
);
/* Index on First Name */
CREATE INDEX demo_fname ON demo (first_name);
/* Will tell us whether our query uses the intended index */
explain SELECT * FROM demo WHERE first_name = “Donald” \G
```

Output

```
*************************** 1. row ***************************
id: 1
select_type: SIMPLE
table: demo
partitions: NULL
type: ref
possible_keys: demo_fname
key: demo_fname
key_len: 27
ref: const
rows: 1
```

filtered: 100.00 Extra: NULL

To harness the index’s benefits, it’s crucial to isolate the column, ensuring it’s not incorporated into a function or expression.

Introduction to Stored Procedures

Stored procedures are a feature in SQL that allows you to define and store a set of SQL statements as a named procedure in a database. These procedures can be called and executed by applications, users, or other database objects. Stored procedures are commonly used for encapsulating business logic, improving code reusability, and enhancing database security. Here's an overview of stored procedures in SQL:

Creating a Stored Procedure

The syntax for creating a stored procedure can vary slightly between different SQL database management systems (DBMS), but the general structure is similar. Below is a simplified example in generic SQL syntax:

CREATE PROCEDURE procedure_name ([parameter_list])

AS BEGIN

-- SQL statements to define the procedure's logic

END;

- procedure_name: This is the name you give to your stored procedure.
- [parameter_list]: You can define input parameters that the procedure can accept. These parameters are optional.

Example of Creating a Simple Stored Procedure in SQL Server:

CREATE PROCEDURE GetEmployee @EmployeeID INT

AS

BEGIN

SELECT * FROM Employees WHERE EmployeeID = @EmployeeID;

END;

In this example, we've created a stored procedure called GetEmployee that takes an EmployeeID as an input parameter and selects the corresponding employee from the Employees table in SQL Server.

Executing a Stored Procedure

To execute a stored procedure, you can use the EXEC or CALL statement, depending on the specific SQL DBMS you are using.

- In SQL Server, you use EXEC:

  EXEC GetEmployee @EmployeeID = 123;

In MySQL, you use CALL:

CALL GetEmployee(123);

Modifying a Stored Procedure:

You can modify an existing stored procedure using the ALTER PROCEDURE statement in SQL Server, or the ALTER PROCEDURE command in other DBMS, which can vary slightly.

Dropping a Stored Procedure:

To remove a stored procedure, you use the DROP PROCEDURE statement:

DROP PROCEDURE IF EXISTS GetEmployee;

Control Flow in Stored Procedures:

Stored procedures support various control flow constructs like IF, ELSEIF, ELSE, CASE, loops (e.g., WHILE, LOOP, REPEAT), and exception handling using BEGIN...END blocks, depending on the specific DBMS.

Security Considerations:

You can control access to stored procedures by granting or revoking execution privileges to specific users or roles. Properly managing permissions is essential to ensure that only authorized users can execute these procedures

Stored procedures are a valuable feature in SQL that allows you to encapsulate complex database operations, enforce security, and promote code reusability. They are widely used in database-driven applications and are an important part of database management and development.

Creating Trigger, Creating Cursor, Using Cursor

Introduction to Triggers

Certainly! Triggers in the context of databases refer to special types of stored procedures that automatically execute in response to specific events on a particular table or view. These events typically involve data manipulation operations, such as INSERT, UPDATE, DELETE, or even certain schema-level events like CREATE, ALTER, or DROP.

“A trigger is an automated code segment, acting as a procedure, that is executed in response to specific events occurring in a table or view within a database. In contrast, a cursor is a control structure utilized in databases for traversing through records. It’s noteworthy that a cursor can be declared and employed within the context of a trigger “

Key Concepts:

1 **Events**

- **INSERT**: Triggered after a new row is added to the table.
- **UPDATE**: Triggered after one or more existing rows are modified.
- **DELETE**: Triggered after one or more rows are removed from the table.

2 **Timing**

- **BEFORE Triggers**: Executed before the triggering event, allowing modification of the data before it is actually written to the database.
- **AFTER Triggers**: Executed after the triggering event has occurred.

3 **Row-Level and Statement-Level Triggers**

- **Row-Level Triggers**: Executed once for each row affected by the triggering event.
- **Statement-Level Triggers**: Executed once for each triggering event, regardless of the number of rows affected.

Use Cases

1 **Data Validation**

- Ensure that certain conditions are met before allowing data changes.

2 **Enforcing Business Rules**

- Implement complex business logic or rules automatically.

3 **Audit Trails**

- Log changes made to a table for auditing purposes.

4 **Cascade Operations**

- Automatically perform additional operations on other tables when a specified event occurs.

5 **Synchronization**

- Keep multiple tables in sync by triggering actions in response to changes in one table.

Syntax (for an AFTER INSERT Trigger):

CREATE TRIGGER trigger_name

AFTER INSERT ON table_name

```
FOR EACH ROW
BEGIN
  -- Trigger logic here
END;
```

- trigger_name: The name you assign to the trigger.
- AFTER INSERT ON table_name: Specifies the timing and event that trigger the execution.
- FOR EACH ROW: Indicates that the trigger will be executed once for each row affected by the triggering event.
- BEGIN...END: The block where you define the logic of the trigger.

Example:

Let's consider a simple example where we want to create an audit trail for an employees table:

```
CREATE TRIGGER after_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
INSERT INTO employee_audit (employee_id, action, timestamp)
VALUES (NEW.employee_id, 'INSERT', NOW());
END;
```

In this example, every time a new row is inserted into the employees table, the trigger logs the action into an employee_audit table with details like the employee ID, the action performed (INSERT), and the timestamp.

Triggers are powerful tools, but they should be used with caution to avoid unintended consequences and to ensure they do not negatively impact database performance.

Introduction to Cursor

A cursor in the context of databases is a programming construct or a database object that enables the traversal and manipulation of records in a result set. It provides a mechanism for iterating over a set of rows returned by a SQL query. Cursors are often used in procedural languages, such as PL/SQL or T-SQL, to perform operations on a row-by-row basis.

There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors.

1 **Implicit Cursors**, often referred to as the Default Cursors of SQL Server, are automatically assigned by SQL Server when users execute Data Manipulation Language (DML) operations. These cursors are generated by the system without explicit declaration by the user.

2 **explicit cursors**: Users create explicit cursors when needed. These cursors are specifically crafted by users for the purpose of retrieving data from a table in a row-by-row manner.

How To Create Explicit Cursor?

1 Declare Cursor Object

Syntax:

```
DECLARE cursor_name CURSOR FOR
SELECT * FROM table_name
```

Query:

```
DECLARE s1 CURSOR FOR
SELECT * FROM studDetails
```

2 Open Cursor Connection

Syntax:

OPEN cursor_connection

Query:

OPEN s

3 Close cursor connection

Syntax:

CLOSE cursor_name

Query:

CLOSE s1

4 Deallocate cursor memory

Syntax:

DEALLOCATE cursor_name

Query:

DEALLOCATE s1

How To Create an Implicit Cursor?

Syntax (for a Simple Cursor):

```
-- Cursor Declaration
DECLARE cursor_name CURSOR FOR
SELECT column1, column2
FROM table_name
WHERE condition;
-- Cursor Opening
OPEN cursor_name;
-- Cursor Fetching and Processing
FETCH NEXT FROM cursor_name INTO variable1, variable2;
-- Loop through the result set
WHILE @@FETCH_STATUS = 0
BEGIN
  -- Process the current row (variable1, variable2)
  -- Fetch the next row
  FETCH NEXT FROM cursor_name INTO variable1, variable2;
END;
-- Cursor Closing
CLOSE cursor_name;
```

Example:

Suppose we want to process each employee's name and salary from an employees table:

```
DECLARE emp_cursor CURSOR FOR
```

```
SELECT employee_name, salary
FROM employees;
OPEN emp_cursor;
FETCH NEXT FROM emp_cursor INTO @employee_name, @salary;
WHILE @@FETCH_STATUS = 0
BEGIN
  -- Process the current employee (e.g., print or update)
  PRINT CONCAT('Employee: ', @employee_name, ', Salary: ', @salary);
  -- Fetch the next employee
  FETCH NEXT FROM emp_cursor INTO @employee_name, @salary;
END;
CLOSE emp_cursor;
```

In this example, the cursor iterates through the result set of the query, fetching the employee name and salary for each row and then processing them within a loop. Cursors provide a flexible mechanism for handling individual rows in a result set within procedural code

Creating Triggers

In this chapter, we will discuss Triggers in PL/SQL. Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events −

- **A database manipulation (DML)** statement (DELETE, INSERT, or UPDATE)
- **A database definition (DDL)** statement (CREATE, ALTER, or DROP).
- **A database operation** (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes −

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

The syntax for creating a trigger is −

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
```

```
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
Declaration-statements
BEGIN
Executable-statements
EXCEPTION
Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the trigger_name.
- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
- [OF col_name] − This specifies the column name that will be updated.
- [ON table_name] − This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

Example

To start with, we will be using the CUSTOMERS table we had created and used in the previous chapters −

Select * from customers;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

The following program creates a row-level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values −

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

Trigger created.

The following points need to be considered here −

- OLD and NEW references are not available for table-level triggers, rather you can use them for record-level triggers.
- If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.
- The above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using the DELETE operation on the table.

Triggering a Trigger

Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table −

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (7, 'Kriti', 22, 'HP', 7500.00 );
```

When a record is created in the CUSTOMERS table, the above create trigger, display_salary_changes will be fired and it will display the following result −

Old salary:

New salary: 7500

Salary difference:

Because this is a new record, old salary is not available and the above result comes as null. Let us now perform one more DML operation on the CUSTOMERS table. The UPDATE statement will update an existing record in the table −

```
UPDATE customers
SET salary = salary + 500
WHERE id = 2;
```

When a record is updated in the CUSTOMERS table, the above create trigger, display_salary_changes will be fired and it will display the following result −

Old salary: 1500

New salary: 2000

Salary difference: 500

Creating Cursor

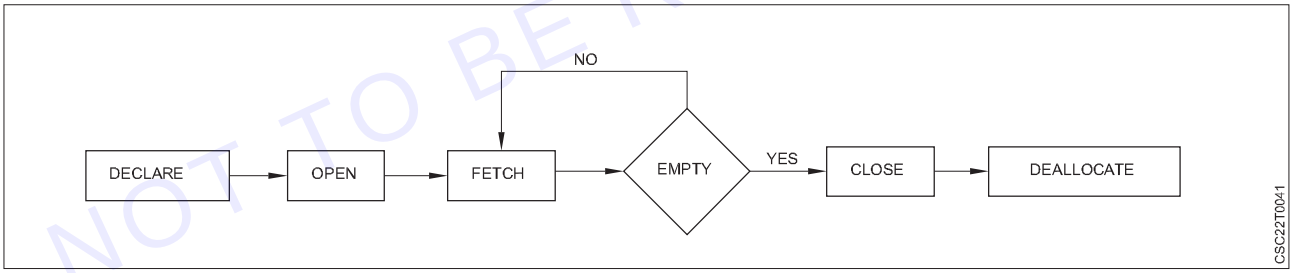
A cursor in SQL Server is a database object that allows us to retrieve each row at a time and manipulate its data. A cursor is nothing more than a pointer to a row. It's always used in conjunction with a SELECT statement. It is usually a collection of SQL logic that loops through a predetermined number of rows one by one. A simple illustration of the cursor is when we have an extensive database of worker's records and want to calculate each worker's salary after deducting taxes and leaves.

The SQL Server cursor's purpose is to update the data row by row, change it, or perform calculations that are not possible when we retrieve all records at once. It's also useful for performing administrative tasks like SQL Server database backups in sequential order. Cursors are mainly used in the development, DBA, and ETL processes.

This article explains everything about SQL Server cursor, such as cursor life cycle, why and when the cursor is used, how to implement cursors, its limitations, and how we can replace a cursor.

Life Cycle of the cursor

We can describe the life cycle of a cursor into the five different sections as follows:



1 **Declare Cursor**

The first step is to declare the cursor using the below SQL statement:

- DECLARE cursor_name CURSOR
- FOR select_statement;

We can declare a cursor by specifying its name with the data type CURSOR after the DECLARE keyword. Then, we will write the SELECT statement that defines the output for the cursor.

2 **Open Cursor**

It's a second step in which we open the cursor to store data retrieved from the result set. We can do this by using the below SQL statement:

1 OPEN cursor_name;

3 **Fetch Cursor**

It's a third step in which rows can be fetched one by one or in a block to do data manipulation like insert,

update, and delete operations on the currently active row in the cursor. We can do this by using the below SQL statement:

```
1  FETCH NEXT FROM cursor INTO variable_list;
```

We can also use the @@FETCHSTATUS function in SQL Server to get the status of the most recent FETCH statement cursor that was executed against the cursor. The FETCH statement was successful when the @@FETCHSTATUS gives zero output. The WHILE statement can be used to retrieve all records from the cursor. The following code explains it more clearly:

```
1  WHILE @@FETCH_STATUS = 0
2  BEGIN
3  FETCH NEXT FROM cursor_name;
4  END;
```

4 Close Cursor

It's a fourth step in which the cursor should be closed after we finished work with a cursor. We can do this by using the below SQL statement:

```
1.      CLOSE cursor_name;
```

5 **Deallocate Cursor**

It is the fifth and final step in which we will erase the cursor definition and release all the system resources associated with the cursor. We can do this by using the below SQL statement:

```
1  DEALLOCATE cursor_name;
```

Uses of SQL Server Cursor

We know that relational database management systems, including SQL Server, are excellent in handling data on a set of rows called result sets. For example, we have a table product_table that contains the product descriptions. If we want to update the price of the product, then the below 'UPDATE' query will update all records that match the condition in the 'WHERE' clause:

```
1  UPDATE product_table SET unit_price = 100 WHERE product_id = 105;
```

Sometimes the application needs to process the rows in a singleton fashion, i.e., on row by row basis rather than the entire result set at once. We can do this process by using cursors in SQL Server. Before using the cursor, we must know that cursors are very bad in performance, so it should always use only when there is no option except the cursor.

The cursor uses the same technique as we use loops like FOREACH, FOR, WHILE, DO WHILE to iterate one object at a time in all programming languages. Hence, it could be chosen because it applies the same logic as the programming language's looping process.

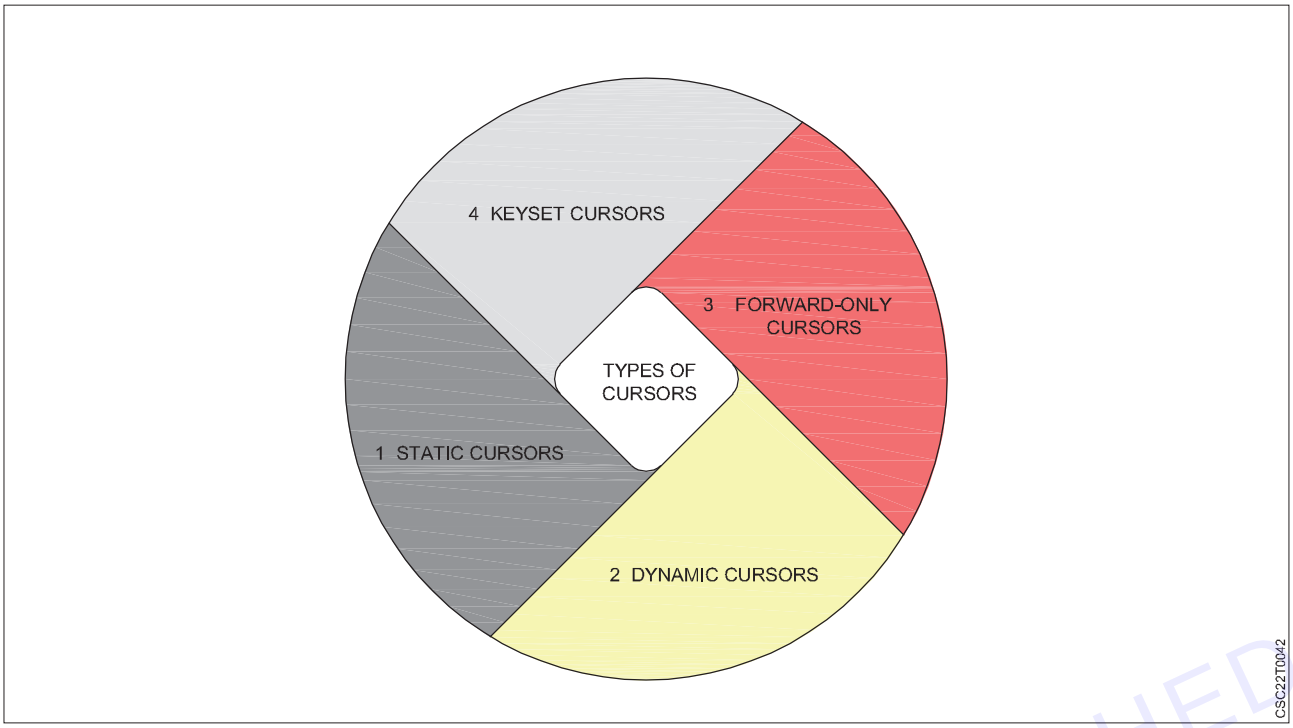
Types of Cursors in SQL Server

The following are the different types of cursors in SQL Server listed below:

- Static Cursors
- Dynamic Cursors
- Forward-Only Cursors
- Keyset Cursors

Static Cursors

The result set shown by the static cursor is always the same as when the cursor was first opened. Since the static cursor will store the result in tempdb, they are always read-only. We can use the static cursor to move both forward and backward. In contrast to other cursors, it is slower and consumes more memory. As a result, we can use it only when scrolling is necessary, and other cursors aren't suitable.



This cursor shows rows that were removed from the database after it was opened. A static cursor does not represent any INSERT, UPDATE, or DELETE operations (unless the cursor is closed and reopened).

Dynamic Cursors

The dynamic cursors are opposite to the static cursors that allow us to perform the data updation, deletion, and insertion operations while the cursor is open. It is scrollable by default. It can detect all changes made to the rows, order, and values in the result set, whether the changes occur inside the cursor or outside the cursor. Outside the cursor, we cannot see the updates until they are committed.

Forward-Only Cursors

It is the default and fastest cursor type among all cursors. It is called a forward-only cursor because it moves only forward through the result set. This cursor doesn't support scrolling. It can only retrieve rows from the beginning to the end of the result set. It allows us to perform insert, update, and delete operations. Here, the effect of insert, update and delete operations made by the user that affect rows in the result set are visible as the rows are fetched from the cursor. When the row was fetched, we cannot see the changes made to rows through the cursor.

The Forward-Only cursors are three categorize into three types:

1 Forward_Only Keyset

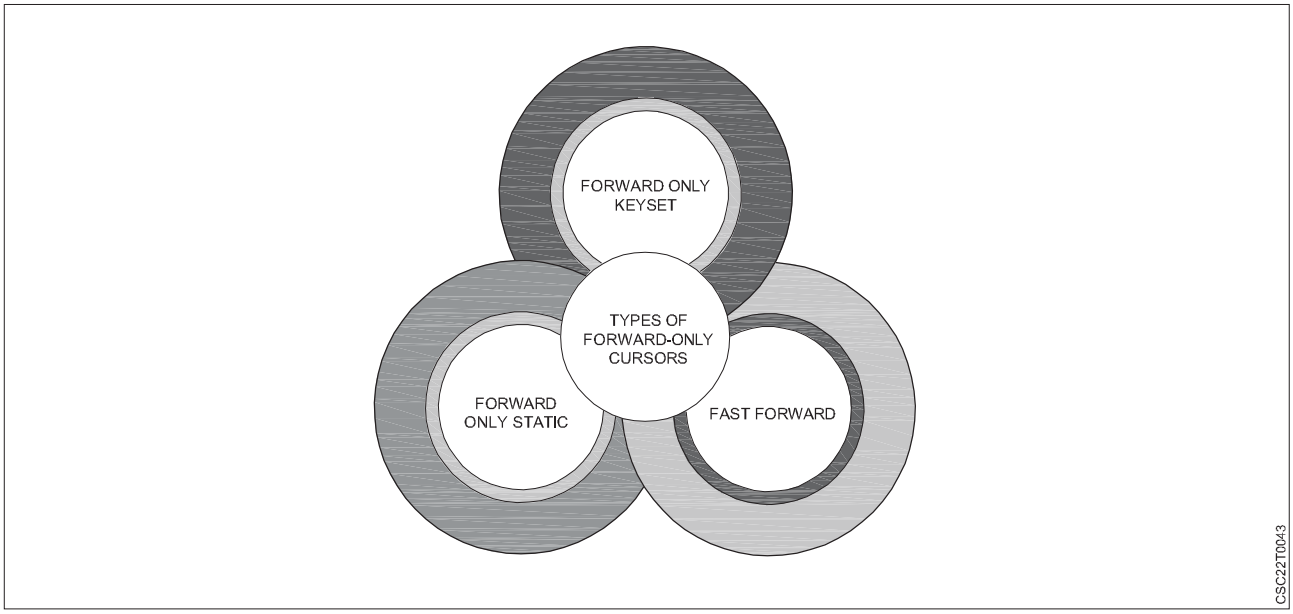
2 Forward_Only Static

3 Fast_Forward

Keyset Driven Cursors

This cursor functionality lies between a static and a dynamic cursor regarding its ability to detect changes. It can't always detect changes in the result set's membership and order like a static cursor. It can detect changes in the result set's rows values as like a dynamic cursor. It can only move from the first to last and last to the first row. The order and the membership are fixed whenever this cursor is opened.

It is operated by a set of unique identifiers the same as the keys in the keyset. The keyset is determined by all rows that qualified the SELECT statement when the cursor was first opened. It can also detect any changes to the data source, which supports update and delete operations. It is scrollable by default.



Using Cursor

Implementation of Example

Let us implement the cursor example in the SQL server. We can do this by first creating a table named “customer” using the below statement:

```
1  CREATE TABLE customer (
2  id int PRIMARY KEY,
3  c_name nvarchar(45) NOT NULL,
4  email nvarchar(45) NOT NULL,
5  city nvarchar(25) NOT NULL
6  );
```

Next, we will insert values into the table. We can execute the below statement to add data into a table:

```
1  INSERT INTO customer (id, c_name, email, city)
2  VALUES (1,’Steffen’, ‘stephen@javatpoint.com’, ‘Texas’),
3  (2, ‘Joseph’, ‘Joseph@javatpoint.com’, ‘Alaska’),
4  (3, ‘Peter’, ‘Peter@javatpoint.com’, ‘California’),
5  (4,’Donald’, ‘donald@javatpoint.com’, ‘New York’),
6  (5, ‘Kevin’, ‘kevin@javatpoint.com’, ‘Florida’),
7  (6, ‘Marielia’, ‘Marielia@javatpoint.com’, ‘Arizona’),
8  (7,’Antonio’, ‘Antonio@javatpoint.com’, ‘New York’),
9  (8, ‘Diego’, ‘Diego@javatpoint.com’, ‘California’);
```

We can verify the data by executing the SELECT statement:

```
1  SELECT * FROM customer;
```

After executing the query, we can see the below output where we have eight rows into the table:

Now, we will create a cursor to display the customer records. The below code snippets explain the all steps of the cursor declaration or creation by putting everything together:

```
1  --Declare the variables for holding data.
2  DECLARE @id INT, @c_name NVARCHAR(50), @city NVARCHAR(50)
3
```

id	c_name	email	city
1	Steffen	stephen@javatpoint.com	Texas
2	Joseph	Joseph@javatpoint.com	Alaska
3	Peter	Peter@javatpoint.com	California
4	Donald	donald@javatpoint.com	New York
5	Kevin	kevin@javatpoint.com	Florida
6	Marielia	Marielia@javatpoint.com	Arizona
7	Antonio	Antonio@javatpoint.com	New York
8	Diego	Diego@javatpoint.com	California

```
--Declare and set counter.
DECLARE @Counter INT
SET @Counter = 1

--Declare a cursor
DECLARE PrintCustomers CURSOR
FOR
SELECT id, c_name, city FROM customer

--Open cursor
OPEN PrintCustomers

--Fetch the record into the variables.
FETCH NEXT FROM PrintCustomers INTO
@id, @c_name, @city

--LOOP UNTIL RECORDS ARE AVAILABLE.
WHILE @@FETCH_STATUS = 0
BEGIN
IF @Counter = 1
BEGIN
PRINT 'id' + CHAR(9) + 'c_name' + CHAR(9) + CHAR(9) + 'city'
PRINT '-------------------------'
END

--Print the current record
PRINT CAST(@id AS NVARCHAR(10)) + CHAR(9) + @c_name + CHAR(9) + CHAR(9) + @city

--Increment the counter variable
SET @Counter = @Counter + 1
```

```
35 --Fetch the next record into the variables.
36 FETCH NEXT FROM PrintCustomers INTO
37 @id, @c_name, @city
38 END
39
40 --Close the cursor
41 CLOSE PrintCustomers
42
43 --Deallocate the cursor
44 DEALLOCATE PrintCustomers
```

After executing a cursor, we will get the below output:

Limitations of SQL Server Cursor

A cursor has some limitations so that it should always use only when there is no option except the cursor. These limitations are:

- Cursor consumes network resources by requiring a network roundtrip each time it fetches a record.
- A cursor is a memory resident set of pointers, which means it takes some memory that other processes could use on our machine.

Messages

id	c_name	city
1	Steffen	Texas
2	Joseph	Alaska
3	Peter	California
4	Donald	New York
5	Kevin	Florida
6	Marielia	Arizona
7	Antonio	New York
8	Diego	California

- It imposes locks on a portion of the table or the entire table when processing data.
- The cursor's performance and speed are slower because they update table records one row at a time.
- Cursors are quicker than while loops, but they do have more overhead.
- The number of rows and columns brought into the cursor is another aspect that affects cursor speed. It refers to how much time it takes to open your cursor and execute a fetch statement.

How can we avoid cursors?

The main job of cursors is to traverse the table row by row. The easiest way to avoid cursors are given below:

Using the SQL while loop

The easiest way to avoid the use of a cursor is by using a while loop that allows the inserting of a result set into the temporary table.

User-defined functions

Sometimes cursors are used to calculate the resultant row set. We can accomplish this by using a user-defined function that meets the requirements.

MODULE 3 : Introduction to JavaScript

LESSON 37 - 46 : Introduction to JavaScript

Objectives

At the end of this lesson, you will be able to:

- write clean & maintainable code using methods in javascript
- create & develop web pages using javascript
- create interactive features without slowing down the web page.

What is JavaScript ?

JavaScript is a lightweight, cross-platform, single-threaded, and interpreted compiled programming language. It is also known as the scripting language for webpages. It is well-known for the development of web pages, and many non-browser environments also use it.

JavaScript is a weakly typed language (dynamically typed). JavaScript can be used for Client-side developments as well as Server-side developments. JavaScript is both an imperative and declarative type of language. JavaScript contains a standard library of objects, like Array, Date, and Math, and a core set of language elements like operators, control structures, and statements

- **Client-side:** It supplies objects to control a browser and its Document Object Model (DOM). Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation. Useful libraries for the client side are AngularJS, ReactJS, VueJS, and so many others.
- **Server-side:** It supplies objects relevant to running JavaScript on a server. For if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is node.js.
- **Imperative language** – In this type of language we are mostly concerned about how it is to be done. It simply controls the flow of computation. The procedural programming approach, object, oriented approach comes under this as async await we are thinking about what is to be done further after the async call.

- **Declarative programming** – In this type of language we are concerned about how it is to be done, basically here logical computation requires. Her main goal is to describe the desired result without direct dictation on how to get it as the arrow function does.

How to Link JavaScript File in HTML ?

JavaScript can be added to HTML file in two ways:

- **Internal JS:** We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.
- **External JS:** We can write JavaScript code in another files having an extension.js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

Syntax:

```
<script>
   // JavaScript Code
</script>
```

Example:

- HTML

```
<!DOCTYPE html>
<html lang=”en”>
<head>
   <title>
      Basic Example to Describe JavaScript
   </title>
</head>
 <body>
   <!-- JavaScript code can be embedded inside
      head section or body section -->
   <script>
      console.log(“Welcome to NSTI”);
   </script>
</body>
</html>
```

Output: The output will display on the console.

Welcome to NSTI

History of JavaScript

It was created in 1995 by Brendan Eich while he was an engineer at Netscape. It was originally going to be named LiveScript but was renamed. Unlike most programming languages, JavaScript language has no concept of input or output. It is designed to run as a scripting language in a host environment, and it is up to the host environment to provide mechanisms for communicating with the outside world. The most common host environment is the browser.

Features of JavaScript

According to a recent survey conducted by Stack Overflow, JavaScript is the most popular language on earth.

With advances in browser technology and JavaScript having moved into the server with Node.js and other frameworks, JavaScript is capable of so much more. Here are a few things that we can do with JavaScript:

- JavaScript was created in the first place for DOM manipulation. Earlier websites were mostly static, after JS was created dynamic Web sites were made.
- Functions in JS are objects. They may have properties and methods just like other objects. They can be passed as arguments in other functions.
- Can handle date and time.
- Performs Form Validation although the forms are created using HTML.
- No compiler is needed.

Applications of JavaScript

- **Web Development:** Adding interactivity and behavior to static sites JavaScript was invented to do this in 1995. By using AngularJS that can be achieved so easily.
- **Web Applications:** With technology, browsers have improved to the extent that a language was required to create robust web applications. When we explore a map in Google Maps then we only need to click and drag the mouse. All detailed view is just a click away, and this is possible only because of JavaScript. It uses Application Programming Interfaces(APIs) that provide extra power to the code. The Electron and React are helpful in this department.
- **Server Applications:** With the help of Node.js, JavaScript made its way from client to server and Node.js is the most powerful on the server side.
- **Games:** Not only in websites, but JavaScript also helps in creating games for leisure. The combination of JavaScript and HTML 5 makes JavaScript popular in game development as well. It provides the EaseJS library which provides solutions for working with rich graphics.
- **Smartwatches:** JavaScript is being used in all possible devices and applications. It provides a library PebbleJS which is used in smartwatch applications. This framework works for applications that require the Internet for their functioning.
- **Art:** Artists and designers can create whatever they want using JavaScript to draw on HTML 5 canvas, and make the sound more effective also can be used p5.js library.
- **Machine Learning:** This JavaScript ml5.js library can be used in web development by using machine learning.
- **Mobile Applications:** JavaScript can also be used to build an application for non-web contexts. The features and uses of JavaScript make it a powerful tool for creating mobile applications. This is a Framework for building web and mobile apps using JavaScript. Using React Native, we can build mobile applications for different operating systems. We do not require to write code for different systems. Write once use it anywhere!

Limitations of JavaScript

- **Security risks:** JavaScript can be used to fetch data using AJAX or by manipulating tags that load data such as <img>, <object>, <script>. These attacks are called cross-site script attacks. They inject JS that is not part of the site into the visitor's browser thus fetching the details.
- **Performance:** JavaScript does not provide the same level of performance as offered by many traditional languages as a complex program written in JavaScript would be comparatively slow. But as JavaScript is used to perform simple tasks in a browser, so performance is not considered a big restriction in its use.
- **Complexity:** To master a scripting language, programmers must have a thorough knowledge of all the programming concepts, core language objects, and client and server-side objects otherwise it would be difficult for them to write advanced scripts using JavaScript.
- **Weak error handling and type checking facilities:** It is a weakly typed language as there is no need to specify the data type of the variable. So wrong type checking is not performed by compile.

Why JavaScript is known as a lightweight programming language ?

JavaScript is considered lightweight due to the fact that it has low CPU usage, is easy to implement, and has a minimalist syntax. Minimalist syntax as in, has no data types. Everything is treated here as an object. It is very easy to learn because of its syntax similar to C++ and Java.

A lightweight language does not consume much of your CPU's resources. It doesn't put excess strain on your CPU or RAM. JavaScript runs in the browser even though it has complex paradigms and logic which means it

uses fewer resources than other languages. For example, NodeJs, a variation of JavaScript not only performs faster computations but also uses fewer resources than its counterparts such as Dart or Java.

Additionally, when compared with other programming languages, it has fewer in-built libraries or frameworks, contributing as another reason for it being lightweight. However, this brings a drawback in that we need to incorporate external libraries and frameworks.

Is JavaScript Compiled or Interpreted or both ?

JavaScript is both compiled and interpreted. In the earlier versions of JavaScript, it used only the interpreter that executed code line by line and shows the result immediately. But with time the performance became an issue as interpretation is quite slow. Therefore, in the newer versions of JS, probably after the V8, the JIT compiler was also incorporated to optimize the execution and display the result more quickly. This JIT compiler generates a bytecode that is relatively easier to code. This bytecode is a set of highly optimized instructions.

The V8 engine initially uses an interpreter, to interpret the code. On further executions, the V8 engine finds patterns such as frequently executed functions, and frequently used variables, and compiles them to improve performance.

JavaScript Syntax, Variables, Operators and Expression

JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

// How to create variables:

var x;

let y;

// How to use variables:

x = 5;

y = 6;

let z = x + y;

JavaScript Values

The JavaScript syntax defines two types of values:

- Fixed values
- Variable values

Fixed values are called Literals.

Variable values are called Variables.

JavaScript Literals

The two most important syntax rules for fixed values are:

1 Numbers are written with or without decimals:

10.50

1001

2 Strings are text, written within double or single quotes:

“John Doe”

‘John Doe’

JavaScript Variables

In a programming language, variables are used to store data values.

JavaScript uses the keywords var, let and const to declare variables.

An equal sign is used to assign values to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

let x;

x = 6;

Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

In this first example, x, y, and z are undeclared variables.

They are automatically declared when first used:

Example

x = 5;

y = 6;

z = x + y;

Note

It is considered good programming practice to always declare variables before use.

From the examples you can guess:

- x stores the value 5
- y stores the value 6
- z stores the value 11

Example using var

var x = 5;

var y = 6;

var z = x + y;

Note

The var keyword was used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

The var keyword should only be used in code written for older browsers.

Example using let

let x = 5;

let y = 6;

let z = x + y;

Example using const

const x = 5;

const y = 6;

const z = x + y;

Mixed Example

const price1 = 5;

const price2 = 6;

let total = price1 + price2;

The two variables price1 and price2 are declared with the const keyword.

These are constant values and cannot be changed.

The variable total is declared with the let keyword.

The value total can be changed.

When to Use var, let, or const?

1 Always declare variables

2 Always use const if the value should not be changed

3 Always use const if the type should not be changed (Arrays and Objects)

4 Only use let if you can't use const

5 Only use var if you MUST support old browsers.

Just Like Algebra

Just like in algebra, variables hold values:

let x = 5;

let y = 6;

Just like in algebra, variables are used in expressions:

let z = x + y;

From the example above, you can guess that the total is calculated to be 11.

Note

Variables are containers for storing values.

JavaScript Identifiers

All JavaScript variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, total Volume).

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter.
- Names can also begin with $ and _ (but we will not use it in this tutorial).
- Names are case sensitive (y and Y are different variables).
- Reserved words (like JavaScript keywords) cannot be used as names.

Note

JavaScript identifiers are case-sensitive.

JavaScript Data Types

JavaScript variables can hold numbers like 100 and text values like "John Doe".

In programming, text values are called text strings.

JavaScript can handle many types of data, but for now, just think of numbers and strings.

Strings are written inside double or single quotes. Numbers are written without quotes.

If you put a number in quotes, it will be treated as a text string.

Example

```
const pi = 3.14;
let person = “John Doe”;
let answer = ‘Yes I am!’;
```

Declaring a JavaScript Variable

Creating a variable in JavaScript is called “declaring” a variable.

You declare a JavaScript variable with the var or the let keyword:

```
var carName;
```

or:

```
let carName;
```

After the declaration, the variable has no value (technically it is undefined).

To assign a value to the variable, use the equal sign:

```
carName = “Volvo”;
```

You can also assign a value to the variable when you declare it:

```
let carName = “Volvo”;
```

In the example below, we create a variable called carName and assign the value “Volvo” to it.

Then we “output” the value inside an HTML paragraph with id=”demo”:

Example

```
<p id=”demo”></p>
<script>
let carName = “Volvo”;
document.getElementById(“demo”).innerHTML = carName;
</script>
```

Note

It’s a good programming practice to declare all variables at the beginning of a script.

One Statement, Many Variables

You can declare many variables in one statement.

Start the statement with let and separate the variables by comma:

Example

```
let person = “John Doe”, carName = “Volvo”, price = 200;
```

A declaration can span multiple lines:

Example

```
let person = “John Doe”,
carName = “Volvo”,
price = 200;
```

Value = undefined

In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input.

A variable declared without a value will have the value undefined.

The variable carName will have the value undefined after the execution of this statement:

Example

let carName;

Re-Declaring JavaScript Variables

If you re-declare a JavaScript variable declared with var, it will not lose its value.

The variable carName will still have the value “Volvo” after the execution of these statements:

Example

var carName = “Volvo”;

var carName;

Note

You cannot re-declare a variable declared with let or const.

This will not work:

let carName = “Volvo”;

let carName;

JavaScript Operators

Operators are the symbols between values that allow different operations like addition, substraction, multiplication, and more. JavaScript has dozens operators, so let’s focus on the ones you’re likely to see most often.

Operators is used to perform some operation on data.

There are many type of Operators:

Arithmetic Operator (+): JavaScript uses arithmetic operators (+ - * /) to compute values:

(5 + 6) * 10

Assignment Operator (%): JavaScript uses an assignment operator (=) to assign values to variables:

let x, y;

x = 5;

y = 6;

Exponentiation Operator (**): JavaScript uses an exponentiation operator (**) to compute the power values to variables:

let x, y;

x = 5;

y = 2;

d = x ** y

Modulus Operator (**): JavaScript uses an modulus operator (%) to check the remainder values to variables:

let x, y;

x = 8;

y = 2;

d = x % y

Unary Operators:

1 **Increment operators (++):** JavaScript uses an increment operators (++) to increasing the value of variable with (+ 1).

Increment Operators are two types:

i **Pre- Increment operator (++a):** It is use for increasing value before use of variable.

ii **Post- Increment operator (a++):** It is use for increasing value after use of variable.

2 **Decrement operators (--):** JavaScript uses an decrement operators (--) to decreasing the value of variable with (- 1) .

i **Pre- decrement operator (++a):** It is use for decreasing value before use of variable.

ii **Post- decrement operator (a++):** It is use for decreasing value after use of variable.

Comparison Operators: JavaScript uses an comparison operators to compare values between two variables.

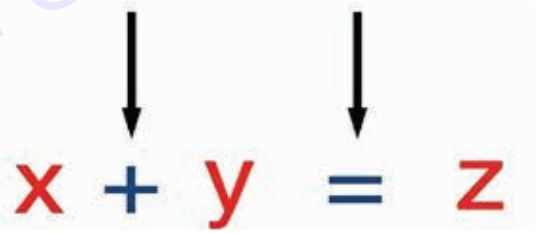
There are many type of comparison operators:

- Greater than(>)
- Greater than and Equal(>=)
- Smaller than(<)
- Smaller than and Equal(<=)
- Equal to(==)
- Equal to & Type(===)
- Not Equal to(!=)
- Not Equal to and Type(!==)

Logical Operators: JavaScript uses an Logical operators to compare and given a boolean value.

There are three types of Logical operators:

- Logical AND (&&)
- Logical OR (| |)
- Logical NOR (!)



Types of JavaScript Operators

There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Bitwise Operators
- Ternary Operators
- Type Operators

JavaScript Arithmetic Operators

Arithmetic Operators are used to perform arithmetic on numbers:

Arithmetic Operators Example

let a = 3;

let x = (100 + 50) * a;

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Note

Arithmetic operators are fully described in the JS Arithmetic chapter.

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

The Addition Assignment Operator (+=) adds a value to a variable.

Assignment

let x = 10;

x += 5;

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Note

Assignment operators are fully described in the JS Assignment chapter.

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to

<=	less than or equal to
?	ternary operator

Note

Comparison operators are fully described in the JS Comparisons chapter.

JavaScript String Comparison

All the comparison operators above can also be used on strings:

Example

let text1 = “A”;

let text2 = “B”;

let result = text1 < text2;

Note that strings are compared alphabetically:

Example

let text1 = “20”;

let text2 = “5”;

let result = text1 < text2;

JavaScript String Addition

The + can also be used to add (concatenate) strings:

Example

let text1 = “John”;

let text2 = “Doe”;

let text3 = text1 + “ “ + text2;

The += assignment operator can also be used to add (concatenate) strings:

Example

let text1 = “What a very “;

text1 += “nice day”;

The result of text1 will be:

What a very nice day

Note

When used on strings, the + operator is called the concatenation operator.

Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

Example

let x = 5 + 5;

let y = “5” + 5;

let z = “Hello” + 5;

The result of x, y, and z will be:

10

55

Hello5

Note

If you add a number and a string, the result will be a string!

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

Note

Logical operators are fully described in the JS Comparisons chapter.

JavaScript Type Operators

Operator	Description
Type of	Returns the type of a variable
Instance of	Returns true if an object is an instance of an object type

Note

Type operators are fully described in the JS Type Conversion chapter.

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

The examples above uses 4 bits unsigned examples. But JavaScript uses 32-bit signed numbers.

Because of this, in JavaScript, ~ 5 will not return 10. It will return -6.

~00000000000000000000000000000101 will return 11111111111111111111111111111010

Bitwise operators are fully described in the JS Bitwise chapter.

JavaScript's expression is a valid set of literals, variables, operators, and expressions that evaluate a single value that is an expression. This single value can be a number, a string, or a logical value depending on the expression.

Example:

- Javascript

```
// Illustration of function* expression
// use of function* keyword
function* func() {
    yield 1;
    yield 2;
    yield 3;
    yield " - Nsti";
}
let obj = '';
// Function calling
for (const i of func()) {
    obj = obj + i;
}
// Output
console.log(obj);
```

Output

123 – Nsti

JavaScript Expressions

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 * 10 evaluates to 50:

5 * 10

Expressions can also contain variable values:

x * 10

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

"John" + " " + "Doe"

JavaScript Keywords

JavaScript keywords are used to identify actions to be performed.

The let keyword tells the browser to create variables:

let x, y;

x = 5 + 6;

y = x * 10;

The var keyword also tells the browser to create variables:

var x, y;

x = 5 + 6;

y = x * 10;

In these examples, using var or let will produce the same result.

You will learn more about var and let later in this tutorial.

JavaScript Comments

Not all JavaScript statements are “executed”.

Code after double slashes // or between /* and */ is treated as a comment.

Comments are ignored, and will not be executed:

let x = 5; // I will be executed

// x = 6; I will NOT be executed

You will learn more about comments in a later chapter.

JavaScript Identifiers / Names

Identifiers are JavaScript names.

Identifiers are used to name variables and keywords, and functions.

The rules for legal names are the same in most programming languages.

A JavaScript name must begin with:

- A letter (A-Z or a-z)
- A dollar sign ($)
- Or an underscore (_)

Subsequent characters may be letters, digits, underscores, or dollar signs.

Note

Numbers are not allowed as the first character in names.

This way JavaScript can easily distinguish identifiers from numbers.

JavaScript is Case Sensitive

All JavaScript identifiers are case sensitive.

The variables lastName and lastname, are two different variables:

let lastname, lastName;

lastName = “Doe”;

lastname = “Peterson”;

JavaScript does not interpret LET or Let as the keyword let.

JavaScript and Camel Case

Historically, programmers have used different ways of joining multiple words into one variable name:

Hyphens:

first-name, last-name, master-card, inter-city.

Hyphens are not allowed in JavaScript. They are reserved for subtractions.

Underscore:

first_name, last_name, master_card, inter_city.

Upper Camel Case (Pascal Case):

FirstName, LastName, MasterCard, InterCity.

Lower Camel Case:

JavaScript programmers tend to use camel case that starts with a lowercase letter:

firstName, lastName, masterCard, interCity.

JavaScript Character Set

JavaScript uses the Unicode character set.

Unicode covers (almost) all the characters, punctuations, and symbols in the world.

Control Flow

JavaScript control statement is used to control the execution of a program based on a specific condition. If the condition meets then a particular block of action will be executed otherwise it will execute another block of action that satisfies that particular condition.

Types of Control Statements in JavaScript

- **Conditional Statement:** These statements are used for decision-making, a decision
- n is made by the conditional statement based on an expression that is passed. Either YES or NO.
- **Iterative Statement:** This is a statement that iterates repeatedly until a condition is met. Simply said, if we have an expression, the statement will keep repeating itself until and unless it is satisfied.

There are several methods that can be used to perform control statements in JavaScript:

Table of Content

- If Statement
- Using If-Else Statement
- Using Switch Statement
- Using the Ternary Operator (Conditional Operator)
- Using For loop

Approach 1: If Statement

In this approach, we are using an if statement to check a specific condition, the code block gets executed when the given condition is satisfied.

Syntax:

```
if (condition_is_given_here ) {
   // If the condition is met,
   //the code  will get executed.
}
```

Example: In this example, we are using an if statement to check our given condition.

- Javascript

```
const num = 5;
if (num > 0) {
   console.log(“The number is positive.”);
};
```

Output

The number is positive.

Approach 2: Using If-Else Statement

The if-else statement will perform some action for a specific condition. If the condition meets then a particular code of action will be executed otherwise it will execute another code of action that satisfies that particular condition.

Syntax:

```
if (condition1) {
   // Executes when condition1 is true
   if (condition2) {
      // Executes when condition2 is true
   }
}
```

Example: In this example, we are using the if..else statement to verify whether the given number is positive or negative.

- Javascript

```
let num = -10;
if (num > 0)
    console.log("The number is positive.");
else
    console.log("The number is negative");
```

Output

The number is negative

Approach 3: Using Switch Statement

The switch case statement in JavaScript is also used for decision-making purposes. In some cases, using the switch case statement is seen to be more convenient than if-else statements.

Syntax:

```
switch (expression) {
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
    .
    .
    case valueN:
        statementN;
        break;
    default:
        statementDefault;
}
```

Example: In this example, we are using the above-explained approach.

- Javascript

```
let num = 5;
switch (num) {
    case 0:
        console.log("Number is zero.");
        break;
    case 1:
        console.log("Nuber is one.");
        break;
```

```
    case 2:
        console.log("Number is two.");
        break;
    default:
        console.log("Number is greater than 2.");
};
```

Output

Number is greater than 2.

Approach 4: Using the Ternary Operator (Conditional Operator)

The conditional operator, also referred to as the ternary operator (?:), is a shortcut for expressing conditional statements in JavaScript.

Syntax:

condition ? value if true : value if false

Example: In this example, we are using the ternary operator to find whether the given number is positive or negative.

- Javascript

```
let num = 10;
let result = num >= 0 ? "Positive" : "Negative";
console.log(`The number is ${result}.`);
```

Output

The number is Positive.

Approach 5: Using For loop

In this approach, we are using for loop in which the execution of a set of instructions repeatedly until some condition evaluates and becomes false

Syntax:

```
for (statement 1; statement 2; statement 3) {
    // Code here . . .
}
```

Example: In this example, we are using Iterative Statement as a for loop, in which we find the even number between 0 to 10.

- Javascript

```
for (let i = 0; i <= 10; i++) {
  if (i % 2 === 0) {
    console.log(i);
  }
};
```

Output:

0

2

4

6

8

10

JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when “something” invokes it (calls it).

Example

```
// Function to compute the product of p1 and p2
function myFunction(p1, p2) {
  return p1 * p2;
}
```

JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:

(parameter1, parameter2, ...)

The code to be executed, by the function, is placed inside curly brackets: {}

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Function parameters are listed inside the parentheses () in the function definition.

Function arguments are the values received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

Function Invocation

The code inside the function will execute when “something” invokes (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

You will learn a lot more about function invocation later in this tutorial.

Function Return

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will “return” to execute the code after the invoking statement.

Functions often compute a return value. The return value is “returned” back to the “caller”:

Example

Calculate the product of two numbers, and return the result:

```
// Function is called, the return value will end up in x
let x = myFunction(4, 3);
function myFunction(a, b) {
// Function returns the product of a and b
  return a * b;
}
```

Why Functions?

With functions you can reuse code

You can write code that can be used many times.

You can use the same code with different arguments, to produce different results.

The () Operator

The () operator invokes (calls) the function:

Example

Convert Fahrenheit to Celsius:

```
function to Celsius (fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
let value = to Celsius(77);
```

Accessing a function with incorrect parameters can return an incorrect answer:

Example

```
function to Celsius (fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
let value = to Celsius();
```

Accessing a function without () returns the function and not the function result:

Example

```
function to Celsius (fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
let value = to Celsius;
```

Note

As you see from the examples above, toCelsius refers to the function object, and toCelsius() refers to the function result.

Functions Used as Variable Values

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

Example

Instead of using a variable to store the return value of a function:

```
let x = to Celsius(77);
let text = “The temperature is “ + x + “ Celsius”;
```

You can use the function directly, as a variable value:

```
let text = “The temperature is “ + to Celsius(77) + “ Celsius”;
```

Local Variables

Variables declared within a JavaScript function, become LOCAL to the function.

Local variables can only be accessed from within the function.

Example

```
// code here can NOT use carName
```

```
function myFunction() {
  let carName = “Volvo”;
  // code here CAN use carName
}
// code here can NOT use carName
```

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.

Test Yourself With Exercises

Exercise:

Execute the function named myFunction.

```
function myFunction() {
  alert(“Hello World!”);
}
```

JavaScript function is a set of statements that take inputs, do some specific computation, and produce output.

A JavaScript function is executed when “something” invokes it (calls it).

Example 1: A basic javascript function, here we create a function that divides the 1st element by the second element.

- Javascript

```
function myFunction(g1, g2) {
    return g1 / g2;
}
const value = myFunction(8, 2); // Calling the function
console.log(value);
```

Output:

0 seconds of 17 secondsVolume 0%

4

You must already have seen some commonly used functions in JavaScript like alert(), which is a built-in function in JavaScript. But JavaScript allows us to create user-defined functions also. We can create functions in JavaScript using the keyword `function`.

Syntax:

The basic syntax to create a function in JavaScript is shown below.

```
functionName(Parameter1, Parameter2, ...)
{
    // Function body
}
```

To create a function in JavaScript, we have to first use the keyword function, separated by the name of the function and parameters within parenthesis. The part of the function inside the curly braces {} is the body of the function.

In javascript, functions can be used in the same way as variables for assignments, or calculations.

Function Invocation:

- Triggered by an event (e.g., a button click by a user).

- When explicitly called from JavaScript code.
- Automatically executed, such as in self-invoking functions.

Function Definition:

Before, using a user-defined function in JavaScript we have to create one. We can use the above syntax to create a function in JavaScript. A function definition is sometimes also termed a function declaration or function statement. Below are the rules for creating a function in JavaScript:

- Every function should begin with the keyword function followed by,
- A user-defined function name that should be unique,
- A list of parameters enclosed within parentheses and separated by commas,
- A list of statements composing the body of the function enclosed within curly braces {}.

Example 2: This example shows a basic declaration of a function in javascript.

- JavaScript

```
function calcAddition(number1, number2) {
   return number1 + number2;
}
console.log(calcAddition(6,9));
```

Output

15

In the above example, we have created a function named calcAddition,

- This function accepts two numbers as parameters and returns the addition of these two numbers.
- Accessing the function with just the function name without () will return the function object instead of the function result.

There are three ways of writing a function in JavaScript:

Function Declaration:

It declares a function with a function keyword. The function declaration must have a function name.

Syntax:

```
function NSTI(paramA, paramB) {
   // Set of statements
}
```

Function Expression:

It is similar to a function declaration without the function name. Function expressions can be stored in a variable assignment.

Syntax:

```
let NSTI= function(paramA, paramB) {
   // Set of statements
}
```

Example 3: This example explains the usage of the Function expression.

- Javascript

```
const square = function (number) {
     return number * number;
};
```

```
const x = square(4); // x gets the value 16
console.log(x);
```

Output

16

Functions as Variable Values:

Functions can be used the same way as you use variables.

Example:

```
// Function to convert Fahrenheit to Celsius
function to Celsius (fahrenheit) {
    return (fahrenheit - 32) * 5/9;
  }
  // Using the function to convert temperature
  let temperatureInFahrenheit = 77;
  let temperatureInCelsius = toCelsius(temperatureInFahrenheit);
  let text = “The temperature is “ + temperatureInCelsius + “ Celsius”;
```

Arrow Function:

It is one of the most used and efficient methods to create a function in JavaScript because of its comparatively easy implementation. It is a simplified as well as a more compact version of a regular or normal function expression or syntax.

Syntax:

```
let function_name = (argument1, argument2 ,..) => expression
```

Example 4: This example describes the usage of the Arrow function.

- Javascript

```
const a = [“Hydrogen”, “Helium”, “Lithium”, “Beryllium”];
const a2 = a.map(function (s) {
    return s.length;
});
console.log(“Normal way “, a2); // [8, 6, 7, 9]
const a3 = a.map((s) => s.length);
console.log(“Using Arrow Function “, a3); // [8, 6, 7, 9]
```

Output

Normal way [8, 6, 7, 9]

Using Arrow Function [8, 6, 7, 9]

Function Parameters:

Till now, we have heard a lot about function parameters but haven’t discussed them in detail. Parameters are additional information passed to a function. For example, in the above example, the task of the function calcAddition is to calculate the addition of two numbers. These two numbers on which we want to perform the addition operation are passed to this function as parameters. The parameters are passed to the function within parentheses after the function name and separated by commas. A function in JavaScript can have any number of parameters and also at the same time, a function in JavaScript can not have a single parameter.

Example 4: In this example, we pass the argument to the function.

- Javascript

```
function multiply(a, b) {
    b = type of b !== “undefined” ? b : 1;
    return a * b;
}
console.log(multiply(69)); // 69
```

Output

69

Calling Functions:

After defining a function, the next step is to call them to make use of the function. We can call a function by using the function name separated by the value of parameters enclosed between the parenthesis and a semicolon at the end. The below syntax shows how to call functions in JavaScript:

Syntax:

functionName(Value1, Value2, ..);

Example 5: Below is a sample program that illustrates the working of functions in JavaScript:

- JavaScript

```
function welcomeMsg(name) {
    return (“Hello “ + name + “ welcome to NSTI”);
}
// creating a variable
let nameVal = “Admin”;
// calling the function
console.log(welcomeMsg(nameVal));
```

Output:

Hello Admin welcome to NSTI

Return Statement:

There are some situations when we want to return some values from a function after performing some operations. In such cases, we can make use of the return statement in JavaScript. This is an optional statement and most of the time the last statement in a JavaScript function. Look at our first example with the function named as calcAddition. This function is calculating two numbers and then returns the result.

Syntax: The most basic syntax for using the return statement is:

return value;

The return statement begins with the keyword return separated by the value which we want to return from it. We can use an expression also instead of directly returning the value.

Functions:

- JavaScript | Arrow functions
- JavaScript | escape()
- JavaScript | unescape()
- JavaScript | Window print()
- Javascript | Window Blur() and Window Focus() Method
- JavaScript | console.log()
- JavaScript | parseFloat()

- JavaScript | uneval()
- JavaScript | parseInt()
- JavaScript | match()
- JavaScript | Date.parse()
- JavaScript | Replace() Method
- JavaScript | Map.get( )
- JavaScript | Map.entries( )
- JavaScript | Map.clear( )
- JavaScript | Map.delete()
- JavaScript | Map.has( )

Concept of Object oriented Development

In JavaScript, object-oriented development is based on a prototype-based model rather than a class-based model, as seen in languages like Java. Here are the key concepts of object-oriented development in JavaScript:

1 Objects and Properties:

- In JavaScript, objects are collections of key-value pairs where keys are strings (or Symbols) and values can be of any data type.
- Properties of an object can hold data or functions (methods).

```
var person = {
firstName: “John”,
lastName: “Doe”,
getFullName: function() {
return this.firstName + “ “ + this.lastName;
}
};
```

2 Prototypes and Inheritance:

- Every object in JavaScript has a prototype, which is another object. If a property or method is not found on the object itself, JavaScript looks for it in the object’s prototype, forming a prototype chain.
- Objects can inherit properties and methods from their prototypes.

```
var student = Object.create(person); // ‘student’ inherits from ‘person’
student.school = “XYZ School”;
```

3 Constructor Functions:

- While JavaScript doesn’t have classes in the traditional sense, constructor functions can be used to create objects with shared properties and methods.
- The new keyword is used to instantiate objects from constructor functions.

```
function Person(firstName, lastName) {
this.firstName = firstName;
this.lastName = lastName;
}
Person.prototype.getFullName = function() {
```

```
return this.firstName + " " + this.lastName;
};
var person1 = new Person("Alice", "Smith");
```

4 Encapsulation:

- Encapsulation can be achieved through closures and private variables within functions.

```
function Counter() {
var count = 0;
this.increment = function() {
count++;
};
this.getCount = function() {
return count;
};
}
var counter = new Counter();
counter.increment();
console.log(counter.getCount()); // Outputs: 1
```

5 Polymorphism:

- JavaScript supports polymorphism through dynamic typing, allowing objects to be used in multiple contexts without explicit type definitions.

```
function displayInfo(obj) {
console.log(obj.getDetails());
}
var student = {
name: "Bob",
getDetails: function() {
return "Student: " + this.name;
}
};
displayInfo(student); // Outputs: Student: Bob
```

In summary, JavaScript's object-oriented development is centered around objects, prototypes, and constructor functions. It offers a flexible and dynamic approach to building and extending objects, making it well-suited for web development where dynamic and responsive behaviors are essential.

Document Object Model

The document object represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the root element that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

As mentioned earlier, it is the object of window. So

window.document

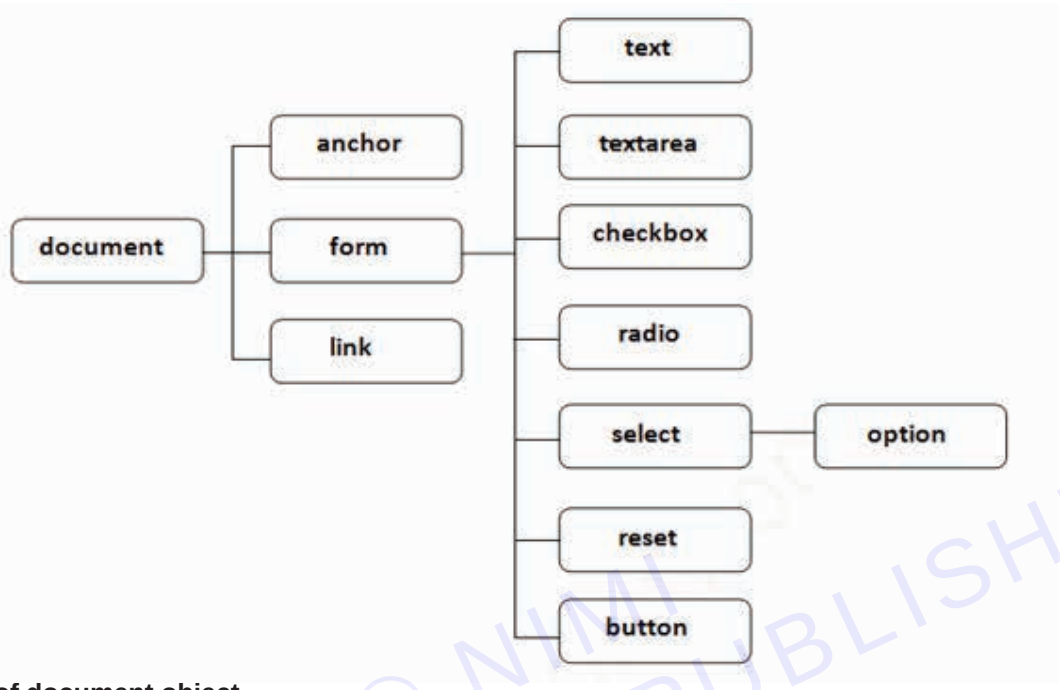
Is same as

Document

Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.



Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the document.
writeln("string")	writes the given string on the document with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

Accessing field value by document object

In this example, we are going to get the value of input text by user. Here, we are using document.form1.name.value to get the value of name field.

Here, document is the root element that represents the html document.

form1 is the name of the form.

name is the attribute name of the input text.

value is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>
<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

the document.getElementById() method returns the element of specified id.In the previous page, we have used document.form1.name.value to get the value of the input value. Instead of this, we can use document. getElementById() method to get value of the input text. But we need to define id for the input field. Let's see the simple example of document.getElementById() method that prints cube of the given number.

```
<script type="text/javascript">
function getcube(){
var number=document.getElementById("number").value;
alert(number*number*number);
}
</script>
<form>
Enter No:<input type="text" id="number" name="number"/><br/>
<input type="button" value="cube" onclick="getcube()"/>
</form>
```

Example of document.getElementsByName() method

In this example, we going to count total number of genders. Here, we are using getElementsByName() method to get all the genders.

```
<script type="text/javascript">
function totalelements()
{
var allgenders=document.getElementsByName("gender");
alert("Total Genders:"+allgenders.length);
}
</script>
<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">
<input type="button" onclick="totalelements()" value="Total Genders">
</form>
```

Example of document.getElementsByTagName() method

In this example, we going to count total number of paragraphs used in the document. To do this, we have called the document.getElementsByTagName("p") method that returns the total paragraphs.

```
<script type="text/javascript">
function countpara(){
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);
 }
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs by getElementByTagName() method.</p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
```

In this example, we are dynamically writing the html form inside the div name having the id mylocation. We are identifing this position by calling the document.getElementById() method.

```
<script type="text/javascript" >
function showcommentform() {
var data="Name:<input type='text' name='name'><br>Comment:<br><textarea rows='5' cols='80'></textarea>
<br><input type='submit' value='Post Comment'>";
document.getElementById('mylocation').innerHTML=data;
}
</script>
<form name="myForm">
<input type="button" value="comment" onclick="showcommentform()">
<div id="mylocation"></div>
</form>
```

JavaScript Form

Summary: in this tutorial, you will learn about JavaScript form API: accessing the form, getting values of the elements, validating form data, and submitting the form.

Form basics

To create a form in HTML, you use the <form> element:

<form action="/signup" method="post" id="signup">

</form>Code language: HTML, XML (xml)

The <form> element has two important attributes: action and method.

- The action attribute specifies a URL that will process the form submission. In this example, the action is the /signup URL.
- The method attribute specifies the HTTP method to submit the form with. Usually, the method is either post or get.

  Generally, you use the get method when you want to retrieve data from the server and the post method when you want to change data on the server.

  JavaScript uses the HTMLFormElement object to represent a form. The HTMLFormElement has the following properties that correspond to the HTML attributes:

- action – is the URL that processes the form data. It is equivalent to the action attribute of the <form> element.
- method – is the HTTP method which is equivalent to the method attribute of the <form> element.

The HTMLFormElement also provides the following useful methods:

- submit() – submit the form.
- reset() – reset the form.

Referencing forms

To reference the <form> element, you can use DOM selecting methods such as getElementById():

const form = document.getElementById('subscribe');Code language: JavaScript (javascript)

An HTML document can have multiple forms. The document.forms property returns a collection of forms (HTMLFormControlsCollection) on the document:

document.formsCode language: JavaScript (javascript)

To reference a form, you use an index. For example, the following statement returns the first form of the HTML document:

document.forms[0]Code language: CSS (css)

Submitting a form

Typically, a form has a submit button. When you click it, the browser sends the form data to the server. To create a submit button, you use <input> or <button> element with the type submit:

<input type="submit" value="Subscribe">Code language: HTML, XML (xml)

Or

<button type="submit">Subscribe</button>Code language: HTML, XML (xml)

If the submit button has focus and you press the Enter key, the browser also submits the form data.

When you submit the form, the submit event is fired before the request is sent to the server. This gives you a chance to validate the form data. If the form data is invalid, you can stop submitting the form.

To attach an event listener to the submit event, you use the addEventListener() method of the form element as follows:

```
const form  = document.getElementById('signup');

form.addEventListener('submit', (event) => {
    // handle the form data
});
```
Code language: JavaScript (javascript)

To stop the form from being submitted, you call the preventDefault() method of the event object inside the submit event handler like this:

```
form.addEventListener('submit', (event) => {
    // stop form submission
    event.preventDefault();
});
```
Code language: PHP (php)

Typically, you call the event.preventDefault() method if the form data is invalid. To submit the form in JavaScript, you call the submit() method of the form object:

form.submit();Code language: CSS (css)

Note that the form.submit() does not fire the submit event. Therefore, you should always validate the form before calling this method.

Accessing form fields

To access form fields, you can use DOM methods like getElementsByName(), getElementById(), querySelector(), etc.

Also, you can use the elements property of the form object. The form.elements property stores a collection of the form elements.

JavaScript allows you to access an element by index, id, or name. Suppose that you have the following signup form with two <input> elements:

```
<form action="signup.html" method="post" id="signup">
    <h1>Sign Up</h1>
    <div class="field">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" placeholder="Enter your fullname" />
        <small></small>
    </div>
    <div class="field">
        <label for="email">Email:</label>
        <input type="text" id="email" name="email" placeholder="Enter your email address" />
        <small></small>
    </div>
    <button type="submit">Subscribe</button>
</form>
```

Code language: HTML, XML (xml)

To access the elements of the form, you get the form element first:

```
const form = document.getElementById('signup');
```

Code language: JavaScript (javascript)

And use index, id, or name to access the element. The following accesses the first form element:

```
form.elements[0]; // by index
form.elements['name']; // by name
form.elements['name']; // by id (name & id are the same in this case)
```

Code language: JavaScript (javascript)

The following accesses the second input element:

```
form.elements[1]; // by index
form.elements['email']; //  by name
form.elements['email']; // by id
```

Code language: JavaScript (javascript)

After accessing a form field, you can use the value property to access its value, for example:

```
const form = document.getElementById('signup');
const name = form.elements['name'];
const email = form.elements['email'];
// getting the element's value
let fullName = name.value;
let emailAddress = email.value;
```

Code language: JavaScript (javascript)

Put it all together: signup form

The following illustrates the HTML document that has a signup form. See the live demo here.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>JavaScript Form Demo</title>
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="css/style.css" />
    </head>
    <body>
        <div class="container">
            <form action="signup.html" method="post" id="signup">
                <h1>Sign Up</h1>
                <div class="field">
                    <label for="name">Name:</label>
                    <input type="text" id="name" name="name" placeholder="Enter your fullname" />
                    <small></small>
                </div>
                <div class="field">
                    <label for="email">Email:</label>
                    <input type="text" id="email" name="email" placeholder="Enter your email address" />
                    <small></small>
                </div>
                <div class="field">
                    <button type="submit" class="full">Subscribe</button>
                </div>
            </form>
        </div>
        <script src="js/app.js"></script>
    </body>
</html>
```

Code language: HTML, XML (xml)

The HTML document references the style.css and app.js files. It uses the <small> element to display an error message in case the <input> element has invalid data.

Submitting the form without providing any information will result in the following error:

:

The following shows the complete app.js file:

```
// show a message with a type of the input
function showMessage(input, message, type) {
    // get the small element and set the message
    const msg = input.parentNode.querySelector("small");
    msg.innerText = message;
    // update the class for the input
    input.className = type ? "success" : "error";
    return type;
```

```
}
function showError(input, message) {
        return showMessage(input, message, false);
}
function showSuccess(input) {
        return showMessage(input, “”, true);
}
function hasValue(input, message) {
        if (input.value.trim() === “”) {
                return showError(input, message);
        }
        return showSuccess(input);
}
function validateEmail(input, requiredMsg, invalidMsg) {
        // check if the value is not empty
        if (!hasValue(input, requiredMsg)) {
                return false;
        }
        // validate email format
        const emailRegex =
                /^(([^<>()\[\]\\.,;:\s@”]+(\.[^<>()\[\]\\.,;:\s@”]+)*)|(“.+”))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
        const email = input.value.trim();
        if (!emailRegex.test(email)) {
                return showError(input, invalidMsg);
        }
        return true;
}
const form = document.querySelector(“#signup”);
const NAME_REQUIRED = “Please enter your name”;
const EMAIL_REQUIRED = “Please enter your email”;
const EMAIL_INVALID = “Please enter a correct email address format”;
form.addEventListener(“submit”, function (event) {
        // stop form submission
        event.preventDefault();
        // validate the form
        let nameValid = hasValue(form.elements[“name”], NAME_REQUIRED);
        let emailValid = validateEmail(form.elements[“email”], EMAIL_REQUIRED, EMAIL_INVALID);
        // if valid, submit the form.
        if (nameValid && emailValid) {
```

```
            alert("Demo only. No form was posted.");
        }
});
```

Code language: JavaScript (javascript)

How it works.

The showMessage() function

The showMessage() function accepts an input element, a message, and a type:

```
// show a message with a type of the input
function showMessage(input, message, type) {
    // get the <small> element and set the message
    const msg = input.parentNode.querySelector("small");
    msg.innerText = message;
    // update the class for the input
    input.className = type ? "success" : "error";
    return type;
}
```

Code language: JavaScript (javascript)

The following shows the name input field on the form:

```
<div class="field">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter your fullname" />
    <small></small>
</div>
```

Code language: HTML, XML (xml)

If the name's value is blank, you need to get its parent first which is the with the class "field".

```
input.parentNode
```

Code language: CSS (css)

Next, you need to select the <small> element:

```
const msg = input.parentNode.querySelector("small");
```

Code language: JavaScript (javascript)

Then, update the message:

```
msg.innerText = message;
```

After that, we change the CSS class of the input field based on the value of the type parameter. If the type is true, we change the class of the input to success. Otherwise, we change the class to error.

```
input.className = type ? "success" : "error";
```

Code language: JavaScript (javascript)

Finally, return the value of the type:

```
return type;
```

Code language: JavaScript (javascript)

The showError() and showSuccess() functions

The the showError() and showSuccess() functions call the showMessage() function. The showError() function always returns false whereas the showSuccess() function always returns true. Also, the showSuccess() function sets the error message to an empty string.

```
function showError(input, message) {
    return showMessage(input, message, false);
}
```

```
function showSuccess(input) {
    return showMessage(input, "", true);
}Code language: JavaScript (javascript)
```

The hasValue() function

The hasValue() function checks if an input element has a value or not and shows an error message using the showError() or showSuccess() function accordingly:

```
function hasValue(input, message) {
    if (input.value.trim() === "") {
        return showError(input, message);
    }
    return showSuccess(input);
}Code language: JavaScript (javascript)
```

The validateEmail() function

The validateEmail() function validates if an email field contains a valid email address:

```
function validateEmail(input, requiredMsg, invalidMsg) {
    // check if the value is not empty
    if (!hasValue(input, requiredMsg)) {
        return false;
    }
    // validate email format
    const emailRegex =
        /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
    const email = input.value.trim();
    if (!emailRegex.test(email)) {
        return showError(input, invalidMsg);
    }
    return true;
}Code language: PHP (php)
```

The validateEmail() function calls the hasValue() function to check if the field value is empty. If the input field is empty, it shows the requiredMsg.

To validate the email, it uses a regular expression. If the email is invalid, the validateEmail() function shows the invalidMsg.

The submit event handler

First, select the signup form by its id using the querySelector() method:

```
const form = document.querySelector("#signup");Code language: JavaScript (javascript)
```

Second, define some constants to store the error messages:

```
const NAME_REQUIRED = "Please enter your name";
const EMAIL_REQUIRED = "Please enter your email";
const EMAIL_INVALID = "Please enter a correct email address format";Code language: JavaScript (javascript)
```

Third, add the submit event listener of the signup form using the addEventListener() method:

```
form.addEventListener("submit", function (event) {
    // stop form submission
    event.preventDefault();
    // validate the form
    let nameValid = hasValue(form.elements["name"], NAME_REQUIRED);
    let emailValid = validateEmail(form.elements["email"], EMAIL_REQUIRED, EMAIL_INVALID);
    // if valid, submit the form.
    if (nameValid && emailValid) {
        alert("Demo only. No form was posted.");
    }
});
```

Code language: JavaScript (javascript)

In the submit event handler:

1 Stop the form submission by calling the event.preventDefault() method.

2 Validate the name and email fields using the hasValue() and validateEmail() functions.

3 If both name and email are valid, show an alert. In a real-world application, you need to call the form.submit() method to submit the form.

Summary

- Use the <form> element to create an HTML form.
- Use DOM methods such as getElementById() and querySelector() to select a <form> element. The document.forms[index] also returns the form element by a numerical index.
- Use form.elements to access form elements.
- The submit event fires when users click the submit button on the form.

Concept of Cookies

Cookies let you store user information in web pages.

What are Cookies?

Cookies are data, stored in small text files, on your computer.

When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.

Cookies were invented to solve the problem "how to remember information about the user":

- When a user visits a web page, his/her name can be stored in a cookie.
- Next time the user visits the page, the cookie "remembers" his/her name.

Cookies are saved in name-value pairs like:

username = John Doe

When a browser requests a web page from a server, cookies belonging to the page are added to the request. This way the server gets the necessary data to "remember" information about users.

None of the examples below will work if your browser has local cookies support turned off.

Create a Cookie with JavaScript

JavaScript can create, read, and delete cookies with the document.cookie property.

With JavaScript, a cookie can be created like this:

document.cookie = “username=John Doe”;

You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed:

document.cookie = “username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC”;

With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.

document.cookie = “username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/”;

Read a Cookie with JavaScript

With JavaScript, cookies can be read like this:

let x = document.cookie;

document.cookie will return all cookies in one string much like: cookie1=value; cookie2=value; cookie3=value;

Change a Cookie with JavaScript

With JavaScript, you can change a cookie the same way as you create it:

document.cookie = “username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/”;

The old cookie is overwritten.

Delete a Cookie with JavaScript

Deleting a cookie is very simple.

You don’t have to specify a cookie value when you delete a cookie.

Just set the expires parameter to a past date:

document.cookie = “username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;”;

You should define the cookie path to ensure that you delete the right cookie.

Some browsers will not let you delete a cookie if you don’t specify the path.

The Cookie String

The document.cookie property looks like a normal text string. But it is not.

Even if you write a whole cookie string to document.cookie, when you read it out again, you can only see the name-value pair of it.

If you set a new cookie, older cookies are not overwritten. The new cookie is added to document.cookie, so if you read document.cookie again you will get something like:

cookie1 = value; cookie2 = value;

Display All Cookies Create Cookie 1 Create Cookie 2 Delete Cookie 1 Delete Cookie 2

If you want to find the value of one specified cookie, you must write a JavaScript function that searches for the cookie value in the cookie string.

JavaScript Cookie Example

In the example to follow, we will create a cookie that stores the name of a visitor.

The first time a visitor arrives to the web page, he/she will be asked to fill in his/her name. The name is then stored in a cookie.

The next time the visitor arrives at the same page, he/she will get a welcome message.

For the example we will create 3 JavaScript functions:

1 A function to set a cookie value

2 A function to get a cookie value

3 A function to check a cookie value

A Function to Set a Cookie

First, we create a function that stores the name of the visitor in a cookie variable:

Example

```
function setCookie(cname, cvalue, exdays) {
  const d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  let expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```

Example explained:

The parameters of the function above are the name of the cookie (cname), the value of the cookie (cvalue), and the number of days until the cookie should expire (exdays).

The function sets a cookie by adding together the cookiename, the cookie value, and the expires string.

A Function to Get a Cookie

Then, we create a function that returns the value of a specified cookie:

Example

```
function getCookie(cname) {
  let name = cname + "=";
  let decodedCookie = decodeURIComponent(document.cookie);
  let ca = decodedCookie.split(';');
  for(let i = 0; i <ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

Function explained:

Take the cookiename as parameter (cname).

Create a variable (name) with the text to search for (cname + "=").

Decode the cookie string, to handle cookies with special characters, e.g. '$'

Split document.cookie on semicolons into an array called ca (ca = decodedCookie.split(';')).

Loop through the ca array (i = 0; i < ca.length; i++), and read out each value c = ca[i]).

If the cookie is found (c.indexOf(name) == 0), return the value of the cookie (c.substring(name.length, c.length).

If the cookie is not found, return "".

A Function to Check a Cookie

Last, we create the function that checks if a cookie is set.

If the cookie is set it will display a greeting.

If the cookie is not set, it will display a prompt box, asking for the name of the user, and stores the username cookie for 365 days, by calling the setCookie function:

Example

```
function checkCookie() {
  let username = getCookie("username");
  if (username != "") {
   alert("Welcome again " + username);
  } else {
    username = prompt("Please enter your name:", "");
    if (username != "" && username != null) {
      setCookie("username", username, 365);
    }
  }
}
```

All Together Now

Example

```
function setCookie(cname, cvalue, exdays) {
  const d = new Date();
  d.setTime(d.getTime() + (exdays * 24 * 60 * 60 * 1000));
  let expires = "expires="+d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
function getCookie(cname) {
  let name = cname + "=";
  let ca = document.cookie.split(';');
  for(let i = 0; i < ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

```
function checkCookie() {
  let user = getCookie("username");
  if (user != "") {
    alert("Welcome again " + user);
  } else {
    user = prompt("Please enter your name:", "");
    if (user != "" && user != null) {
      setCookie("username", user, 365);
    }
  }
}
```

The example above runs the checkCookie() function when the page loads

Cascaded Style Sheets

It seems there might be a slight confusion in your question. Cascading Style Sheets (CSS) and JavaScript are two distinct technologies used in web development, each with its own purpose. CSS is used for styling and layout, while JavaScript is a scripting language used for enhancing interactivity and manipulating the behavior of web pages. However, it's common for JavaScript to interact with and manipulate CSS to achieve dynamic effects on a webpage.

Here's a brief overview of how JavaScript can be used with CSS in web development:

1 DOM Manipulation

- JavaScript can manipulate the Document Object Model (DOM), which represents the structure of a webpage. Through the DOM, you can dynamically change the styles of HTML elements.

// Example: Change the color of a paragraph using JavaScript var paragraph = document. getElementById("myParagraph"); paragraph.style.color = "blue";

2 Event Handling:

- JavaScript is often used to handle events such as clicks, mouseovers, or keyboard inputs. These events can trigger changes in CSS styles.

// Example: Change the background color on button click var button = document.getElementById("myButton"); button.addEventListener("click", function() { document.body.style.backgroundColor = "yellow"; });

3 Animations and Transitions:

- JavaScript can be used to create animations and transitions by dynamically modifying CSS properties over time.

// Example: Create a simple fade-in effect var element = document.getElementById("fadeInElement"); element.style.opacity = 0; function fadeIn() { var opacity = 0; var interval = setInterval(function() { if (opacity >= 1) { clearInterval(interval); } else { opacity += 0.1; element.style.opacity = opacity; } }, 100); } fadeIn();

4 CSS Class Manipulation:

- JavaScript can add, remove, or toggle CSS classes dynamically, allowing for more organized and modular styling.

// Example: Toggle a class on button click var button = document.getElementById("myButton"); var element = document.getElementById("myElement"); button.addEventListener("click", function() { element.classList. toggle("highlight"); });

In summary, while CSS and JavaScript serve different purposes, JavaScript is commonly used to dynamically interact with and modify CSS styles to create more interactive and responsive web pages.

JavaScript Errors

Throw, and Try...Catch...Finally

The try statement defines a code block to run (to try).

The catch statement defines a code block to handle any error.

The finally statement defines a code block to run regardless of the result.

The throw statement defines a custom error.

Errors Will Happen!

When executing JavaScript code, different errors can occur.

Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.

Example

In this example we misspelled “alert” as “adddlert” to deliberately produce an error:

```
<p id=”demo”></p>
<script>
try {
  adddlert(“Welcome guest!”);
}
catch(err) {
  document.getElementById(“demo”).innerHTML = err.message;
}
</script>
```

JavaScript catches add alert as an error, and executes the catch code to handle it.

JavaScript try and catch

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

The JavaScript statements try and catch come in pairs:

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
```

JavaScript Throws Errors

When an error occurs, JavaScript will normally stop and generate an error message.

The technical term for this is: JavaScript will throw an exception (throw an error).

JavaScript will actually create an Error object with two properties: name and message.

The throw Statement

The throw statement allows you to create a custom error.

Technically you can throw an exception (throw an error).

The exception can be a JavaScript String, a Number, a Boolean or an Object:

```
throw “Too big”;    // throw a text
```

```
throw 500;          // throw a number
```

If you use throw together with try and catch, you can control program flow and generate custom error messages.

Input Validation Example

This example examines input. If the value is wrong, an exception (err) is thrown.

The exception (err) is caught by the catch statement and a custom error message is displayed:

```
<!DOCTYPE html>
<html>
<body>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="p01"></p>
<script>
function myFunction() {
  const message = document.getElementById("p01");
  message.innerHTML = "";
  let x = document.getElementById("demo").value;
  try {
    if(x.trim() == "") throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5) throw "too low";
    if(x > 10) throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
}
</script>
</body>
</html>
```

HTML Validation

The code above is just an example.

Modern browsers will often use a combination of JavaScript and built-in HTML validation, using predefined validation rules defined in HTML attributes:

```
<input id="demo" type="number" min="5" max="10" step="1">
```

You can read more about forms validation in a later chapter of this tutorial.

The finally Statement

The finally statement lets you execute code, after try and catch, regardless of the result:

Syntax

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
finally {
  Block of code to be executed regardless of the try / catch result
}
```

Example

```
function myFunction() {
  const message = document.getElementById("p01");
  message.innerHTML = "";
  let x = document.getElementById("demo").value;
  try {
    if(x.trim() == "") throw "is empty";
    if(isNaN(x)) throw "is not a number";
    x = Number(x);
    if(x > 10) throw "is too high";
    if(x < 5) throw "is too low";
  }
  catch(err) {
    message.innerHTML = "Error: " + err + ".";
  }
  finally {
    document.getElementById("demo").value = "";
  }
}
```

The Error Object

JavaScript has a built in error object that provides error information when an error occurs.

The error object provides two useful properties: name and message.

Error Object Properties

Property	Description
name	Sets or returns an error name
message	Sets or returns an error message (a string)

Error Name Values

Six different values can be returned by the error name property:

Error Name	Description
EvalError	An error has occurred in the eval() function
RangeError	A number “out of range” has occurred
ReferenceError	An illegal reference has occurred
SyntaxError	A syntax error has occurred
TypeError	A type error has occurred
URIError	An error in encodeURI() has occurred

The six different values are described below.

Eval Error

An EvalError indicates an error in the eval() function.

Newer versions of JavaScript do not throw EvalError. Use SyntaxError instead.

Range Error

A RangeError is thrown if you use a number that is outside the range of legal values.

For example: You cannot set the number of significant digits of a number to 500.

Example

```
let num = 1;
try {
  num.toPrecision(500);   // A number cannot have 500 significant digits
}
catch(err) {
  document.getElementById(“demo”).innerHTML = err.name;
}
```

Reference Error

A ReferenceError is thrown if you use (reference) a variable that has not been declared:

Example

```
let x = 5;
try {
  x = y + 1;   // y cannot be used (referenced)
}
catch(err) {
  document.getElementById(“demo”).innerHTML = err.name;
}
```

Syntax Error

A SyntaxError is thrown if you try to evaluate code with a syntax error.

Example

```
try {
  eval(“alert(‘Hello)”);   // Missing ‘ will produce an error
}
```

```
catch(err) {
  document.getElementById("demo").innerHTML = err.name;
}
```

Type Error

A TypeError is thrown if an operand or argument is incompatible with the type expected by an operator or function.

Example

```
let num = 1;
try {
  num.toUpperCase();   // You cannot convert a number to upper case
}
catch(err) {
  document.getElementById("demo").innerHTML = err.name;
}
```

URI (Uniform Resource Identifier) Error

A URIError is thrown if you use illegal characters in a URI function:

Example

```
try {
  decodeURI("%%%");   // You cannot URI decode percent signs
}
catch(err) {
  document.getElementById("demo").innerHTML = err.name;
}
```

Non-Standard Error Object Properties

Mozilla and Microsoft define some non-standard error object properties:

fileName (Mozilla)

lineNumber (Mozilla)

columnNumber (Mozilla)

stack (Mozilla)

description (Microsoft)

number (Microsoft)

Do not use these properties in public web sites. They will not work in all browsers.

Concept of AJAX

AJAX is a developer's dream, because you can:

- Read data from a web server - after a web page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

HTML Page

```
<!DOCTYPE html>
<html>
```

```
<body>
<div id=”demo”>
  <h2>Let AJAX change this text</h2>
  <button type=”button” onclick=”loadDoc()”>Change Content</button>
</div>
</body>
</html>
```

The HTML page contains a section and a <button>.

The section is used to display information from a server.

The <button> calls a function (if it is clicked).

The function requests data from a web server and displays it:

Function loadDoc()

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById(“demo”).innerHTML = this.responseText;
    }
  };
  xhttp.open(“GET”, “ajax_info.txt”, true);
  xhttp.send();
}
```

What is AJAX?

AJAX is an acronym for Asynchronous JavaScript and XML. It is a group of inter-related technologies like JavaScript, DOM, XML, HTML/XHTML, CSS, XMLHttpRequest etc.

AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

AJAX = Asynchronous JavaScript And XML.

AJAX is not a programming language.

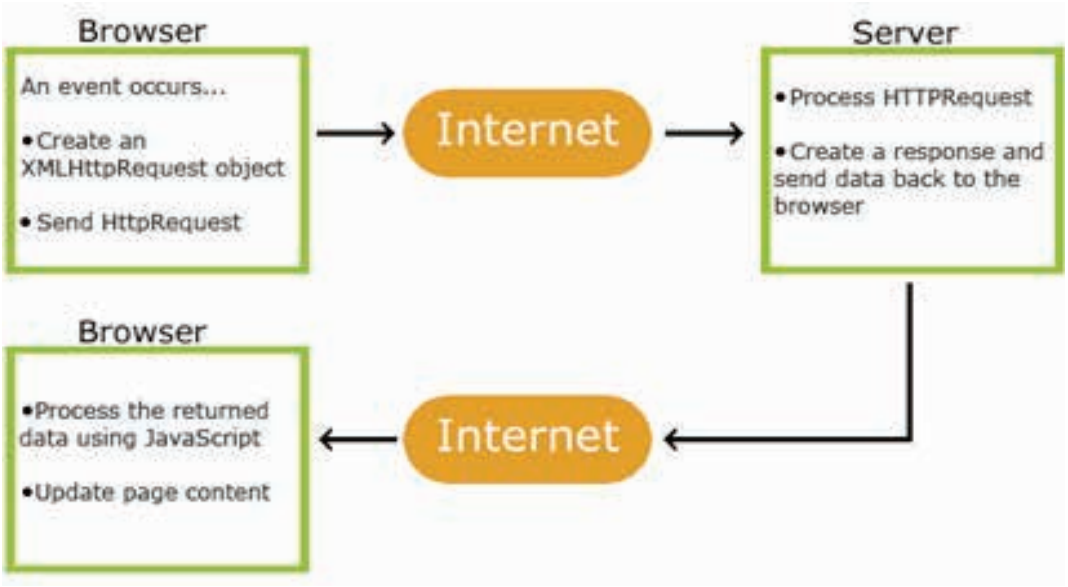
AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works



1 An event occurs in a web page (the page is loaded, a button is clicked)

2 An XMLHttpRequest object is created by JavaScript

3 The XMLHttpRequest object sends a request to a web server

4 The server processes the request

5 The server sends a response back to the web page

6 The response is read by JavaScript

7 Proper action (like page update) is performed by JavaScript

MODULE 4 : PHP (Hyper Text Pre Processor)

LESSON 47 - 62: PHP (Hyper Text Pre Processor)

Objectives

At the end of this lesson, you will be able to:

- create, open, read, write, delete & close files on the server
- develop server side scripting
- create & develop interactive website & web applications.

PHP, its features and advantages

Introduction to PHP

What is PHP (Hyper Text Pre Processor)?

PHP, which stands for Hyper Text Preprocessor, is a widely-used open-source server-side scripting language. Originally designed for web development, PHP is now also used as a general-purpose programming language. It was created by Rasmus Lerdorf in 1994 and has since evolved into one of the most widely used programming languages on the web.

It is embedded within HTML code and executed on the server, generating dynamic content that is sent to the user's web browser.

Origins and Evolution:

PHP's inception was as a set of Common Gateway Interface (CGI) binaries written in the C programming language to track online visits to Rasmus Lerdorf's resume. Over time, as more functionality was added, it transformed into a scripting language capable of building dynamic web pages.

Server-Side Scripting:

One of PHP's defining features is its server-side scripting capability. Unlike client-side scripting languages such as JavaScript, PHP code is executed on the server, generating HTML or other output sent to the user's browser. This allows for the creation of dynamic and interactive web pages.

Example:

Consider a simple "Hello, World!" program in PHP:

```php
<?php
   // PHP code goes here
   echo "Hello, PHP!";
?>
```

In this example, the PHP code is enclosed within `<?php ... ?>` tags. The `echo` statement is used to output the text "Hello, World!" to the web page.

PHP - Environment Setup

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

What Do I Need?

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

To install PHP, it is best to install AMP (Apache, MySQL, PHP). There are many options for different operating systems. As,

- WAMP (Window Apache, MySQL, PHP) for Windows Operating Systems
- LAMP (Linux Apache, MySQL, PHP) for Linux Operating Systems
- MAMP (Mac Apache, MySQL, PHP) for MAC Operating System
- XAMPP (Cross Apache, MySQL, PHP, PERL) It is cross platform, apart from this it also provides FileZilla, Mercury Mail,

  To install PHP, it is best to install AMP (Apache, MySQL, PHP). There are many options for different operating systems.

Download WAMP Server

https://www.wampserver.com/en/

Download LAMP Server

https://csg.sph.umich.edu/abecasis/LAMP/download/

Download MAMP Server

https://www.mamp.info/en/downloads/

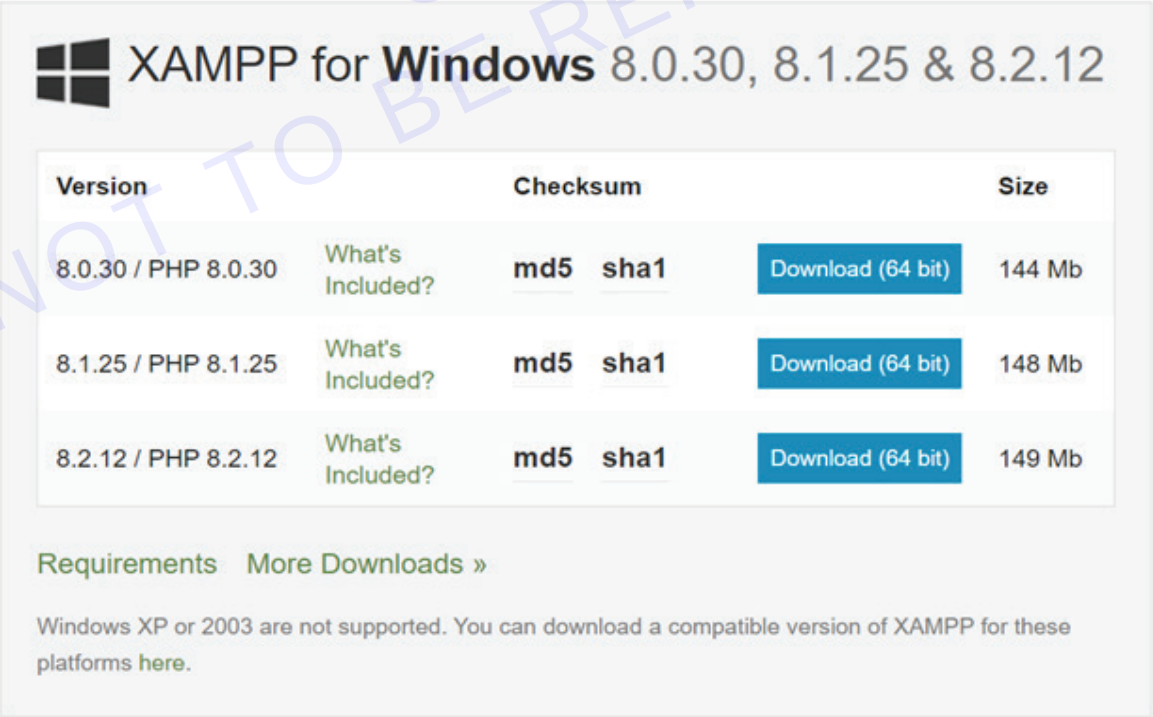
Download and Install XAMPP Server

https://www.apachefriends.org/download.html

After downloading the application by clicking on the links given above, you can go to the download directory and install it by double clicking.

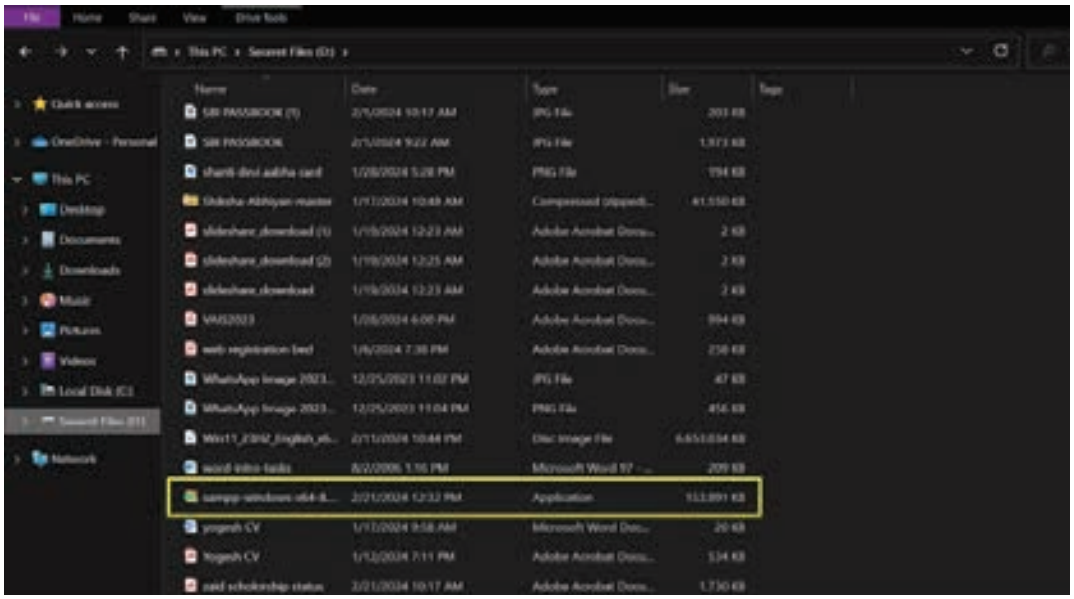
Steps to Install XAMPP

To download XAMPP, visit the official website.



Now download XAMPP according to your operating system (32 Bit or 64 Bit)

Now install XAMPP by double clicking on the downloaded file.



PHP Version

To check your php version you can use the phpversion() function:

PHP Run First Program

In this program you will learn how to run PHP programs on XAMPP.

Write and save program

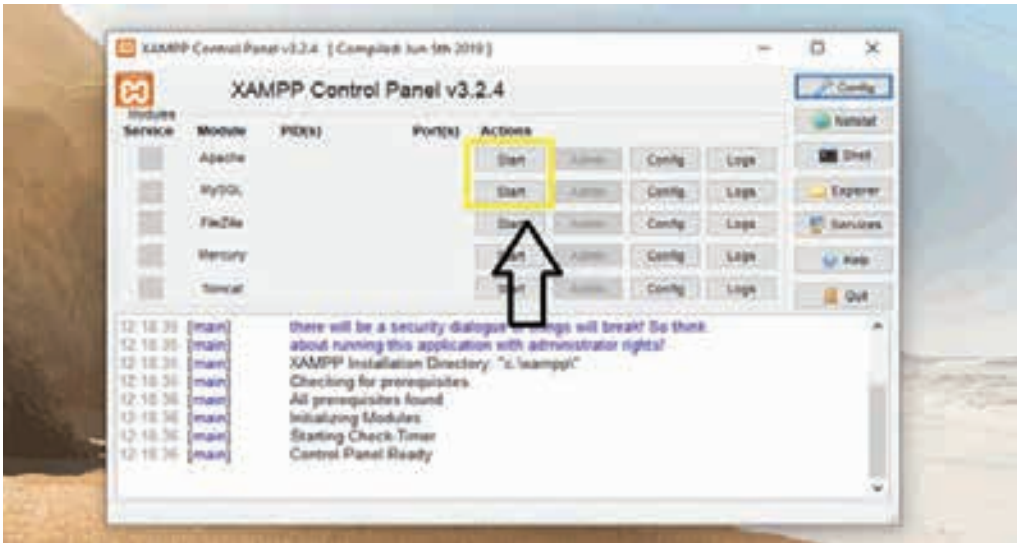
To run a PHP program, first write the PHP program in a text editor. And keep it anywhere in C:/xampp/htdocs. If possible, you can also create a new directory. I have created a directory named test inside the htdocs folder. And the name test.php is saved inside this directory.

Ex. C:/xampp/htdocs/test/test.php

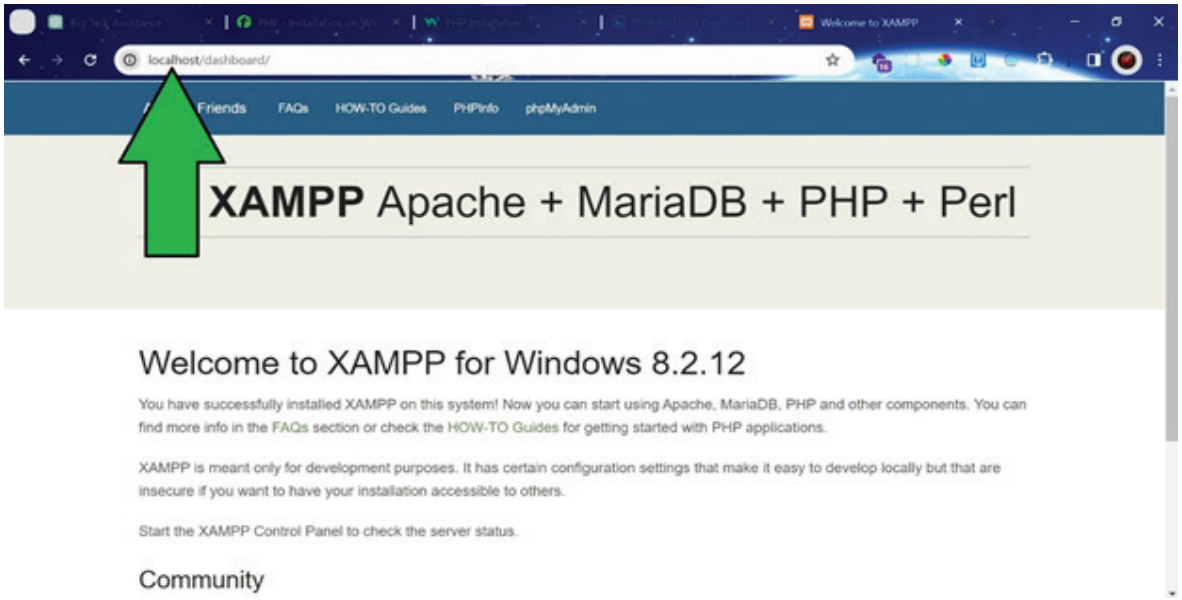
```
<?php     $var = "Hello World !";
    echo $var;
  ?>
```

Open XAMPP Control Panel

Now open the XAMPP Control Panel, and then start the PHP Server. If you need MYSQL then you can also start it.



Open Browser



Now Locate the file

Now remove dashboard after localhost and locate wherever you have saved the file and enter. I saved my file in C:/xampp/htdocs/test/test.php.

So write like this in the browser localhost/test/test.php then write like this in the browser and you will see the result something like this.

Output

Hello World !

Now open the browser or click on Admin of PHP and MySQL in XAMPP Control Panel or write localhost in the browser and enter. After entering you will see the output something like this.

Features and Advantages of PHP

PHP comes with several features and advantages that contribute to its popularity in web development.

1 **Easy to Learn and Use:**

- PHP has a syntax that is similar to C and other programming languages, making it easy for developers to learn and use.

```php
<?php
   $variable = "Hello, PHP!";
   echo $variable;
?>
```

2 **Open Source:**

- PHP is open-source, meaning it is freely available for use, distribution, and modification.

3 **Platform Independence:**

- PHP is platform-independent, which means it can run on various operating systems like Windows, Linux, and macOS.

4 **Support for Multiple Databases:**

- PHP supports a wide range of databases, including MySQL, PostgreSQL, and SQLite, making it flexible for database-driven web applications.

5 **Server-Side Scripting:**

- Being a server-side scripting language, PHP performs tasks on the server before sending the results to the client's browser. This allows for dynamic content generation.

6 **Integration with Other Technologies:**

- PHP can be easily integrated with other technologies like HTML, CSS, JavaScript, and various web servers.

7 **Extensive Community Support:**

- PHP has a large and active community of developers who contribute to its growth, providing support, resources, and a rich ecosystem of libraries and frameworks.

These features make PHP a powerful and versatile language for building dynamic and interactive web applications.

Basic PHP Syntax, tags, Data types, Constants and Variables, Operators and expressions

Getting Started with PHP

Basic PHP Syntax & Tags

PHP is a server-side scripting language, meaning that PHP scripts are executed on the server before the results are sent back to the browser as plain HTML. Understanding the basic syntax is crucial for effective PHP coding.

```
<?php
    // PHP code goes here
?>
```

Placing PHP Script: A PHP script can be placed anywhere in the HTML document. It starts with <?**php** and ends with **?**>. The default file extension for PHP files is ".php".

Example PHP File: A typical PHP file contains a mix of HTML and PHP scripting code. Below is an example of a simple PHP file that outputs "Hello World!" using the built-in PHP function

echo

Note that PHP statements end with a semicolon (;).

```
<!DOCTYPE html>
<html>
<body>

<?php
   ECHO "Hello World!<br>";
   echo "Hello World!<br>";
   EcHo "Hello World!<br>";
?>

</body>
</html>
```

PHP Case Sensitivity: PHP keywords, classes, functions, and user-defined functions are not case-sensitive. However, variable names are case-sensitive. In the example below, various case variations of the **echo** keyword are considered equal and legal.

Note: Variable names, on the other hand, must be used consistently with their defined case.

Data Types in PHP

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)

- Boolean
- Array
- Object
- NULL
- Resource

Getting the Data Type

You can get the data type of any object by using the **var_dump()** function.

Example

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5;
var_dump($x);
?>

</body>
</html>
```

```
int(5)
```

The **var_dump()** function returns the data type and the value:

PHP String

A string is a sequence of characters, like “Hello world!”.

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Hello world!";
$y = 'Hello world!';

var_dump($x);
echo "<br>";
var_dump($y);
?>

</body>
</html>
```

```
string(12) "Hello world!"
string(12) "Hello world!"
```

A string can be any text inside quotes. You can use single or double quotes:

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example **$x** is an integer. The PHP **var_dump()** function returns the data type and value:

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 10.365;
var_dump($x);
?>

</body>
</html>
```

```
float(10.365)
```

In the following example **$x** is a float. The PHP **var_dump()** function returns the data type and value:

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = true;
var_dump($x);
?>

</body>
</html>
```

```
bool(true)
```

Booleans are often used in conditional testing.

PHP Array

An array stores multiple values in one single variable.

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>

</body>
</html>
```

```
array(3) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
}
```

In the following example **$cars** is an array. The PHP **var_dump()** function returns the data type and value:

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named **Car** that can have properties like model, color, etc. We can define variables like **$model, $color**, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

```
<!DOCTYPE html>
<html>
<body>

<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("red", "Volvo");
var_dump($myCar);
?>

</body>
</html>
```

```
object(Car)#1 (2) { ["color"]=> string(3) "red" ["model"]=> string(5) "Volvo" }
```

If you create a **__construct()** function, PHP will automatically call this function when you create an object from a class.

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>

</body>
</html>
```

```
NULL
```

Tip: If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

Change Data Type

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5;
var_dump($x);

echo "<br>";

$x = "Hello";
var_dump($x);
?>

<p>Line breaks were added for better readability.</p>

</body>
</html>
```

```
int(5)
string(5) "Hello"

Line breaks were added for better readability.
```

If you assign an integer value to a variable, the type will automatically be an integer.

If you assign a string to the same variable, the type will change to a string

PHP Variables

In PHP, a variable is declared using a $ sign followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.

```
$variablename=value;
```

- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

```php
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

```
string is: hello string
integer is: 200
float is: 44.6
```

PHP Variable: Declaring string, integer, and float

```php
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

```
11
```

Let's see the example to store string, integer, and float values in PHP variables.

File: variable1.php

File: variable2.php

```php
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Output:

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5
My boat is
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

PHP Variable: Rules

```php
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>
```

Output:

```
hello
hello
```

PHP variables must start with letter or underscore only.

```
<?php
$4c="hello";//number (invalid)
$*d="hello";//special symbol (invalid)

echo "$4c <br/> $*d";
?>
```

Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting variable (T_VA
 or '$' in C:\wamp\www\variableinvalid.php on line 2
```

PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

File: variableinvalid.php

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

1 Using define() function

2 Using const keyword

Constants are similar to the variable except once they defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. For example, it can be started with a letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

```
define(name, value, case-insensitive)
```

Note: Unlike variables, constants are automatically global throughout the script.

PHP constant: define()

Use the define() function to create a constant. It defines constant at run time. Let's see the syntax of define() function in PHP.

name: It specifies the constant name.

value: It specifies the constant value.

case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

```
<?php
define("MESSAGE","Hello PHP");
echo MESSAGE;
?>
```

Hello PHP

Let's see the example to define PHP constant using define().

```
<?php
define("MESSAGE","Hello PHP",true);//not case sensitive
echo MESSAGE, "</br>";
echo message;
?>
```

Hello PHP
Hello PHP

File: constant1.php

```php
<?php
define("MESSAGE","Hello PHP",false);//case sensitive
echo MESSAGE;
echo message;
?>
```

```
Hello PHPmessage
Notice: Use of undefined constant message - assumed 'message'
in C:\wamp\www\vconstant3.php on line 4
message
```

Create a constant with case-insensitive name:

File: constant2.php

File: constant3.php

```php
<?php
const MESSAGE="Hello PHP const";
echo MESSAGE;
?>
```

```
Hello PHP const
Notice: Use of undefined constant message - assumed 'message'
in C:\wamp\www\vconstant3.php on line 4
message
```

PHP constant: const keyword

PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are case-sensitive.

File: constant4.php

```php
constant (name)
```

Constant() function

```php
<?php
    define("MSG", "Hello World");
    echo MSG, "</br>";
    echo constant("MSG");
    //both are similar
?>
```

```
Hello World
Hello World
```

There is another way to print the value of constants using constant() function instead of using the echo statement.

Constant	Variables
Once the constant is defined, it can never be redefined.	A variable can be undefined as well as redefined easily.

Constant	Variables
A constant can only be defined using define() function. It cannot be defined by any simple assignment.	A variable can be defined by simple assignment (=) operator.
There is no need to use the dollar ($) sign before constant during the assignment.	To declare a variable, always use the dollar ($) sign before the variable.
Constants do not follow any variable scoping rules, and they can be defined and accessed anywhere.	Variables can be declared anywhere in the program, but they follow variable scoping rules.
Constants are the variables whose values can't be changed throughout the program.	The value of the variable can be changed.
By default, constants are global.	Variables can be local, global, or static.

Syntax:

The syntax for the following constant function:

File: constant5.php

Constant vs Variables

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

- Conditional assignment operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

expression on the right.

The PHP comparison operators are used to compare two values (number or string):

PHP Comparison Operators

Operator	Same as...	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

The PHP comparison operators are used to compare two values (number or string):

PHP Increment / Decrement Operators

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP String Operators

PHP has two operators that are specially designed for strings

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

Paginators, popovers, progress, spinner

PHP Array Operators

The PHP array operators are used to compare arrays.

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

User Interface Components

In PHP, you typically handle the backend logic and data processing, while user interface components like paginators, popovers, progress bars, spinners, tables, toasts, and tooltips are usually implemented using HTML, CSS, and JavaScript. However, you can integrate PHP with these components to dynamically generate or manipulate them based on server-side data.

Here's a brief overview of how you can integrate PHP with these UI components:

- Paginators
- Popovers
- Progress and Spinner
- Tables, Toasts, Tooltips in php

Paginators

Pagination in PHP refers to the process of dividing a large set of data into smaller, more manageable sections called "pages." It is commonly used in web applications to display a limited number of records or results per page, allowing users to navigate through the data easily.

Pagination is essential when dealing with large data sets because displaying all the records on a single page can lead to performance issues and an overwhelming user interface. By implementing pagination, you can improve the user experience and optimize the performance of your application.

```php
<?php
   // Assume $db is a MySQLi database connection
   $resultsPerPage = 10;
   $currentPage = isset($_GET['page']) ? $_GET['page'] : 1;
   $offset = ($currentPage - 1) * $resultsPerPage;

   $query = "SELECT * FROM your_table LIMIT $offset, $resultsPerPage";
   $result = mysqli_query($db, $query);

   // Display the results
   while ($row = mysqli_fetch_assoc($result)) {
      // Display data
   }

   // Implement pagination links
   // Example: << < 1 2 3 > > for navigating between pages
?>
```

PHP program to pagination

- Create a new PHP file in your working location and save it with your desired name.
- Here we are using student data table with 121 records in localhost database.

Example:

Consider a scenario where you fetch data from a MySQL database and implement pagination:

Popovers

Popovers are interactive UI components that display additional information or options when triggered. They are commonly used for tooltips or contextual information.

Example:

Using Bootstrap, you can create a simple popover:

```
<!DOCTYPE html>
<html>
<head>
   <title>Progress Bar Example</title>
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>

   <div class="progress">
      <div class="progress-bar" style="width:70%"></div>
   </div>

</body>
</html>
```

Tables, Toasts, Tooltips

Progress and Spinner

Progress bars and spinners provide visual feedback on the status of ongoing processes. They are particularly useful when loading data or performing time-consuming operations.

Example:

```
<!DOCTYPE html>
<html>
<head>
   <title>Table with Toasts and Tooltips</title>
   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>

   <table class="table">
      <thead>
         <tr>
            <th scope="col">#</th>
            <th scope="col">First Name</th>
            <th scope="col">Last Name</th>
            <th scope="col" data-toggle="tooltip" title="Age Tooltip">Age</th>
         </tr>
      </thead>
      <tbody>
         <tr data-toggle="toast" title="Click for Details">
            <th scope="row">1</th>
            <td>John</td>
            <td>Doe</td>
            <td>30</td>
         </tr>
         <!-- More rows... -->
      </tbody>
   </table>

   <script>
      $(document).ready(function(){
         $('[data-toggle="tooltip"]').tooltip();
         $('[data-toggle="toast"]').toast();
      });
   </script>

</body>
</html>
```

Using Bootstrap for a progress bar:

Tables are foundational for displaying structured data, while toasts and tooltips enhance user notifications and interactions.

Bootstrap styling essentials and its use. Explain typography, floats, flex, Alignment, borders, position of elements shadow and visibility in bootstrap

Example:

Creating a table with toasts and tooltips:

These examples showcase professional and interactive UI components that enhance user experience. As you incorporate these elements into your PHP projects, remember to tailor them to suit your specific requirements.

Bootstrap

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter, and released as an open source product in August 2011 on GitHub. Bootstrap is a free front-end framework for faster and easier web development. It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins. It also gives you the ability to easily create responsive designs. Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

Uses of Bootstrap

- **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
- **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework
- **Browser compatibility:** Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Edge, Safari, and Opera

Bootstrap 3, which was released in 2013. However, we also cover newer versions; **Bootstrap 4 (released 2018)** and **Bootstrap 5 (released 2021)**. **Bootstrap 5** is the newest version of **Bootstrap**; with new components, faster stylesheets, more responsiveness etc. It supports the latest, stable releases of all major browsers and platforms, **Bootstrap 3** and Bootstrap 4 is still supported by the team for critical bug fixes and documentation changes, and it is perfectly safe to continue to use them. However, new features will NOT be added to them.

1 **Typography:** Typography refers to the style, appearance, and arrangement of text on a page. Bootstrap provides a comprehensive set of typographic styles through its utility classes. These classes allow you to easily style text elements such as headings, paragraphs, lists, and more. For example, you can use classes like .h1, .lead, .text-muted, .font-weight-bold, etc., to apply various typographic styles to your content.

2 **Float:** Bootstrap's float classes (float-left and float-right) are used to float elements to the left or right within their parent containers. This is commonly used for creating multi-column layouts where elements are aligned side by side.

3 **Flex:** Bootstrap 4 introduced Flexbox-based layout utilities that allow for more flexible and responsive layouts. With Flexbox utilities, you can easily create complex layouts and control the alignment, direction, order, and sizing of elements within a container. Classes like .d-flex, .justify-content-*, .align-items-*, .flex-grow-*, etc., enable you to manipulate the layout of elements using Flexbox.

4 **Alignment:** Bootstrap provides alignment utilities to control the alignment of content within its containers. Classes like .text-left, .text-center, .text-right, .align-items-*, .justify-content-*, etc., allow you to align text, inline elements, and flex items both horizontally and vertically.

5 **Borders:** Bootstrap offers border utilities to add or remove borders from elements. You can use classes like. border, .border-top, .border-bottom, .border-left, .border-right, .rounded, .rounded-*, etc., to customize the borders of elements. Additionally, Bootstrap provides responsive border color and border radius utilities.

6 **Position of Element:** Bootstrap provides positioning utilities to control the position of elements within their containers or on the page. Classes like. position-static, .position-relative, .position-absolute, .position-fixed, .position-sticky, etc., enable you to specify the positioning behavior of elements. These utilities are particularly useful for creating sticky headers, footers, or sidebars.

7 **Shadow:** Bootstrap includes shadow utilities to add shadows to elements. You can use classes like. shadow, .shadow-sm, .shadow-lg, etc., to apply different levels of shadow to elements. These utilities are handy for adding depth and dimension to elements, such as cards, modals, or buttons.

8 **Visibility:** Bootstrap provides visibility utilities to control the visibility of elements at different breakpoints. Classes like. visible-* and .invisible-* allow you to show or hide elements based on the screen size. This helps in creating responsive designs where certain elements are displayed or hidden depending on the device's viewport size

PHP Conditional Events, Flow control and looping in PHP

PHP Conditional Events and Flow Control

Control structures in php are fundamental programming concepts that enable developers to write powerful and efficient code. They allow the program to execute specific instructions based on certain conditions. There are three primary types of control structures: selection, iteration, and sequence. The selection structure executes a set of instructions if a specific condition is met. The iteration structure is used when a set of instructions needs to be repeated multiple times based on a given condition. The sequence structure is used to execute instructions in sequential order. These structures are essential to writing efficient and reliable code.

Introduction

Control structures in php are an essential part of any programming language, including PHP. They are used to control the flow of a program by allowing the programmer to specify which statements should be executed under certain conditions. Understanding control structures in PHP is crucial for any aspiring PHP developer, as they form the backbone of many programming tasks.

Control structures in PHP are used to make decisions based on conditions, loop through a set of instructions multiple times, and break out of a loop or switch statement. The most common control structures used in PHP are if-else statements, loops such as for and while loops, and switch statements.

By using if-else statements, a programmer can check a certain condition and execute a block of code if the condition is true or execute a different block of code if the condition is false. Loops allow a programmer to repeat a set of instructions multiple times, either for a specified number of times or until a certain condition is met. Switch statements provide an alternative to multiple if-else statements, allowing a programmer to check multiple conditions with a single statement. Control structures are a vital part of any programming language, and understanding them is essential for any PHP developer. By mastering the use of control structures, a programmer can create more efficient and effective PHP code, leading to better software applications.

Conditional statements

Conditional statements are used to perform different actions based on different conditions.

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions

```
if (condition) {
  // code to be executed if condition is true;
}
```

- switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

```
<!DOCTYPE html>
<html>
<body>

<?php
if (5 > 3) {
  echo "Have a good day!";
}
?>

</body>
</html>
```

Have a good day!

The if statement executes some code if one condition is true.

Syntax

Example

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = 14;

if ($t < 20) {
  echo "Have a good day!";
}
?>

</body>
</html>
```

Have a good day!

Output “Have a good day!” if 5 is larger than 3:

We can also use variables in the if statement:

Example

```
if (condition) {
  // code to be executed if condition is true;
} else {
  // code to be executed if condition is false;
}
```

Output “Have a good day!” if $t is less than 20:

PHP - The if...else Statement

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>

</body>
</html>
```

Have a good day!

The if...else statement executes some code if a condition is true and another code if that condition is false.

Syntax

Example

```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  // code to be executed if first condition is false and this condition is true;
} else {
  // code to be executed if all conditions are false;
}
```

Output “Have a good day!” if the current time is less than 20, and “Have a good night!” otherwise:

PHP - The if...elseif...else Statement

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>

</body>
</html>
```

The hour (of the server) is 10, and will give the following message:
Have a good day!

The if...elseif...else statement executes different codes for more than two conditions.

Syntax

Example

```
switch (expression) {
  case label1:
    //code block
    break;
  case label2:
    //code block;
    break;
  case label3:
    //code block
    break;
  default:
    //code block
}
```

Output “Have a good morning!” if the current time is less than 10, and “Have a good day!” if the current time is less than 20. Otherwise it will output “Have a good night!”:

The switch statement is used to perform different actions based on different conditions.

The PHP switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

This is how it works:

- The expression is evaluated once

```
<!DOCTYPE html>
<html>
<body>

<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>

</body>
</html>
```

Your favorite color is red!Your favorite color is blue!

- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The break keyword breaks out of the switch block
- The default code block is executed if there is no match

Example

The break Keyword

When PHP reaches a break keyword, it breaks out of the switch block.

This will stop the execution of more code, and no more cases are tested.

The last block does not need a break, the block breaks (ends) there anyway.

```
<!DOCTYPE html>
<html>
<body>

<?php
$d = 4;

switch ($d) {
  case 6:
    echo "Today is Saturday";
    break;
  case 0:
    echo "Today is Sunday";
    break;
  default:
    echo "Looking forward to the Weekend";
}
?>

</body>
</html>
```

Looking forward to the Weekend

Warning: If you omit the break statement in a case that is not the last, and that case gets a match, the next case will also be executed even if the evaluation does not match the case!

Example

What happens if we remove the break statement from case “red”?

```
<!DOCTYPE html>
<html>
<body>

<?php
$d = 4;

switch ($d) {
  case 6:
    echo "Today is Saturday";
    break;
  case 0:
    echo "Today is Sunday";
    break;
  default:
    echo "Looking forward to the Weekend";
}
?>

</body>
</html>
```

Looking forward to the Weekend

$favcolor is red, so the code block from case “red” is executed, but since it has no break statement, the code block from case “blue” will also be executed:

The default Keyword

```
<!DOCTYPE html>
<html>
<body>

<?php
$d = 4;

switch ($d) {
  default:
    echo "Looking forward to the Weekend";
    break;
  case 6:
    echo "Today is Saturday";
    break;
  case 0:
    echo "Today is Sunday";
}
?>

</body>
</html>
```

Looking forward to the Weekend

The default keyword specifies the code to run if there is no case match:

Example

If no cases get a match, the default block is executed:

The default case does not have to be the last case in a switch block:

Example

Putting the default block elsewhere than at the end of the switch block is allowed, but not recommended.

Note: If default is not the last block in the switch block, remember to end the default block with a break statement.

Looping In PHP

In the following chapters you will learn how to repeat code by using loops in PHP.

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true

```
<!DOCTYPE html>
<html>
<body>

<?php
$i = 1;

while ($i < 6) {
  echo $i;
  $i++;
}
?>

</body>
</html>
```

```
12345
```

- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

The following chapters will explain and give examples of each loop type.

PHP while Loop

The while loop - Loops through a block of code as long as the specified condition is true.

Example:

Print $i as long as $i is less than 6:

Note: remember to increment $i, or else the loop will continue forever.

The while loop does not run a specific number of times, but checks after each iteration if the condition is still true.

The condition does not have to be a counter, it could be the status of an operation or any condition that evaluates to either true or false.

PHP do while Loop

The do...while loop - Loops through a block of code once, and then repeats the loop as long as the specified condition is true.

```
<html>
<body>

<?php
$i = 8;

do {
  echo $i;
  $i++;
} while ($i < 6);
?>

<p>As you can see, the code is executed once, even if the condition is never true.</p>

</body>
</html>
```

```
8

As you can see, the code is executed once, even if the condition is never true.
```

Example:

Print $i as long as $i is less than 6:

```
for (expression1, expression2, expression3) {
  // code block
}
```

Note: In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false. See example below.

Let us see what happens if we set the $i variable to 8 instead of 1, before execute the same do...while loop again:

Example:

Set $i = 8, then print $i as long as $i is less than 6:

PHP for Loop

```
<!DOCTYPE html>
<html>
<body>

<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>

</body>
</html>
```

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

The for loop - Loops through a block of code a specified number of times.

Syntax

This is how it works:

- expression1 is evaluated once
- expression2 is evaluated before each iterarion
- expression3 is evaluated after each iterarion

Example:

Print the numbers from 0 to 10:

Example Explained

1 The first expression, $x = 0;, is evaluated once and sets a counter to 0.

2 The second expression, $x <= 10;, is evaluated before each iteration, and the code block is only executed if this expression evaluates to true. In this example the expression is true as long as $x is less than, or equal to, 10.

```
<!DOCTYPE html>
<html>
<body>

<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $x) {
  echo "$x <br>";
}
?>

</body>
</html>
```

red
green
blue
yellow

3 The third expression, $x++;, is evaluated after each iteration, and in this example, the expression increases the value of $x by one at each iteration.

PHP for each Loop

The for each loop - Loops through a block of code for each element in an array or each property in an object.

The for each Loop on Arrays

The most common use of the foreach loop, is to loop through the items of an array.

Example:

Loop through the items of an indexed array:

For every loop iteration, the value of the current array element is assigned to the variable $x. The iteration continues until it reaches the last array element.

```
<!DOCTYPE html>
<html>
<body>

<?php
$members = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach ($members as $x => $y) {
  echo "$x : $y <br>";
}
?>

</body>
</html>
```

Peter : 35
Ben : 37
Joe : 43

Keys and Values

The array above is an indexed array, where the first item has the key 0, the second has the key 1, and so on.

Associative arrays are different, associative arrays use named keys that you assign to them, and when looping through associative arrays, you might want to keep the key as well as the value.

This can be done by specifying both the key and value in the foreach defintition, like this:

Example:

Print both the key and the value from the $members array:

The for each Loop on Objects

The for each loop can also be used to loop through properties of an object:

Example:

Print the property names and values of the $myCar object:

The break Statement

With the break statement we can stop the loop even if it has not reached the end:

Example:

Stop the loop if $x is “blue”:

The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

Example:

top, and jump to the next iteration if $x is “blue”:

Functions in PHP

Functions in PHP

In this section, we’ll explore the fundamental concepts of functions in PHP. Understanding the role of functions in programming and how they contribute to code organization and reusability is crucial. We’ll cover:

- What are Functions: Explanation of the concept of functions in programming.
- Why Use Functions: Benefits of using functions, such as code modularity and maintainability.
- Function Syntax: An overview of the basic structure of PHP functions

Introduction to Functions

In modern programming, functions play a vital role in organizing code and promoting reusability. They encapsulate a set of instructions and can be called upon to perform a specific task. Let’s explore the core concepts:

What are Functions:

A function is a block of code that performs a specific task. It enhances code modularity, making it easier to understand and maintain. Functions can be reused in different parts of a program.

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

Example:

```
function greet() {
    echo "Hello, World!";
}

// Calling the function
greet();
```

Why Use Functions:

Code Organization: Functions help break down a program into smaller, manageable pieces.

Reusability: Once defined, functions can be reused, reducing redundancy in your code.

Function Syntax:

In PHP, a basic function looks like this:

```
function functionName() {
    // code to be executed
}
```

Defining Functions:

Let’s dive into the process of creating and using custom functions in PHP.

Function Declaration:

To declare a function, use the `function` keyword, followed by the function name, and a pair of parentheses.

```php
function addNumbers($a, $b) {
    $sum = $a + $b;
    echo "Sum: $sum";
}
```

Note: A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

Tip: Give the function a name that reflects what the function does!

Call a Function

To call the function, just write its name followed by parentheses ():

Example:

```php
<!DOCTYPE html>
<html>
<body>

<?php
function myMessage() {
  echo "Hello world!";
}

myMessage();
?>

</body>
</html>
```

Hello world!

In our example, we create a function named myMessage().

The opening curly brace { indicates the beginning of the function code, and the closing curly brace } indicates the end of the function.

The function outputs “Hello world!”.

Function Name and Scope:

Function names are case-insensitive. However, it’s good practice to follow a consistent naming convention. Functions have their own scope, meaning variables inside a function are not accessible outside of it.

Function Parameters:

Parameters allow us to pass data into functions. They are defined within the parentheses during function declaration.

Example:

```php
function greetUser($name) {
    echo "Hello, $name!";
}

greetUser("John");
```

Function Body:

The function body contains the code to be executed when the function is called.

Parameters and Return Values

Functions can accept parameters and return values, making them versatile and adaptable to various scenarios.

There are three types of parameters:

- Required Parameters: Must be passed during the function call.
- Default Parameters: Have default values and can be omitted.
- Variable-length Parameter Lists: Allow an arbitrary number of arguments.

Example:

```php
function multiply($a, $b = 2) {
    return $a * $b;
}

$result = multiply(5); // $result is 10
```

Required Parameters:

Definition: Required parameters are values that must be provided when calling a function. These parameters are essential for the function to execute properly.

Example:

```php
function multiply($a, $b) {
    return $a * $b;
}

$result = multiply(5, 3);
```

In this example, the multiply function requires two parameters,` $a `and `$b`. When calling the function, you must provide values for both parameters (5 and 3 in this case).

PHP Default Parameter Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example:

```php
<!DOCTYPE html>
<html>
<body>

<?php
function setHeight($minheight = 50) {
  echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight();
setHeight(135);
setHeight(80);
?>

</body>
</html>
```

```
The height is : 350
The height is : 50
The height is : 135
The height is : 80
```

Variable-length Parameter Lists:

Definition: Variable-length parameter lists, also known as variadic functions, allow a function to accept an arbitrary number of arguments. This is useful when you don't know in advance how many arguments will be passed.

Example:

```php
function calculateSum(...$numbers) {
    $sum = 0;
    foreach ($numbers as $number) {
        $sum += $number;
    }
    return $sum;
}

$total = calculateSum(1, 2, 3, 4, 5);
```

In this example, the `calculateSum` function accepts any number of arguments using the `...$numbers` syntax. It then calculates the sum of all provided numbers. You can call this function with different numbers of arguments, and it will work accordingly.

```
$total = calculateSum(1, 2); // $total is 3
$total = calculateSum(1, 2, 3, 4, 5); // $total is 15
```

This flexibility is beneficial when you need to create functions that can handle various scenarios without explicitly defining the number of parameters.

PHP Functions - Returning values:

To let a function return a value, use the **return** statement:

Example:

```
<!DOCTYPE html>
<html>
<body>

<?php
function sum($x, $y) {
  $z = $x + $y;
  return $z;
}

echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);
?>

</body>
</html>
```

5 + 10 = 15
7 + 13 = 20
2 + 4 = 6

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Built-in functions in PHP are pre-defined functions that come with the language. These functions are readily available for use and cover a wide range of functionalities, from string manipulation to mathematical operations. Here's a more in-depth look:

Common PHP Functions:

Explore commonly used functions like `strlen`, `str`_`replace`, `count`, etc.

Example:

```
$string = "Hello, World!";
$length = strlen($string); // $length is 13
```

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

PHP MySql connection-Get connected with mysqli_connect

PHP MySQL Connection

Introduction:

Establishing a connection between PHP and MySQL is fundamental in building dynamic web applications. MySQL is a popular relational database management system, and PHP provides robust functions, particularly through the `mysqli` extension, to interact seamlessly with MySQL databases. This chapter will cover the process of connecting to MySQL using `mysqli_connect` and performing basic MySQL queries and operations.

Connecting to MySQL using mysqli_connect:

Explanation:

`mysqli_connect` is a PHP function that establishes a connection to a MySQL database. It requires specific parameters such as the MySQL server address, username, password, and the target database name.

Example:

```php
<?php
   // Database credentials
   $server = "localhost";
   $username = "root";
   $password = "";
   $database = "example_db";

   // Create a connection
   $connection = mysqli_connect($server, $username, $password,
$database);

   // Check the connection
   if (!$connection) {
      die("Connection failed: " . mysqli_connect_error());
   } else {
      echo "Connected successfully!";
   }
?>
```

In this example, we attempt to connect to a MySQL database on the local server with the given credentials. If the connection fails, an error message is displayed; otherwise, a success message is echoed.

Basic MySQL queries and operations

A list of commonly used MySQL queries to create database, use database, create table, insert record, update record, delete record, select record, truncate table and drop table are given below.

PHP MySQL Create Table

A database table has its own unique name and consists of columns and rows.

Create a MySQL Table Using MySQLi

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named “MyGuests”, with five columns: “id”, “firstname”, “lastname”, “email” and “reg_date”:

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
```

Notes on the table above:

The data type specifies what type of data the column can hold.

After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with

  PRIMARY KEY setting is often an ID number, and is often used with

  AUTO_INCREMENT

  Each table should have a primary key column (in this case: the “id” column). Its value must be unique for each record in the table.

The following examples shows how to create the table in PHP:

Example:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

Insert Data Into MySQL Using MySQLi

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

INSERT INTO table_name (column1, column2, column3,...)

VALUES (value1, value2, value3,...)

To learn more about SQL, please visit our SQL tutorial.

In the previous chapter we created an empty table named “MyGuests” with five columns: “id”, “firstname”, “lastname”, “email” and “reg_date”. Now, let us fill the table with data.

Note: If a column is AUTO_INCREMENT (like the “id” column) or TIMESTAMP with default update of current_ timesamp (like the “reg_date” column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

The following examples add a new record to the “MyGuests” table:

Example (MySQLi Object-oriented):

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:

```
SELECT column_name(s) FROM table_name
```

or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

Select Data With MySQLi

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

Example (MySQLi Object-oriented):

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // output data of each row
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

Update Data In a MySQL Table Using MySQLi

The UPDATE statement is used to update existing records in a table:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

The following examples update the record with id=2 in the “MyGuests” table:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

Example (MySQLi Object-oriented):

Delete Data From a MySQL Table Using MySQLi

```
DELETE FROM table_name
WHERE some_column = some_value
```

The DELETE statement is used to delete records from a table:

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Example (MySQLi Object-oriented):

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

Password encryption with SHA in online forms

Security Measures in PHP Password Encryption with SHA in Online Forms

Introduction:

Ensuring the confidentiality of user passwords is paramount in web development. Employing strong encryption techniques is vital, and one widely used method is hashing passwords with SHA (Secure Hash Algorithm).

Explanation:

The `password_hash` function in PHP simplifies the process of securely hashing passwords. When a user registers and sets a password, the hashed version is stored in the database. During login attempts, the entered password is hashed and compared to the stored hash.

```php
<?php
    // User registration: Hashing the password
    $password = "user_password";
    $hashedPassword = password_hash($password, PASSWORD_DEFAULT);

    // Storing $hashedPassword in the database
?>

<?php
    // User login: Verifying the entered password
    $enteredPassword = "entered_password";
    $storedHashedPassword = "retrieved_hashed_password_from_database";

    if (password_verify($enteredPassword, $storedHashedPassword)) {
        // Password is correct
    } else {
        // Password is incorrect
    }
?>
```

Cookies in PHP

Using Cookies in PHP

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the **setcookie()** function.

Syntax

setcookie(name, value, expire, path, domain, secure, httponly);

Only the name parameter is required. All other parameters are optional

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the **isset()** function to find out if the cookie is set:

Example:

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example:

```php
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
  echo "Cookies are enabled.";
} else {
  echo "Cookies are disabled.";
}
?>

</body>
</html>
```

Captcha text generation

CAPTCHA Text Generation in PHP

Introduction:

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a security mechanism designed to differentiate between human users and automated bots. Text-based CAPTCHAs involve generating and validating distorted text that users must interpret and enter correctly to prove they are human.

Explanation:

The primary purpose of a text-based CAPTCHA is to prevent automated scripts from submitting forms on websites, as bots often struggle to decipher and interpret distorted text. In PHP, CAPTCHAs can be implemented by generating a random text string, displaying it as an image, and validating the entered text against the stored value.

Implementation:

Let's break down the steps to create a simple text-based CAPTCHA in PHP using the GD library:

1 Generate a Random Text String:

```php
<?php
   session_start();

   // Generate a random string (customize the length as needed)
   $randomText = substr(md5(time()), 0, 5);

   // Store the generated text in the session for validation
   $_SESSION['captcha'] = $randomText;
?>
```

In this example, we use the `md5` function with the current timestamp to create a random string and store it in the session variable 'captcha.'

2 Create and Output the CAPTCHA Image:

```php
<?php
    header('Content-type: image/png');

    // Create an image with specified dimensions
    $image = imagecreate(150, 50);

    // Set background and text colors
    $background_color = imagecolorallocate($image, 255, 255, 255);
    $text_color = imagecolorallocate($image, 0, 0, 0);

    // Place the random text on the image
    imagestring($image, 5, 30, 15, $randomText, $text_color);

    // Output the image as PNG
    imagepng($image);
    imagedestroy($image);
?>
```

In this code snippet, we use the GD library to create an image with a specified background color, set text color, and place the random text on the image. The image is then output as PNG.

3 Embed the CAPTCHA in HTML:

```html
<img src="captcha.php" alt="CAPTCHA Image">
```

Include this HTML code on your form page to display the generated CAPTCHA image.

4 Validate the Entered Text:

Upon form submission, retrieve the entered text and validate it against the stored value in the session.

```php
<?php
    session_start();

    // Assuming $enteredText is the text entered by the user
    if ($enteredText === $_SESSION['captcha']) {
        // CAPTCHA validation successful
    } else {
        // CAPTCHA validation failed
    }
?>
```

Security Considerations:

- Randomness: Ensure the generated text is sufficiently random to challenge automated scripts effectively.
- Length and Complexity: Adjust the length and complexity of the generated text based on the desired security level.
- Session Storage: Store the generated CAPTCHA text securely in the session to validate it during form submission.

Implementing a text-based CAPTCHA adds an additional layer of security to your PHP forms, reducing the risk of automated submissions and enhancing the overall security of your web application.

MODULE 5 : Advanced Excel

LESSON 63 - 77 : Advance data Analysis using Excel

Objectives

At the end of this lesson, you will be able to:

- develop spread sheets by using advanced excel formulae
- manage advanced charts, tables & graphs
- develop simple power queries & power B1 for simple data visualisation.

Sheet Protection with password

Sheet Protection with password

In Microsoft Excel, you can protect a worksheet to prevent other users from making changes to the data, formatting, or other contents. Additionally, you can apply a password to the protection to ensure that only users who know the password can make changes to the protected sheet. Here's how you can do it in Excel:

1 **Open Excel:** Open the Excel workbook that contains the worksheet you want to protect.

2 **Select the Worksheet:** Click on the worksheet tab at the bottom of the Excel window to select the worksheet you want to protect.

3 **Protect Sheet**

- Go to the "Review" tab on the Excel ribbon.
- Click on "Protect Sheet" in the "Changes" group. This will open the "Protect Sheet" dialog box.

4 **Set Protection Options**

- In the "Protect Sheet" dialog box, you can specify what actions users are allowed to perform on the protected sheet. For example, you can allow users to select locked cells, format cells, or insert/delete rows and columns.
- Check or uncheck the options according to your preferences.

5 **Enter Password (Optional)**

- If you want to apply a password to the protection, enter the password in the "Password to unprotect sheet" field.
- Make sure to remember this password as it will be required to unprotect the sheet later. If you forget the password, you won't be able to make changes to the protected sheet.

6 **Confirm Password (Optional)**

- If you've entered a password, you will be prompted to confirm it by typing it again in the "Re-enter password to proceed" field.

7 **Click OK**

- Once you've set the protection options and entered the password (if desired), click the "OK" button to protect the sheet.

8 **Re-enter Password (Optional)**

- If you've set a password, Excel will prompt you to re-enter the password to confirm it. Type the password again and click "OK".

9 **Confirm Protection**

- After you've clicked "OK", the sheet will be protected based on the options you selected. You'll notice that some of the options in the ribbon are now or unavailable, indicating that the sheet is protected.

10 **Save Changes**

- Finally, don't forget to save your changes to the Excel workbook.

Now, your Excel worksheet is protected, and only users who know the password (if you've set one) can make changes to the protected elements based on the options you selected during the protection process.

Protecting a workbook in Excel

Protecting a workbook in Excel allows you to restrict access to the structure of the workbook, including its sheets, windows, and specific elements like macros and VBA code. Here's how you can protect a workbook in Excel.

1 **Open Excel Workbook:** Launch Microsoft Excel and open the workbook you want to protect.

2 **Protect Workbook Structure**

- Go to the "Review" tab on the Excel ribbon.
- Click on "Protect Workbook" in the "Changes" group. This will open the "Protect Structure and Windows" dialog box.

3 **Enter Password (Optional)**

- If you want to apply a password to the protection, enter the password in the "Password" field.
- Make sure to remember this password as it will be required to unprotect the workbook later. If you forget the password, you won't be able to make changes to the protected workbook structure.

4 **Confirm Password (Optional)**

- If you've entered a password, you'll be prompted to confirm it by typing it again in the "Reenter password to proceed" field.

5 **Set Protection Options**

- In the "Protect Structure and Windows" dialog box, you can choose which elements of the workbook you want to protect:
- "Structure": Protects the structure of the workbook, preventing users from adding, deleting, moving, hiding, or renaming sheets.
- "Windows": Prevents users from resizing, moving, or closing workbook windows.
- Check the checkboxes according to your preferences.

6 **Click OK**

- Once you've set the protection options and entered the password (if desired), click the "OK" button to protect the workbook.

7 **Re-enter Password (Optional)**

- If you've set a password, Excel will prompt you to re-enter the password to confirm it. Type the password again and click "OK".

8 **Confirm Protection**

- After you've clicked "OK", the workbook will be protected based on the options you selected.

9 **Save Changes**

- Don't forget to save your changes to the Excel workbook.

Now, your Excel workbook is protected, and only users who know the password (if you've set one) can modify the protected elements based on the options you selected during the protection process.

Flash Fill option

Using Flash Fill in Excel

Flash Fill automatically fills your data when it senses a pattern. For example, you can use Flash Fill to separate first and last names from a single column, or combine first and last names from two different columns.

Note: Flash Fill is only available in Excel 2013 and later.

How to use Flash Fill in Excel

Flash Fill is one of the most amazing features of Excel. It grabs a tedious task that would take hours to be performed manually and executes it automatically in a flash (hence the name). And it does so quickly and simply without you having to do a thing, but only provide an example of what you want.

What is Flash Fill in Excel?

Excel Flash Fill is a special tool that analyses the information you are entering and automatically fills data when it identifies a pattern.

The Flash Fill feature was introduced in Excel 2013 and is available in all later versions of Excel 2016, Excel 2019, Excel 2021, and Excel for Microsoft 365.

History

Started in December 2009 as an attempt of Sumit Gulwani, a senior researcher at Microsoft, to help a businesswoman he accidentally met at the airport with her merging challenge, a few years later it has evolved into a powerful ability to automate many Excel chores.

Flash Fill easily copes with dozens of different tasks that otherwise would require complex formulas or even VBA code such as splitting and combining text strings, cleaning data and correcting inconsistencies, formatting text and numbers, converting dates to the desired format, and a lot more.

Each time, Flash Fill combines millions of small programs that might accomplish the task, then sorts those code snippets using machine-learning techniques and finds the one that suits best for the job. All this is done in milliseconds in the background, and the user sees the results almost immediately!

Where is Flash Fill in Excel?

In Excel 2013 and later, the Flash Fill tool resides on the Data tab, in the Data tools group:

Excel Flash Fill shortcut

Those of you who prefer working from a keyboard most of the time, can run Flash Fill with this key combination: Ctrl + E

How to use Flash Fill in Excel

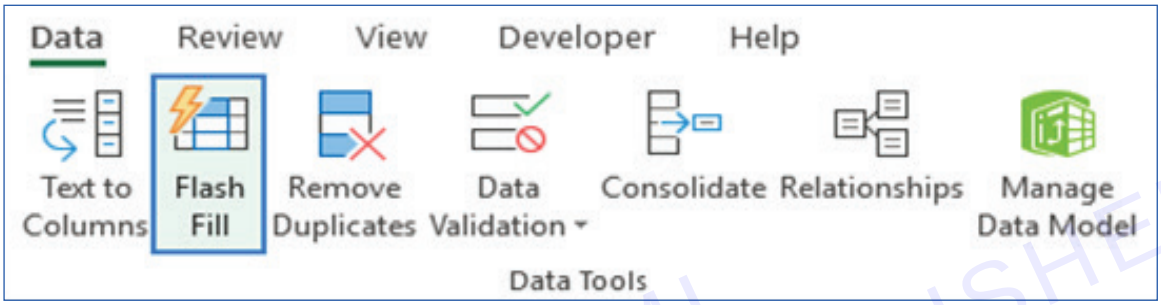
Usually Flash Fill starts automatically, and you only need to provide a pattern. Here's how:

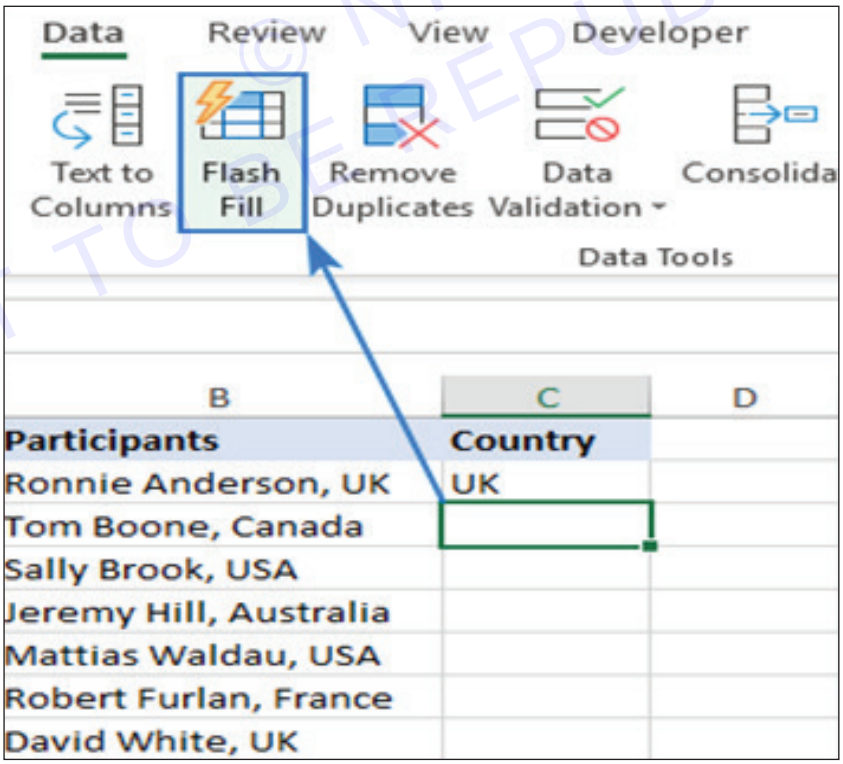
1 Insert a new column adjacent to the column with your source data.

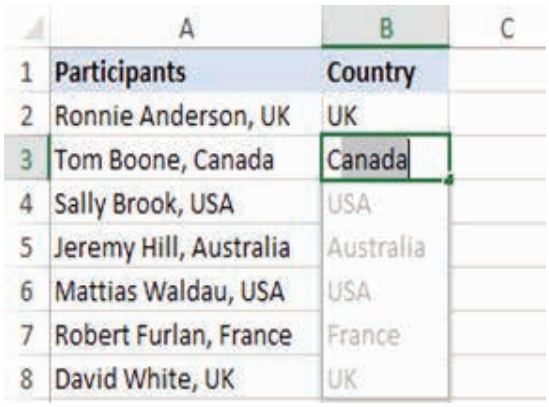
2 In the first cell of a newly added column, type the desired value.

3 Start typing in the next cell, and if Excel senses a pattern, it will show a preview of data to be auto-filled in the below cells.

4 Press the Enter key to accept the preview. Done!







	A	B	C
1	Participants	Country	
2	Ronnie Anderson, UK	UK	
3	Tom Boone, Canada	Canada	
4	Sally Brook, USA	USA	
5	Jeremy Hill, Australia	Australia	
6	Mattias Waldau, USA	USA	
7	Robert Furlan, France	France	
8	David White, UK	UK	

Tips:

- **If you are unhappy with the Flash Fill results, you can undo them by pressing Ctrl + Z or via the Flash Fill options menu.**
- **If Flash Fill does not start automatically, try these simple troubleshooting techniques.**
- **To find the number of the Flash Fill changed cells, just look at the Excel status bar.**

How to Flash Fill in Excel with a button click or shortcut

In most situations, Flash Fill kicks in automatically as soon as Excel establishes a pattern in the data you are entering. If a preview does not show up, you can activate Flash Fill manually in this way:

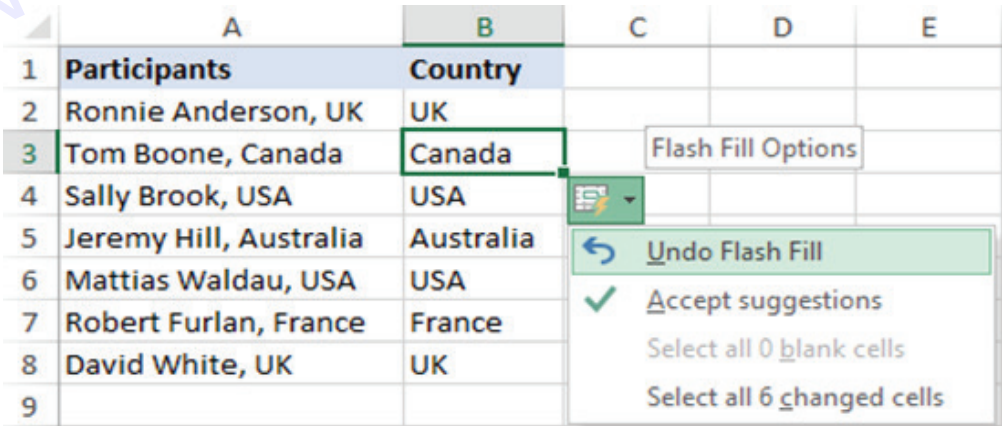
1 Fill in the first cell and press Enter.

2 Click the Flash Fill button on the Data tab or press the Ctrl + E shortcut.

Excel Flash Fill options

When using Flash Fill in Excel to automate data entry, the Flash Fill Options button appears near the auto-filled cells. Clicking this button opens the menu that lets you do the following:

- Undo the Flash Fill results.
- Select blank cells that Excel has failed to populate.
- Select the changed cells, for example, to format them all at once.



	A	B	C	D	E
1	Participants	Country			
2	Ronnie Anderson, UK	UK			
3	Tom Boone, Canada	Canada			
4	Sally Brook, USA	USA			
5	Jeremy Hill, Australia	Australia			
6	Mattias Waldau, USA	USA			
7	Robert Furlan, France	France			
8	David White, UK	UK			
9					

Excel Flash Fill examples

As already mentioned, Flash Fill is a very versatile tool. The below examples demonstrate some of its capabilities, but there is much more to it!

Extract text from cell (split columns)

Before Flash Fill came into existence, splitting the contents of one cell into several cells required the use of the **Text to Columns** feature or **Excel Text functions**. With Flash Fill, you can get the results instantly without intricate text manipulations.

Supposing you have a column of addresses and you want to extract zip codes into a separate column. Indicate your goal by typing the zip code in the first cell. As soon as Excel understands what you are trying to do, it fills in all the rows below the example with the extracted zip codes. You only need to hit Enter to accept them all.

	A	B
1	Address	Zip code
2	St-Joris Weert 3051 Belgium	3051
3	Illinois, 60606, USA	60606
4	California, 92618, USA	92618
5	Madrid 28014 Spain	28014
6	San Francisco, CA, 94105, USA	94105

Formulas to split cells and extract text

- Extract substring - formulas to extract text of a specific length or get a substring before or after a given character.
- Extract number from string - formulas to extract numbers from alphanumeric strings.
- Split names in Excel - formulas to extract first, last and middle names.

Extracting and splitting tools

- Split Text - a versatile tool to divide cell content into multiple columns or rows. Easily split string by any character, string, or mask.
- Split Names tool - fast and easy way to separate names in Excel.

Combine data from several cells (merge columns)

If you have an opposite task to perform, no problem, Flash Fill can concatenate cells too. Moreover, it can separate the combined values with a space, comma, semicolon or any other character - you just need to show Excel the required punctuation in the first cell:

	A	B	C	D
1	City	Country	Zip code	Address
2	Alden	USA	60001	Alden, USA, 60001
3	Los Angeles	USA	900001	Los Angeles, USA, 900001
4	Madrid	Spain	28014	Madrid, Spain, 28014
5	San Francisco	USA	94105	San Francisco, USA, 94105
6	St-Joris Weert	Belgium	3051	St-Joris Weert, Belgium, 305

Formulas to join cell values

- CONCATENATE function in Excel - formulas to combine text strings, cells and columns.

Merging tools

- Merge Tables Wizard - quick way to combine two tables by common columns.
- Merge Duplicates Wizard - Combine similar rows into one by key columns.

Clean data

If some data entries in your worksheet begin with a leading space, Flash Fill can get rid of them in a blink. Type the first value without a preceding space, and all extra spaces in other cells are gone too:

	A	B
1	Source data	Cleaned data
2	Alden	Alden
3	Los Angeles	Los Angeles
4	Madrid	Madrid
5	San Francisco	San Francisco
6	St-Joris Weert	St-Joris Weert

Formulas to clean data

- Excel TRIM function - formulas to remove excess spaces in Excel.

Data cleaning tools

- Trim Spaces - trim all leading, trailing and in-between spaces but a single space character between words.

Format text, numbers and dates

Quite often the data in your spread sheets is formatted in one way while you want it in another. Just start typing the values exactly as you want them to appear, and Flash Fill will do the rest.

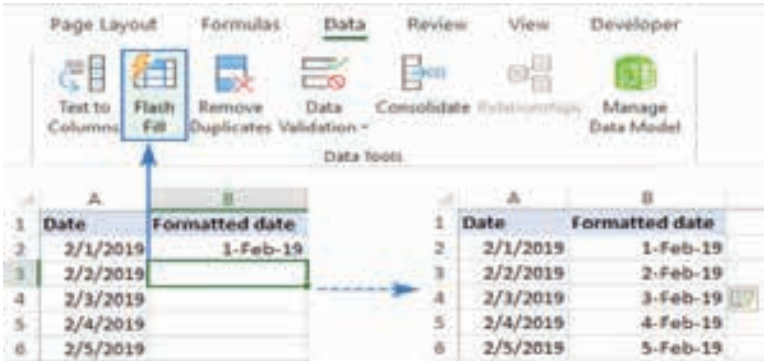
Perhaps you have a column of first and last names in lowercase. You wish the last and first names to be in proper case, separated with a comma. A piece of cake for Flash Fill.

	A	B
1	Name	Properly formatted name
2	ronnie anderson	Anderson, Ronnie
3	tom boone	Boone, Tom
4	sally brook	Brook, Sally
5	jeremy hill	Hill, Jeremy
6	mattias waldau	Waldau, Mattias
7	robert furlan	Furlan, Robert
8	david white	White, David

Maybe you are working with a column of numbers that need to be formatted as phone numbers. The task can be accomplished by using a predefined **Special format** or creating a c**ustom number format**. Or you can do it an easy way with Flash Fill.

	A	B
1	Number	Phone number
2	1234567891	123-456-7891
3	2345678912	234-567-8912
4	3456789123	345-678-9123
5	4567891234	456-789-1234
6	5678912345	567-891-2345

To re-format the dates to your liking, you can apply the corresponding **Date format** or type a properly formatted date into the first cell. Oops, no suggestions have appeared… What if we press the Flash Fill shortcut **(Ctrl + E)** or click its button on the ribbon? Yep, it works beautifully!



Replace part of cell contents

Replacing part of a string with some other text is a very common operation in Excel, which Flash Fill can also automate.

Let's say, you have a column of social security numbers and you want to censor this sensitive information by replacing the last 4 digits with **XXXX**.

To have it done, either use the **REPLACE** function or type the desired value in the first cell and let Flash Fill auto fill the remaining cells.

	A	B
1	Social Security Number	Censored data
2	123-45-6789	123-45-XXXX
3	012-34-6789	012-34-XXXX
4	234-56-7891	234-56-XXXX
5	234-56-7890	234-56-XXXX
6	456-78-9012	456-78-XXXX

Advanced combinations

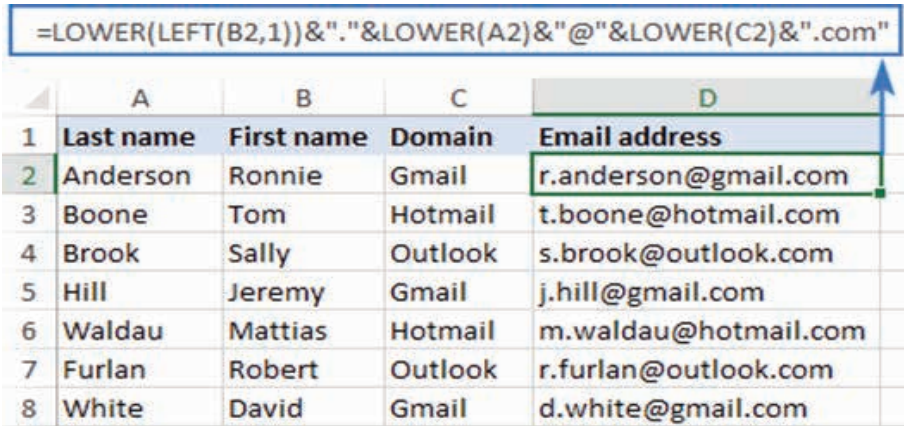
Flash Fill in Excel can accomplish not only straightforward tasks like demonstrated in the above examples but also perform more sophisticated data re-arrangements.

As an example, let's combine different pieces of information from 3 columns and add a few custom characters to the result.

Supposing, you have first names in column A, last names in column B, and domain names in column C. Based on this information, you want to generate email addresses in this format: initial.surname@domain.com.

For experienced Excel users, there is no problem to extract the initial with the LEFT function, convert all the characters to lowercase with the LOWER function and concatenate all the pieces by using the **concatenation operator.**

```
=LOWER(LEFT(B2,1))&"."&LOWER(A2)&"@"&LOWER(C2)&".com"
```



But can Excel Flash Fill create these email addresses for us automatically? Sure thing!

	A	B	C	D
1	Last name	First name	Domain	Email address
2	Anderson	Ronnie	Gmail	r.anderson@gmail.com
3	Boone	Tom	Hotmail	t.boone@hotmail.com
4	Brook	Sally	Outlook	s.brook@outlook.com
5	Hill	Jeremy	Gmail	j.hill@gmail.com
6	Waldau	Mattias	Hotmail	m.waldau@hotmail.com
7	Furlan	Robert	Outlook	r.furlan@outlook.com
8	White	David	Gmail	d.white@gmail.com

Excel Flash Fill limitations and caveats

Flash Fill is an awesome tool, but it does have a couple of limitations that you should be aware of before you start using this feature on your real data sets.

1 Flash Fill results do not update automatically

Unlike formulas, the results of Flash Fill are static. If you make any changes to the original data, they won't be reflected in the Flash Fill results.

2 May fail to identify a pattern

In some situations, especially when your original data are arranged or formatted differently, Flash Fill may stumble and produce incorrect results.

For example, if you use Flash Fill to extract middle names from the list where some entries contain only the First and Last names, the results for those cells will be wrong. So, it is wise to always review the Flash Fill output.

	A	B
1	Fll name	Middle Name
2	Ronnie D. Anderson	D.
3	Tom Andrew Boone	Andrew
4	Sally Ann Brook	Ann
5	Jeremy Hill	Jeremy Hi ← Wrong result
6	Robert M. Furlan	M.

3 Ignores cells with non-printable characters

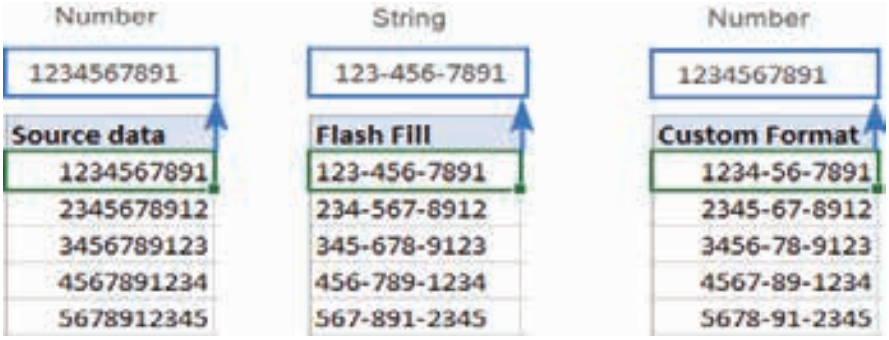
If some of the cells to be auto-filled contain spaces or other non-printable characters, Flash Fill will skip such cells.

	A	B
1	Address	Zip code
2	St-Joris Weert 3051 Belgium	3051
3	Illinois, 60606, USA	60606
4	California, 92618, USA	← There is a non-printable character in this cell
5	Madrid 28014 Spain	28014
6	San Francisco, CA, 94105, USA	94105

So, if any of the resulting cells are blank, clear those cells (Home tab >Formats group > Clear > Clear All) and run Flash Fill again.

4 May convert numbers to strings

When using Flash Fill for reformatting numbers, please be aware that it may convert your numbers to alphanumeric strings. If you prefer to keep the numbers, use the capabilities of Excel format that changes only the visual representation, but not the underlying values.



How to turn Flash Fill on and off

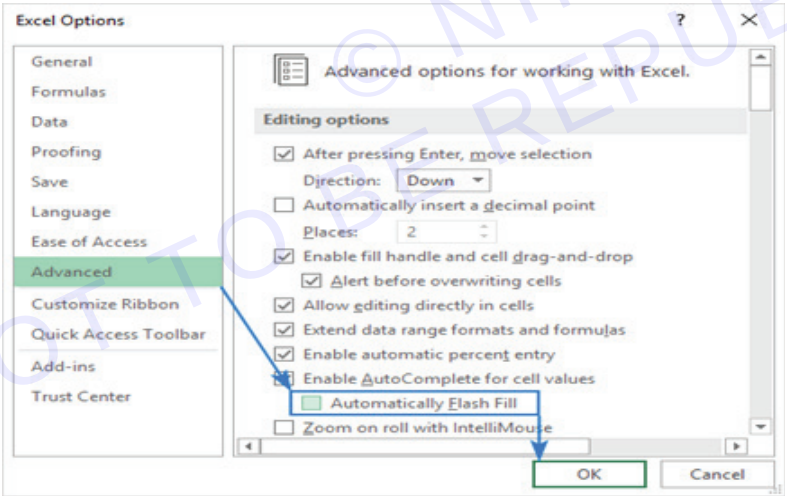
Flash Fill in Excel is turned on by default. If you do not want any suggestions or automatic changes in your worksheets, you can disable Flash Fill in this way:

1 In your Excel, go to File> Options.

2 On the left panel, click Advanced.

3 Under Editing options, clear the Automatically Flash Fill box.

4 Click OK to save the changes.



To re-enable Flash Fill, simply select this box again.

Excel Flash Fill not working

In most cases, Flash Fill works without a hitch. When it falters, the below error may show up, and the following tips will help you get it fixed.

1 Provide more examples

Flash Fill learns by example. If it is unable to recognize a pattern in your data, fill in a couple more cells manually, so that Excel could try out different patterns and find the one best suited for your needs.

2 Force it to run

If Flash Fill suggestions do not appear automatically as you type, try to run it manually.

3 Make sure Flash Fill is enabled

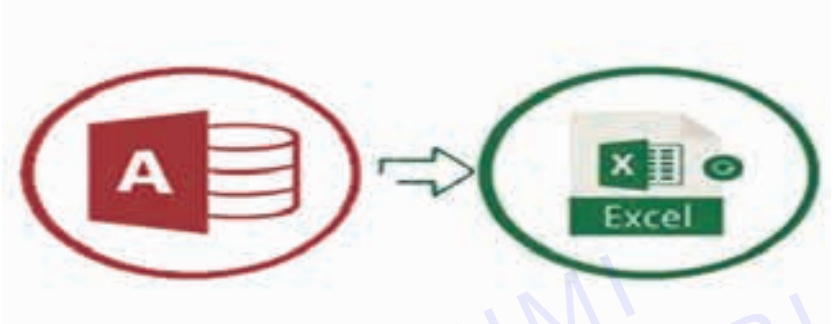
If it does not start either automatically or manually, check if the Flash Fill functionality is turned on in your Excel.

4 Flash Fill error persists

If none of the above suggestions has worked and Excel Flash Fill still throws an error, there is nothing else you can do but enter the data manually or with formulas.

That's how you use Flash Fill in Excel. I thank you for reading and hope to see you on our blog next week!

Database access from MS Excel



"Microsoft Access is an excellent general-purpose database management system whereas Excel is a spread sheet package, with lots of extra features thrown in."

MS Excel vs Access Comparison

MS Excel and Access are both Microsoft Office products. Microsoft Access, from the Office suite, is an excellent general-purpose database management system made for storing and analysing data.

Excel on the other hand is a spread sheet package, with lots of extra features thrown in which allow you to do things like creating pivot tables for summarising data. Using Excel as a database is great for small businesses and database beginners.

What Is MS Excel Used For?

Many people use Excel in business for a range of tasks from simple admin to full-scale automation. Thanks to its easy to use features Excel is very flexible and allows you to summarise data quickly without having to do anything too complicated.

Some of the uses for Excel include:

- Tabulating and analysing complex numerical data
- Creating tables and forms
- Writing and running macros to automate tasks
- General day-to-day data management (e.g., creating reports, charts, etc.)
- Store and manipulate audio and video files

Excel is very intuitive and easy to use, which allows most people to learn it quickly. If you're job hunting, including Excel skills on your CV can help you out a lot!

What Is An MS Access Database Used For?

MS Access is a database management system that gives you the ability to store your data in a relational database. This means that you can have multiple tables, linked together to form a relationship, which is all stored in one file.

The advantage of this is that if you update data in one table, it automatically updates all related records. This eliminates a lot of time-consuming admin and manual errors.

Access is great for

- Storing and managing large amounts of data
- Running reports that allow for grouping and sorting data
- Managing relational databases (e.g., adding and deleting tables, etc.)

The main disadvantage of Access is that it can be quite confusing to set up a database the first time. This means you should work with an MS Access Expert to help set it up and ensure everything is running smoothly.

"Excel and Access share some traits but are ultimately different tools. Comparing them is a useful way to highlight the key differences and understand where the strengths and weaknesses are."

Key Differences between Excel vs. Access

Excel and Access share some traits but are ultimately different tools. Comparing Excel vs. Access is a useful way to highlight the key differences between them and understand where the strengths and weaknesses are for both products.

Excel Do's and Don'ts: How to Make Your Spread sheets shine!

Excel vs. Access – Data Visualisation

Excel allows you to add graphics such as charts, pictures, etc. to your worksheets. You can also link these pictures or graphics to your data so they automatically update when the data updates. This is great if you have a report that requires data to be displayed in the form of a chart.

Access allows you to add custom forms when creating a database. These forms can take multiple pieces of information from a user and display them in an organised fashion, so they are easy to read and navigate.

You can also add buttons or links on these forms so your users can navigate to the information they require. This is useful if you need your users to provide data on a form or screen that they must print and sign, etc.

Excel vs. Access – Automation

Excel allows you to add macros so you can automate Excel tasks such as deleting reports once they have been printed and saving files on a regular basis. These are useful if your company requires the same routine tasks to be performed on various data files on a regular basis.

On the other hand, Microsoft Access allows you to create modules in which you can write VBA (Visual Basic for Applications) code. This allows you to automate certain tasks, such as inserting new records into a table or deleting old records. This is great if your company needs a certain set of tasks to be completed on a regular basis.

Excel vs. Access – Data Security

Excel does not allow you to add security measures to your files, so anyone with access will be able to view or edit the data. You can lock or protect individual worksheets with user-level security features, which might be useful if you need to restrict access to certain worksheets.

Access allows you to control who has access to your database files and what they can do with them. You can create usernames for multiple users so different people have different levels of access, depending on which username they use.

This is great if you want some users (e.g. managers) to update data, while others (e.g. MS Access consultants) are only allowed to view the data and access data when needed.

Excel vs. Access – Data Analysis

Excel allows you to perform basic data analysis by adding formulas and performing certain functions on written values.

Access has specific tools that are designed for analysing and summarising your data. These include PivotTables, which allow you to create tables that summarise the data in various ways (e.g., number of sales per region, etc.). This is great if you want to quickly analyse large amounts of data**.**

"Because Access is a professional database programme, it has some significant benefits over Microsoft Excel when building and using databases. Here are five reasons to use Access vs. Excel."

5 Reasons to Use Microsoft Access vs. Excel

Because Access is a professional database programme, it has some significant benefits over Microsoft Excel when building and using databases. Here are five reasons to use Access vs. Excel.

1 User Security Model

Access provides all the tools needed to set access permissions on objects such as tables and queries. The security model allows you to grant or deny certain users (e.g., managers) rights to edit or update data while restricting other users (e.g., consultants) so they are only allowed to view data in pre-defined ways.

2 Data Storage Capacity

Access is superior to Excel when it comes to the amount of data you can store in a database. This is because Access is specifically designed for storing large amounts of data, whereas Excel was not.

3 Maintaining Data Integrity

Access allows you to define field types and limits, which enforces appropriate data entry and prevents users from unintentionally inserting invalid values into fields.

For example, if your database requires a specific data format, Access can enforce this by defining the field as a Date type and setting an appropriate limit (e.g., only valid dates). The same applies to numerical values, currency types or text containing specific characters.

4 Data Analysis

Access provides various tools that are specifically designed for performing data analysis on large data sets. Use Pivot Tables to create tables summarising the data in multiple ways quickly (e.g., number of sales per region). This way, you don't have to enter all the data manually to find the information you need.

5 Summarising Data

Visual Basic for Applications (VBA) allows you to write code to execute specific tasks regularly. This makes it very easy to bring your database up-to-date by adding new records or deleting old ones. The VBA capability allows you access to the tables and fields within your database, which means that you can automate tasks like importing new records and exporting data to other applications.

"While Access is better suited to being used as a database, Excel can still be helpful too."

5 Examples Where an Excel Database Is Useful

While Access is better suited to being used as a database, Excel can still be helpful too. Here are three examples where an Excel database might be a good idea:

1 Phone Book

If you have a small phone book, using Microsoft Excel may be sufficient to store contact details. However, if you have a larger phone book, you could create a copy of the contacts in Excel and convert this into a database to make it easier to sort, update and retrieve information. In this case, you can utilise the standard functionality in Excel to create and manage your contacts.

2 Product Information Management (PIM)

Microsoft Excel's ability to create and update records makes it very useful for storing PIM data. The spread sheet can be set up with one column per product and another column for the price.

One of the most important fields is probably stock, which specifies how many products you have in stock. You can then quickly update this when you receive new products from suppliers or sell existing ones to customers.

3 Budget Planning

You may want to use a spread sheet in your budget planning because it makes it easy for you to create and manage several budgets.

Simply enter the budget items for each month, the beginning balance, create a formula to calculate the year-to-date totals and then sum up all of these values at the end.

4 Inventory Management

One of Excel's strengths is its ability to insert large amounts of data quickly. This can be useful if you need to keep track of thousands or even millions of products.

For instance, you could create a spread sheet where one row is created per product and include the price in these rows. This way, if you add new products, update prices or remove items from stock, you can simply enter the new values into Excel and then write a macro to update your inventory system automatically.

5 Sales & Order Tracking

Another way to use Excel as a database is by setting up one sheet per order. This allows you to create and update records easily and quickly view information such as the customer name and address, the list of items ordered their prices, and quantities.

"Excel and Access have their advantages and disadvantages. It really depends on what you want to do with them and how much knowledge you have on the product before deciding which is right for you."

Choosing Microsoft Access vs. Excel for Your Business

Both of these software packages have their advantages and disadvantages. It really depends on what you want to do with them and how much knowledge you have on the product before deciding which is right for you.

If you need to use a database regularly and need to automate tasks, Microsoft Access may be a better choice as it allows you to write VBA code. If you only need the software occasionally or want a simple solution for managing data, then Excel is probably more suited for your needs.

If you need help with databases in Excel or Microsoft Access contact our database experts today. We've worked with businesses across multiple industries to create professional databases solutions for millions of data records.

What if Analysis-Goalseek, solver & Scenerios

Introduction to What-If Analysis

By using What-If Analysis tools in Excel, you can use several different sets of values in one or more formulas to explore all the various results.

For example, you can do what-If Analysis to build two budgets that each assumes a certain level of revenue. Or, you can specify a result that you want a formula to produce, and then determine what sets of values will produce that result. Excel provides several different tools to help you perform the type of analysis that fits your needs.

How to use Goal Seek in Excel for What-If analysis

What-If Analysis is one of the most powerful Excel features and one of the least understood. In most general terms, What-If Analysis allows you to test out various scenarios and determine a range of possible outcomes. In other words, it enables you to see the impact of making a certain change without changing the real data. In this particular tutorial, we will focus on one of Excel's What-If Analysis tools - Goal Seek.

What is Goal Seek in Excel?

Goal Seek is Excel's built-in What-If Analysis tool that shows how one value in a formula impacts another. More precisely, it determines what value you should enter in an input cell to get the desired result in a formula cell.

The best thing about Excel Goal Seek is that it performs all calculations behind the scenes, and you are only asked to specify these three parameters:

- Formula cell
- Target/desired value
- The cell to change in order to achieve the target

The Goal Seek tool is especially useful for doing sensitivity analysis in financial modelling and is widely used by management majors and business owner. But there are many other uses that may prove helpful to you.

For instance, Goal Seek can tell you how much sales you have to make in a certain period to reach $100,000 annual net profit (example 1). Or, what score you must achieve for your last exam to receive an overall passing score of 70% (example 2). Or, how many votes you need to get in order to win the election (example 3).

On the whole, whenever you want a formula to return a specific result but are not sure what input value within the formula to adjust to get that result, stop guessing and use the Excel Goal Seek function!

Note. Goal Seek can process only one input value at a time. If you are working on an advanced business model with multiple input values, use the Solver add-in to find the optimal solution.

How to use Goal Seek in Excel

The purpose of this section is to walk you through how to use the Goal Seek function. So, we'll be working with a very simple data set:

Data set

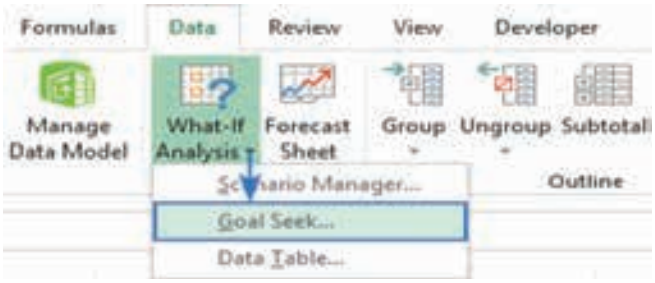
	A	B	C
1		Goal Seek	
2	Item price	$5	
3	Qty.	100	Variable
4	Commission	10%	
5	Revenue	$450	=B2*B3*(1-B4)

The above table indicates that if you sell 100 items at $5 each, minus the 10% commission, you will make $450. The question is: How many items do you have to sell to make $1,000?

Let's see how to find the answer with Goal Seek

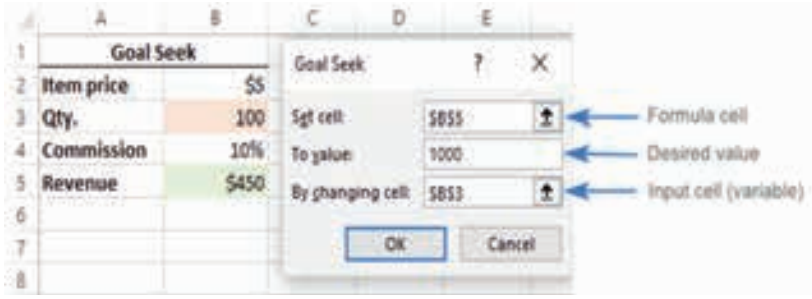
1 Set up your data so that you have a formula cell and a changing cell dependent on the formula cell.

2 Go to the Data tab > Forecast group, click the what if Analysis button, and select Goal Seek…



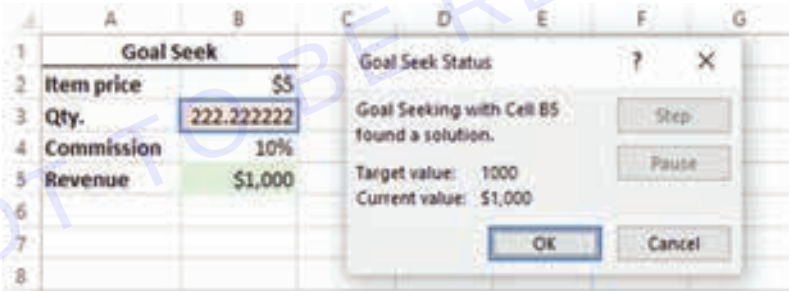
3 In the Goal Seek dialog box, define the cells/values to test and click OK

- Set cell - the reference to the cell containing the formula (B5).
- To value - the formula result you are trying to achieve (1000).
- By changing cell - the reference for the input cell that you want to adjust (B3).



4 The Goal Seek Status dialog box will appear and let you know if a solution has been found. If it succeeded, the value in the "changing cell" will be replaced with a new one. Click OK to keep the new value or Cancel to restore the original one.

In this example, Goal Seek has found that 223 items (rounded up to the next integer) need to be sold to achieve revenue of $1,000.



If you are not sure you will be able to sell that many items, then maybe you can reach the target revenue by changing the item price? To test this scenario, do Goal Seek analysis exactly as described above except that you specify different Changing cell (B2).

As the result, you will find out that if you increase the unit price to $11, you can reach $1,000 revenue by selling only 100 items.

Tips and notes

- Excel Goal Seek does not change the formula, it only changes the input value that you supply to the By changing cell box.
- If Goal Seek is not able to find the solution, it displays the closest value it has come up with.
- You can restore the original input value by clicking the Undo button or pressing the Undo shortcut (Ctrl + Z).

Examples of using Goal Seek in Excel

Below you will find a few more examples of using the Goal Seek function in Excel. The complexity of your business model does not really matter as long as your formula in the Set cell depends on the value in the Changing cell, directly or through intermediate formulas in other cells.

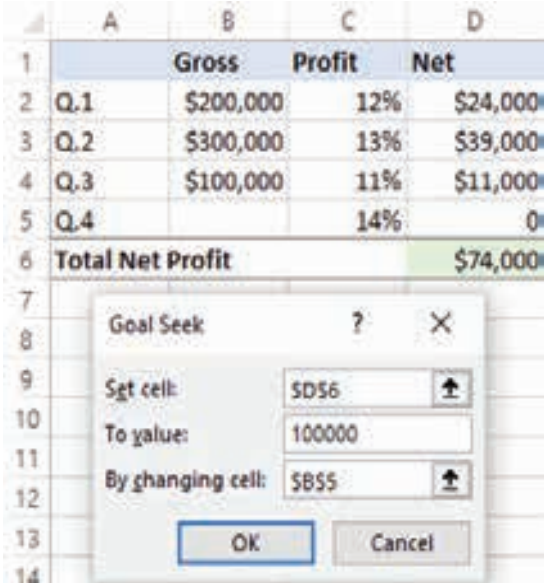
Example 1: Reach the profit goal

Problem: It is a typical business situation - you have the sales figures for the first 3 quarters and you want to know how much sales you have to make in the last quarter to achieve the target net profit for the year, say, $100,000.

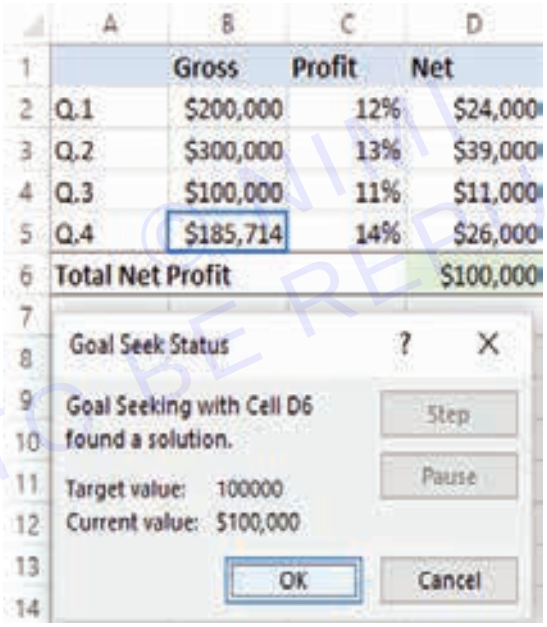
	A	B	C	D	E
1		Gross	Profit	Net	
2	Q.1	$200,000	12%	$24,000	=B2*C2
3	Q.2	$300,000	13%	$39,000	=B3*C3
4	Q.3	$100,000	11%	$11,000	=B4*C4
5	Q.4		14%	0	=B5*C5
6	Total Net Profit			$74,000	=SUM(D2:D5)

Solution: With the source data organized like shown in the screenshot above, set up the following parameters for the Goal Seek function.

- Set cell - the formula that calculates the total net profit (D6).
- To value - the formula result you are looking for ($100,000).
- By changing cell - the cell to contain the gross revenue for quarter 4 (B5).



Result: The Goal Seek analysis shows that in order to obtain $100,000 annual net profit, your fourth-quarter revenue must be $185,714.



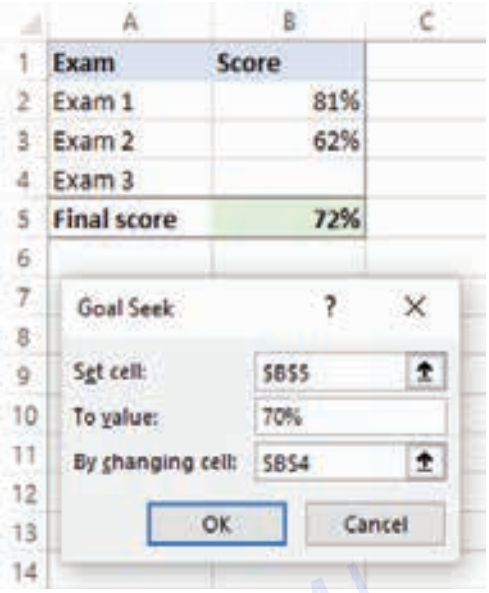
Example 2: Determine the exam passing score

Problem: At the end of the course, a student takes 3 exams. The passing score is 70%. All the exams have the same weight, so the overall score is calculated by averaging the 3 scores. The student has already taken 2 out of 3 exams. The question is: What score does the student need to get for the third exam to pass the entire course?

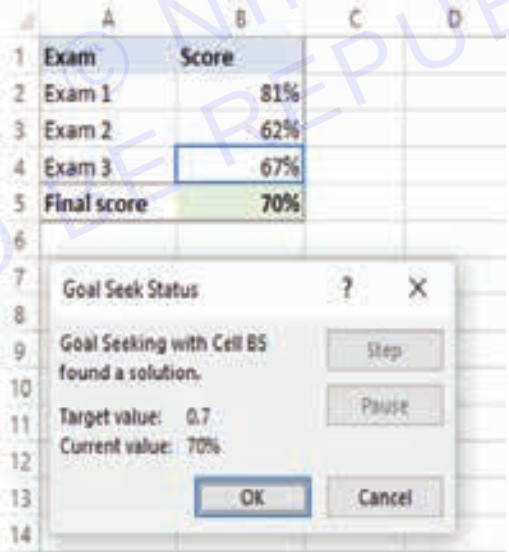
	A	B	C	D
1	Exam	Score		
2	Exam 1	81%		
3	Exam 2	62%		
4	Exam 3			
5	Final score	72%	=AVERAGE(B2:B4)	

Solution: Let's do Goal Seek to determine the minimum score on exam 3:

- Cell - the formula that averages the scores of the 3 exams (B5).
- To value - the passing score (70%).
- By changing cell - the 3rd Set exam score (B4).



Result: In order get the desired overall score; the student must achieve a minimum of 67% on the last exam.



Example 3: What-If analysis of the election

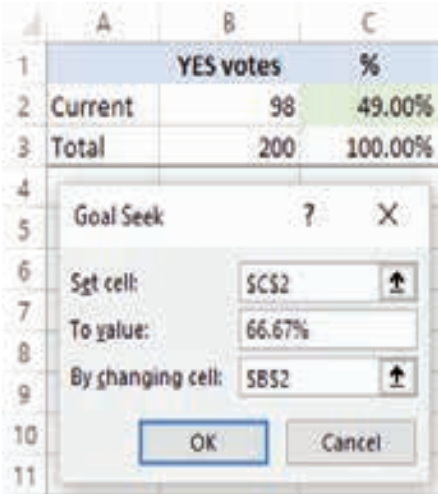
Problem: You are running for some elected position where a two-thirds majority (66.67% of votes) is required to win the election. Assuming there are 200 total voting members, how many votes do you have to secure?

Currently, you have 98 votes, which is quite good but not sufficient because it only makes 49% of the total voters.

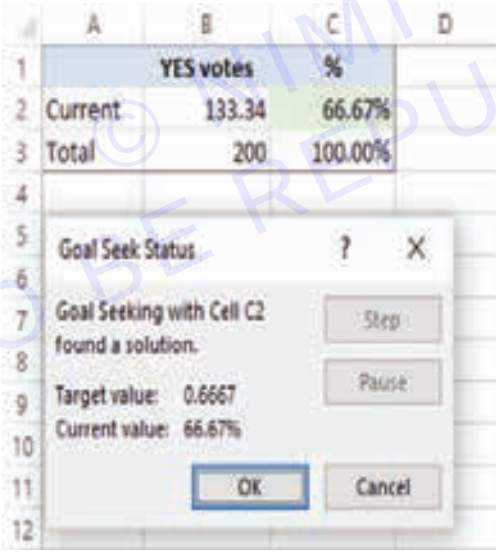
	A	B	C	D
1		YES votes	%	
2	Current	98	49.00%	=B2/B3
3	Total	200	100.00%	

Solution: Use Goal Seek to find out the minimum number of "Yes" votes you need to get:

- Set cell - the formula that calculates the percentage of the current "Yes" votes (C2).
- To value - the required percentage of "Yes" votes (66.67%).
- By changing cell - the number of "Yes" votes (B2).

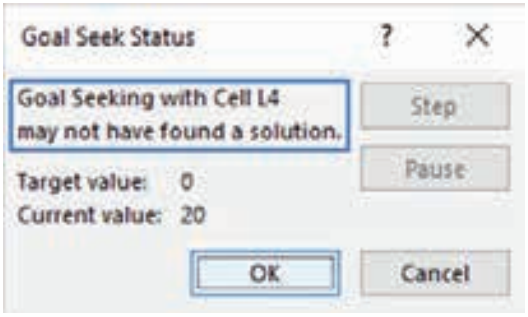


Result: What-If analysis with Goal Seek shows that to achieve the two-thirds mark or 66.67%, you need 133 "yes" votes.



Excel Goal Seek not working

Sometimes Goal Seek is not able to find a solution simply because it does not exist. In such situations, Excel will get the closest value and inform you that Goal Seeking may not have found a solution.



If you are certain that a solution to the formula you are trying to resolve does exist, check out the following troubleshooting tips.

1 Double check Goal Seek parameters

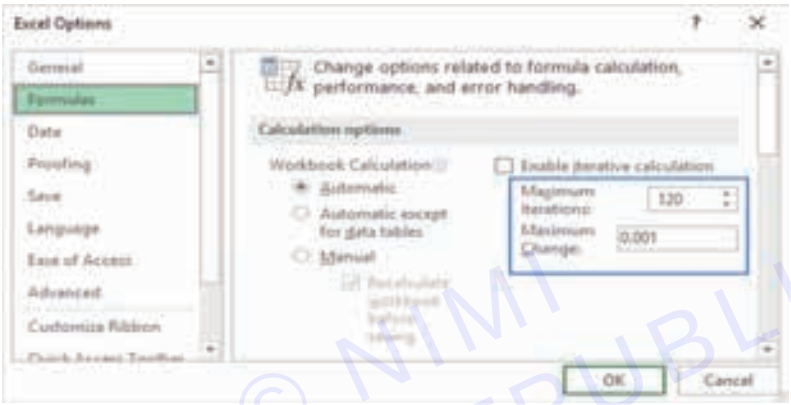
First off, make sure the Set cell refers to the cell containing a formula, and then, check if the formula cell depends, directly or indirectly, on the changing cell.

2 Adjust iteration settings

In your Excel, click File > Options > Formulas and change these options:

- Maximum Iterations - increase this number if you want Excel to test more possible solutions.
- Maximum Change - decrease this number if your formula requires more accuracy. For example, if you are testing a formula with an input cell equal to 0 but Goal Seek stops at 0.001, setting Maximum Change to 0.0001 should fix the issue.

The below screenshot shows the default iteration settings



3 No circular references

For Goal Seek (or any Excel formula) to work properly, the involved formulas should not be co-dependent on each other, i.e. there should be no circular references.

That's how you perform What-If analysis in Excel with the Goal Seek tool. I thank you for reading and hope to see you on our blog next week!

Macros & VBA

How to create, change, copy and delete macros

This tutorial will set you on your way to learning Excel macros. You will find how to record a macro and insert VBA code in Excel, copy macros from one workbook to another, enable and disable them, view the code, make changes, and a lot more.

For Excel newbies, the concept of macros often looks insurmountable. Indeed, it may take months or even years of training to master VBA. However, this does not mean you cannot take advantage of the automation power of Excel macros right away. Even if you are a complete novice in VBA programming, you can easily record a macro to automate some of your repetitive tasks.

This article your entry point to the fascinating world of Excel macros. It covers the essential basics that you need to know to get started and provides links to the related in-depth tutorials.

What are macros in Excel?

Excel macro is a set of commands or instructions stored in a workbook in the form of VBA code. You can think of it as a small program to perform a predefined sequence of actions. Once created, macros can be re-used anytime. Running a macro executes the commands it contains.

Typically, macros are used to automate repetitive tasks and daily routines. Skilled VBA developers can write really sophisticated macros that go well beyond reducing the number of keystrokes.

Quite often, you may hear people referring to a "macro" as "VBA". Technically, there is a distinction: a macro is a piece of code while Visual Basic for Applications (VBA) is the programming language created by Microsoft to write macros.

Why use Excel macros?

The main purpose of macros is to have more work done in less time. Like you use formulas to crunch numbers and manipulate text strings, you can use macros to perform frequent tasks automatically.

Let's say, you are to create a weekly report for your supervisor. For this, you import various analytics data from a couple or more external resources. The problem is that those data are messy, superfluous, or not in the format that Excel can understand. That means you need to reformat dates and numbers, trim extra spaces and delete blanks, copy and paste information into appropriate columns, build charts to visualize trends, and do a lot more different things to make your report clear and user-friendly. Now, imaging that all these operations can be performed for you instantly in a mouse click!

Of course, building a complex macro takes time. Sometimes, it can take even more time than performing the same manipulations manually. But creating a macro is a onetime set-up. Once written, debugged and tested, VBA code will do the job quickly and flawlessly, minimizing human errors and costly mistakes.

How to create a macro in Excel

There are two ways to create macros in Excel - by using the Macro Recorder and Visual Basic Editor.

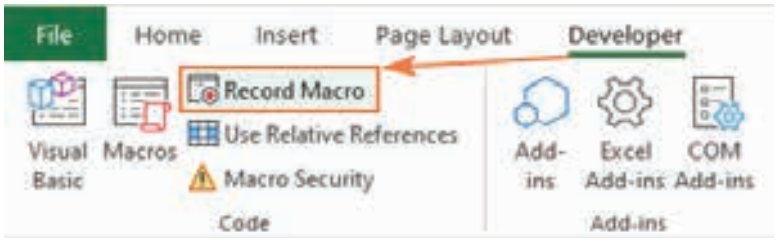
Tip. Within Excel, most operations with macros are done via the Developer tab, so be sure to add Developer tab to your Excel ribbon.

Recording a macro

Even if you don't know anything about programing in general and VBA in particular, you can easily automate some of your work just by letting Excel record your actions as a macro. While you are performing the steps, Excel closely watches and writes down your mouse clicks and keystrokes in the VBA language.

The Macro Recorder captures nearly everything that you do and produces a very detailed (often redundant) code. After you've stopped the recording and saved the macro, you can view its code in the Visual Basic Editor and make small changes. When you run the macro, Excel goes back to the recorded VBA code and executes the exact same moves.

To start recording, click the Record Macro button on either the Developer tab or the Status bar.



Writing a macro in Visual Basic Editor

The Visual Basic for Applications (VBA) Editor is the place where Microsoft Excel keeps the code of all macros, both recorded and written manually.

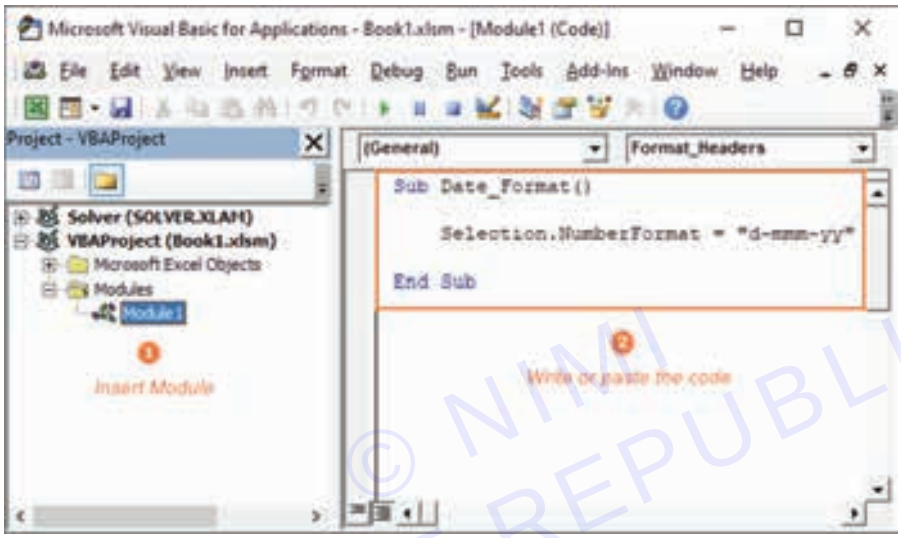
In the VBA Editor, you can not only program a sequence of actions, but also create custom functions, display your own dialog boxes, evaluate various conditions, and most importantly code the logic! Naturally, creating your own macro requires some knowledge of the structure and syntax of the VBA language, which is beyond the scope of this tutorial for beginners. But there is nothing that would prevent you from reusing someone else's code (say, the one you've found on our blog :) and even a complete novice in Excel VBA should have no difficulties with that!

First, press Alt + F11 to open the Visual Basic Editor. And then, insert the code in these two quick steps:

1 In the Project Explorer on the left, right-click the target workbook, and then click Insert > Module.

2 In the Code window on the right, paste the VBA code.

When done, press F5 to run the macro.



How to run macros in Excel

There are several ways to start a macro in Excel:

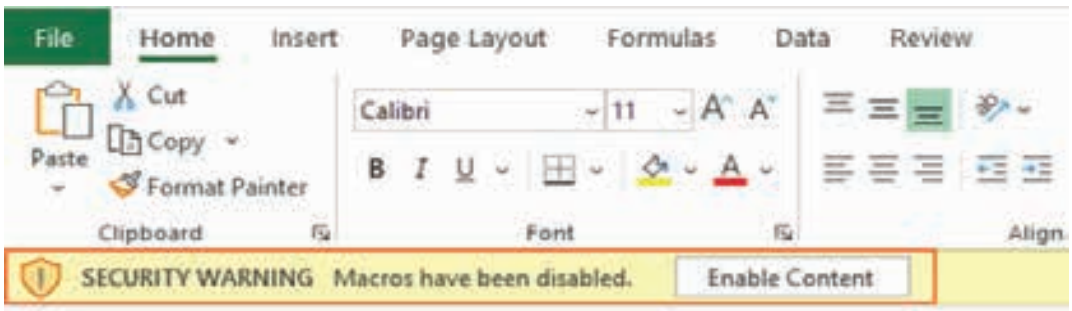
- To run a macro from a worksheet, click the Macros button on the Developer tab or press the Alt + F8 shortcut.
- To run a macro from the VBA Editor, press either:
- F5 to run the entire code.
- F8 to go through the code line-by-line. This is very useful for testing and troubleshooting.

Additionally, you can launch a macro by clicking a custom button or pressing the assigned shortcut.

How to enable macros in Excel

Because of security reasons, all macros in Excel are disabled by default. So, to use the magic of VBA codes to your advantage, you need to know how to enable them.

The easiest way to turn on macros for a specific workbook is to click the Enable Content button in the yellow security warning bar that appears at the top of the sheet when you first open a workbook with macros.



How to change macro settings

Microsoft Excel determines whether to allow or disallow VBA codes to be executed in your workbooks based on the macro setting selected in the Trust Center.

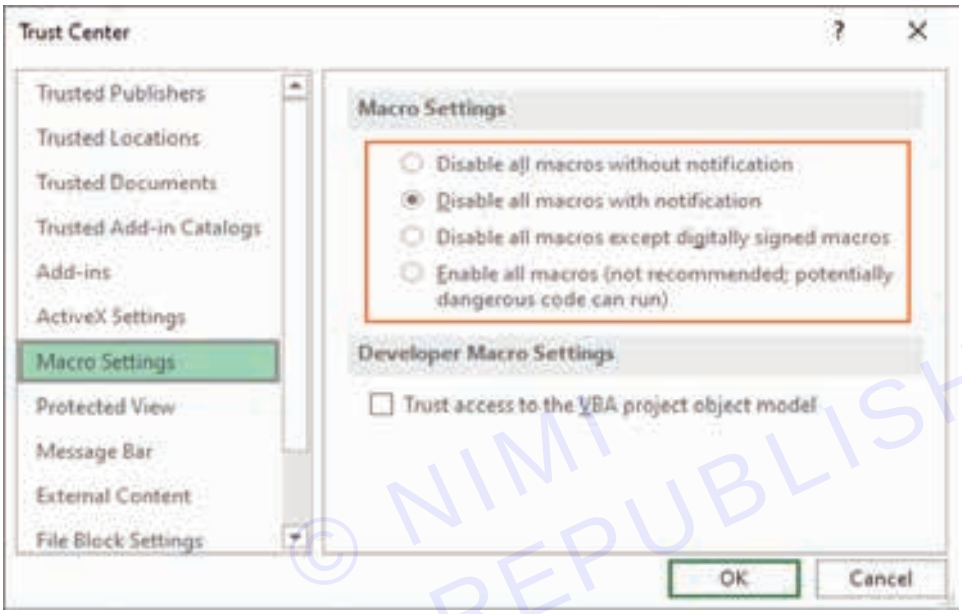
Here are the steps to access the Excel macro settings and change them if needed:

1 Go to the File tab and select Options.

2 On the left-side pane, select Trust Center, and then click Trust Center Settings.

3 In the Trust Center dialog box, click Macro Settings on the left, select the desired option, and click OK.

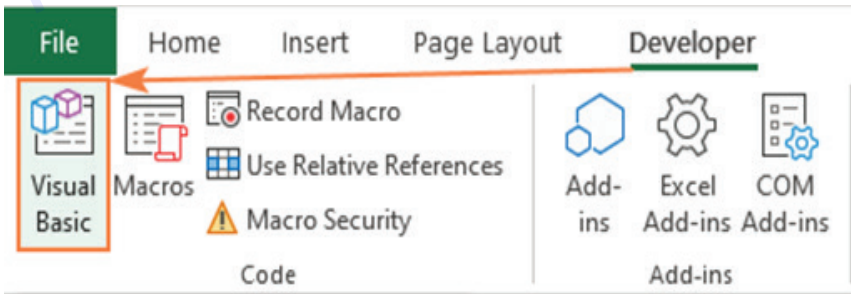
In the screenshot below, the default macro setting is selected.



How to view, edit and debug VBA codes in Excel

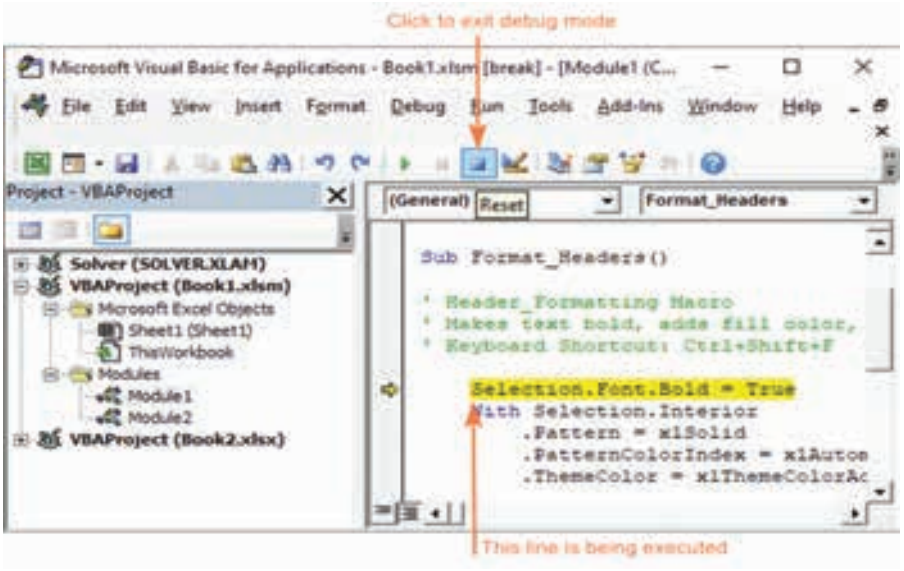
Any changes to the code of a macro, whether it's generated automatically by the Excel macro recorder or written by you, are made in the Visual Basic Editor.

To open the VB Editor, either press Alt + F11 or click the Visual Basic button on the Developer tab.



To view and edit the code of a specific macro, in the Project Explorer on the left, double-click the module that contains it, or right-click the module and pick View Code. This opens the Code window where you can edit the code.

To test and debug a macro, use the F8 key. This will take you through the macro code line-by-line letting you see the effect that each line has on your worksheet. The line currently being executed is highlighted in yellow. To exit debug mode, click the Reset button on the toolbar (blue square).



How to copy a macro to another workbook

You created a macro in one workbook and now want to reuse it in other files too? There are two ways to copy a macro in Excel.

Copy the module containing a macro

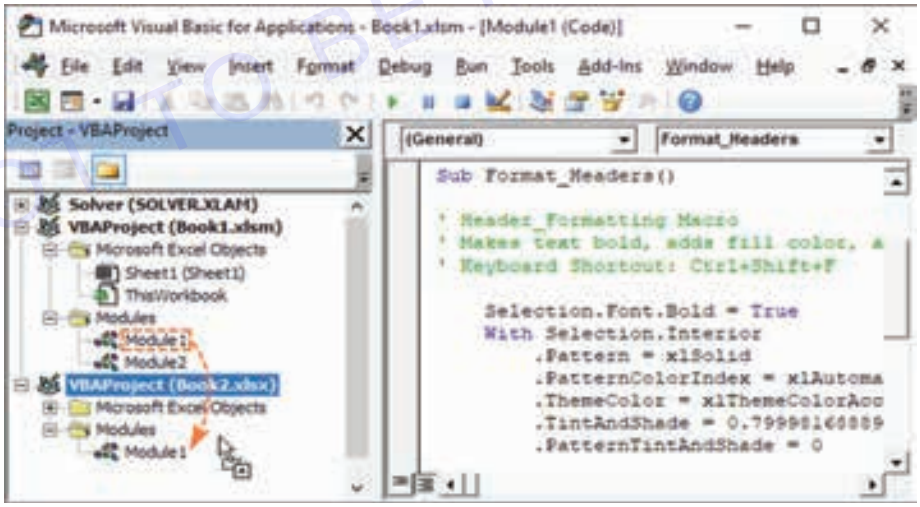
In case the target macro resides in a separate module or all the macros in the module are useful for you, then it makes sense to copy the whole module from one workbook to another:

1 Open both workbooks - the one that contains the macro and the one where you wish to copy it.

2 Open the Visual Basic Editor.

3 In the Project Explorer pane, find the module containing the macro and drag it to the destination workbook.

In the screenshot below, we are copying Module1 from Book1 to Book 2.



Copy the source code of a macro

If the module contains many different macros while you need just one, then copy only the code of that specific macro. Here's how:

1 Open both workbooks.

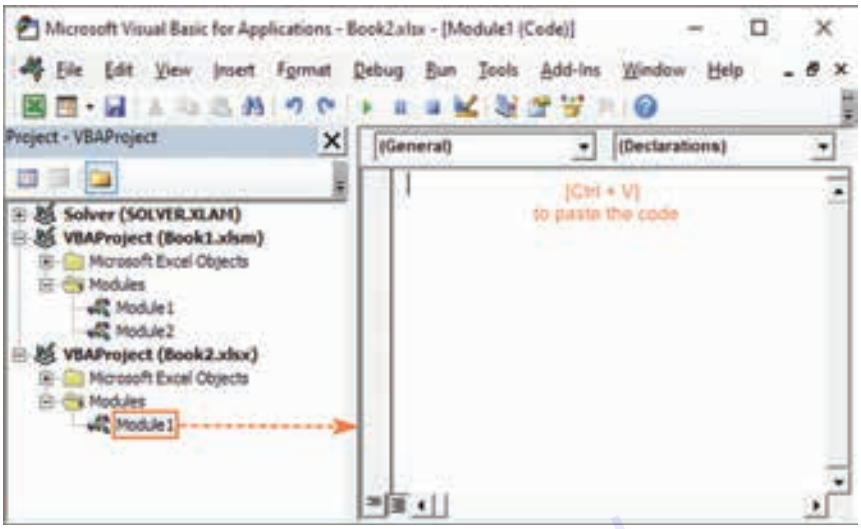
2 Open the Visual Basic Editor.

3 In the Project Explorer pane, double-click the module containing the macro that you'd like to copy to open its Code window.

4 In the Code window, find the target macro, select its code (beginning with Sub and ending with End Sub) and press Ctrl + C to copy it.

5 In the Project Explorer, find the destination workbook, and then either insert a new module into it (right-click the workbook and click Insert > Module) or double-click an existing module to open its Code window.

6 In the Code window of the destination module, press Ctrl + V to paste the code. If the module already contains some code, scroll down to the last code line, and then paste the copied macro.



How to delete macros in Excel

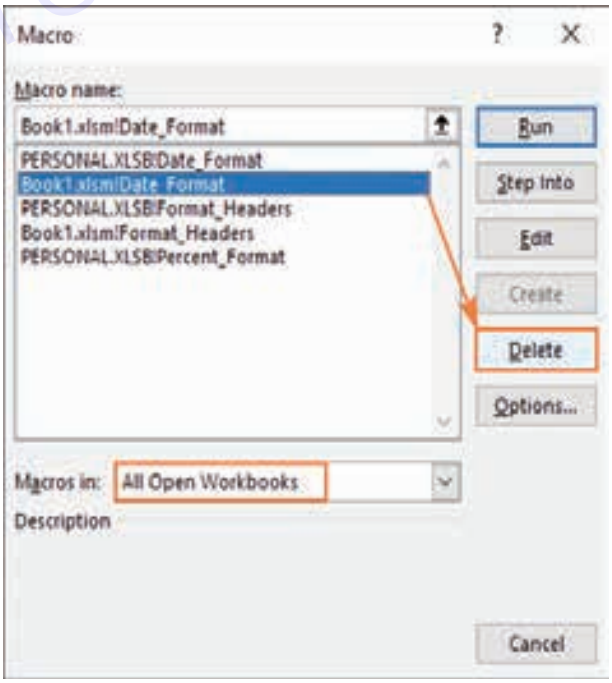
If you no longer need a certain VBA code, you can delete it by using the Macro dialog box or the Visual Basic Editor.

Deleting a macro from a workbook

To delete a macro directly from your Excel workbook, carry out these steps:

1 On the Developer tab, in the Code group, click the Macros button or press the Alt + F8 shortcut.

2 In the Macro dialog box, select the macro you want to remove and click Delete.



Tips

- To view all the macros in all open files, select All Open Workbooks from the Macros in drop-down list.
- To be able delete a macro in the Personal Macro Workbook, you need to unhide Personal.xlsb first.

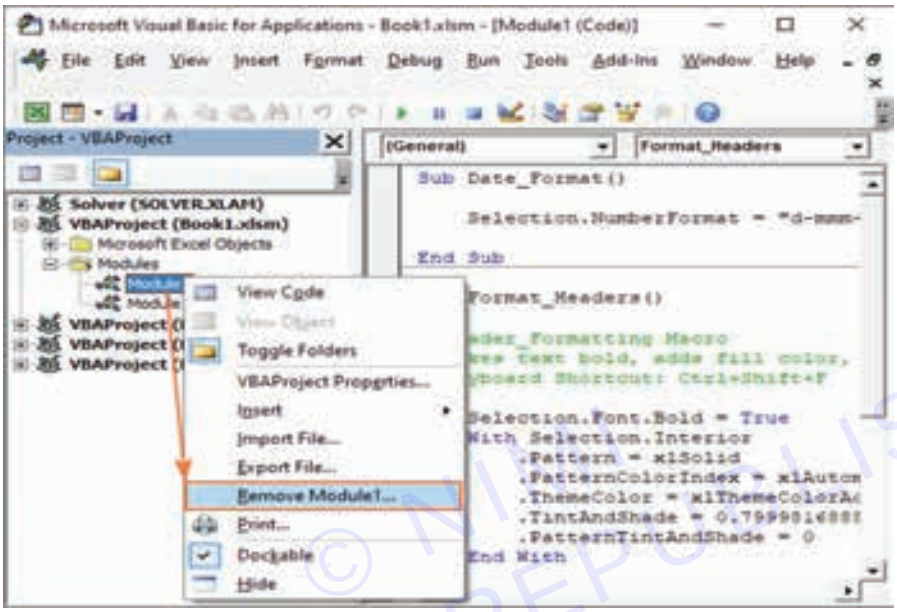
Deleting a macro via Visual Basic Editor

A benefit of using the VBA Editor is that it enables you to delete an entire module with all the macros it contains in one go. Also, the VBA Editor allows deleting macros in the Personal Macro Workbook without unhiding it.

To permanently delete a module, perform these steps

1 In the Project Explorer, right-click on the module and choose Remove from the context menu.

2 When asked whether you want to export the module before removing it, click No.



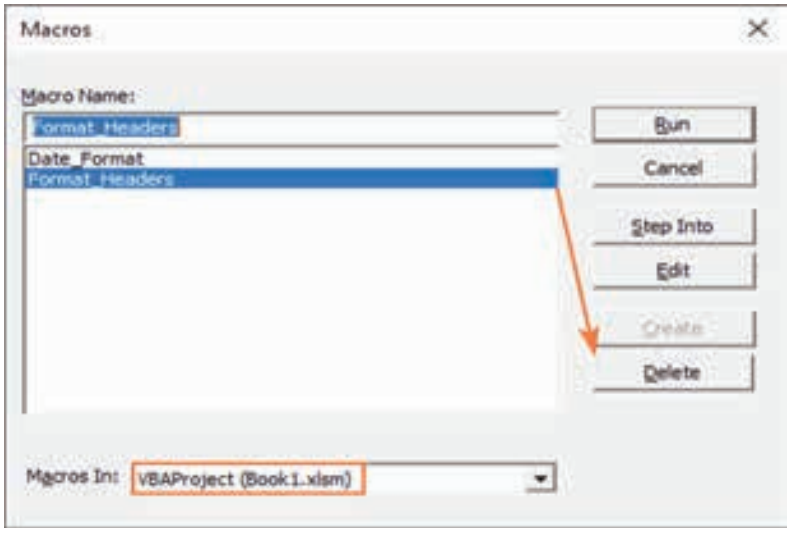
To remove a specific macro, simply delete its source code directly in the Code window. Or, you can delete a macro by using the Tools menu of the VBA Editor:

1 From the Tools menu, choose Macros. The Macros dialog box will appear.

2 In the Macros In drop-down list, select the project containing the unwanted macro.

3 In the Macro Name box, select the macro.

4 Click the Delete button.

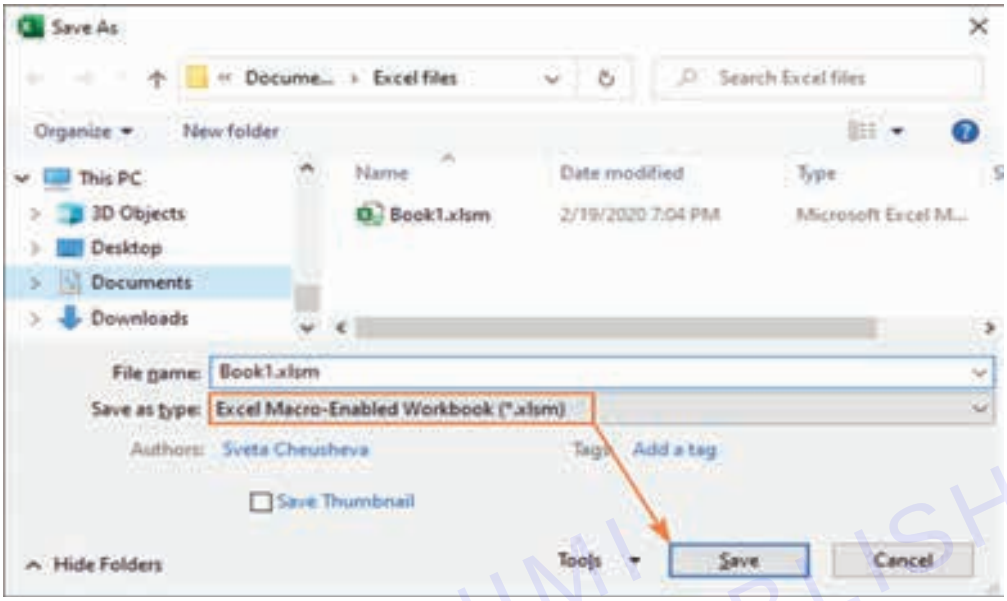


How to save macros in Excel

To save a macro in Excel, either recorded or written manually, just save the workbook as macro enabled (*.xlms). Here's how:

1 In the file containing the macro, click the Save button or press Ctrl + S.

2 The Save As dialog box will appear. Choose Excel Macro-Enabled Workbook (*.xlsm) from the Save as type drop-down list and click Save.



How to export and import macros in Excel

If you'd like to share your VBA codes with someone or move them to another computer, the fastest way is to export the entire module as a bas file.

Exporting macros

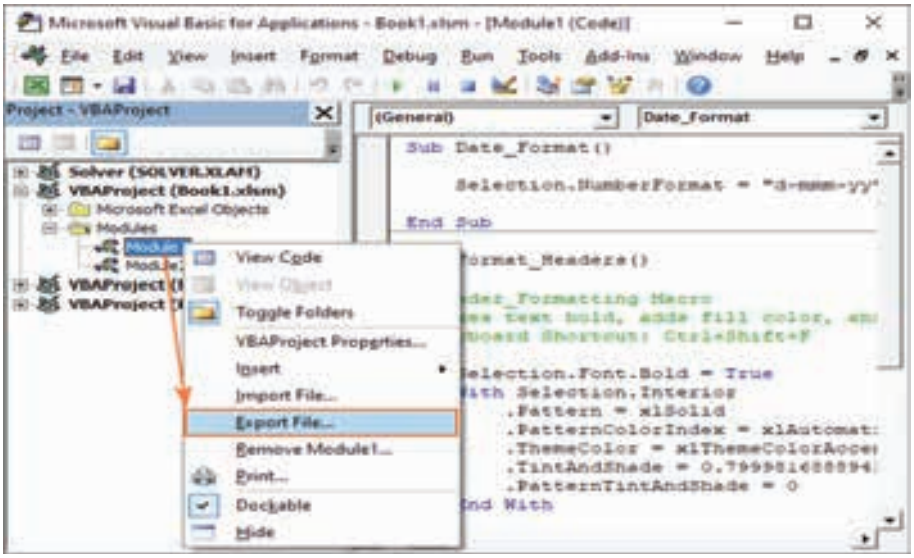
To export your VBA codes, this is what you need to do:

1 Open the workbook containing the macros.

2 Press Alt + F11 to open the Visual Basic Editor.

3 In the Project Explorer, right-click the Module containing the macros and select Export File.

4 Navigate to the folder where you want to save the exported file, name the file, and click Save.



Importing macros

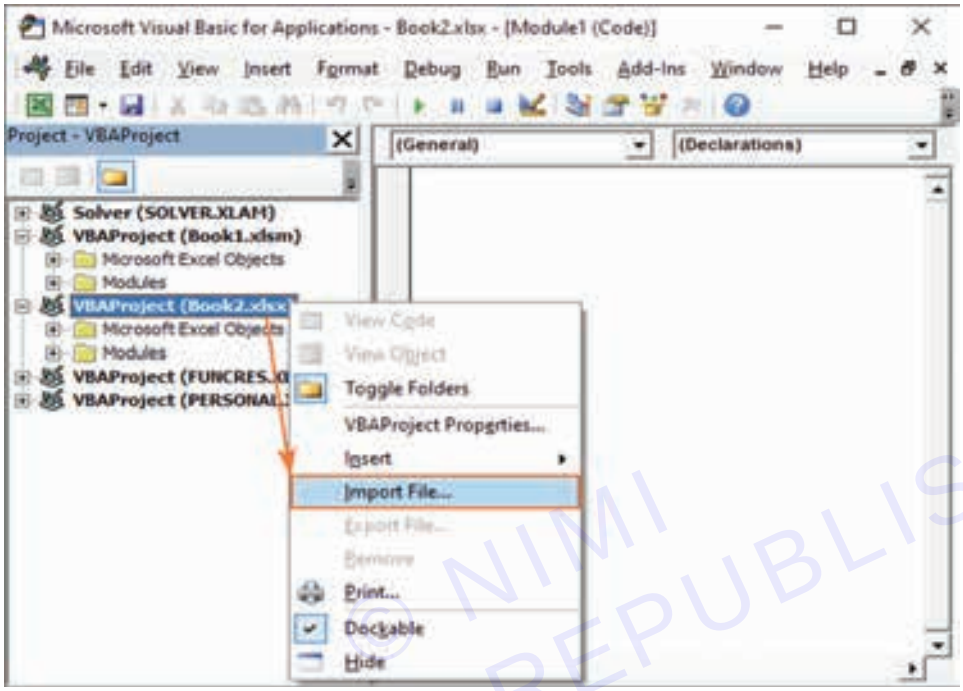
To import a bas file with VBA codes into your Excel, please follow these steps:

1 Open the workbook into which you want to import macros.

2 Open the Visual Basic Editor.

3 In the Project Explorer, right-click the project name and select Import File.

4 Navigate to the bas file and click Open.



Excel macro examples

One of the best ways to learn Excel VBA is by exploring code samples. Below you will find examples of very simple VBA codes that automate some basic operations. Of course, these examples won't teach you coding, for this there exist hundreds of professional-grade VBA tutorials. We just aim to illustrate a few common features of VBA that will hopefully make its philosophy a little more familiar to you.

Unhide all sheets in a workbook

In this example, we use the Active Workbook object to return the currently active workbook and the For Each loop to go through all the sheets in the workbook one-by-one. For each found sheet, we set the Visible property to xlSheetVisible.

Sub Unhide_All_Sheets () Dim wks. As Worksheet For Each wks. In ActiveWorkbook.Worksheets wks. Visible = xlSheetVisible Next wks. End Sub

Hide active worksheet or make it very hidden

To manipulate the currently active sheet, use the Active Sheet object. This sample macro changes the visible property of the active sheet to xlSheetHidden to hide it. To make the sheet very hidden, set the visible property to xlSheetVeryHidden.

Sub Hide_Active_Sheet () ActiveSheet.Visible = xlSheetHidden End Sub

Unmerge all merged cells in selected range

If you want to perform certain operations on a range rather than the entire worksheet, use the Selection object. For example, the below code will unmerge all the merged cells in a selected range at one fell swoop.

Sub Unmerge Cells () Selection.Cells.UnMerge End Sub

Show a message box

To show some message to your users, use the MsgBox function. Here's an example of such a macro in its simplest form:

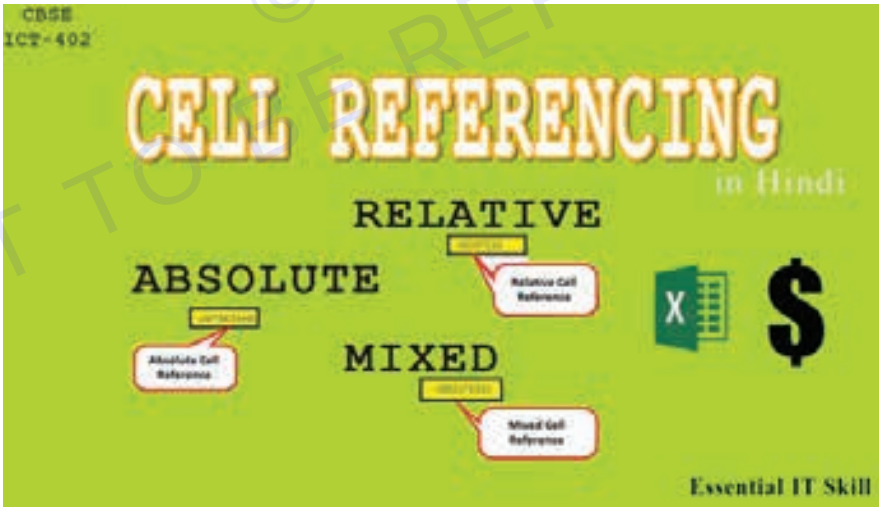
Sub Show Message () MsgBox ("Hello World!") End Sub

In real-life macros, a message box is typically used for information or confirmation purposes. For instance, before performing a action (unmerging cells in our case), you display a Yes/No message box. If the user clicks "Yes", the selected cells are unmerged.

Sub Unmerge_Selected_Cells () Dim Answer As String Answer = MsgBox ("Are you sure you want to unmerge these cells?" vbQuestion + vbYesNo, "Unmerge Cells") If Answer = vbYes Then Selection.Cells.UnMerge End If End Sub

To test the code, select one or more range containing merged cells and run the macro. The following message will appear.

Cell reference-Relative, Absolute, Row Absolute & Column Absolute, Reference form other sheet



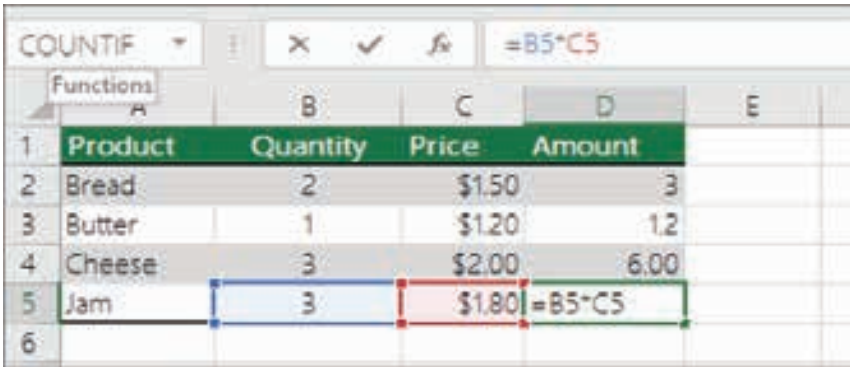
Relative, Absolute, and Mixed Cell References in Excel and Sheets

A cell reference in spread sheet programs such as Excel and Google Sheets identifies the location of a cell in the worksheet. These references use Auto fill to adjust and change information as needed in your spread sheet.

By default, a cell reference is a relative reference, which means that the reference is relative to the location of the cell. If, for example, you refer to cell A2 from cell C2, you are actually referring to a cell that is two columns to the left (C minus A)—in the same row (2). When you copy a formula that contains a relative cell reference, that reference in the formula will change.

As an example, if you copy the formula =B4*C4 from cell D4 to D5, the formula in D5 adjusts to the right by one column and becomes =B5*C5. If you want to maintain the original cell reference in this example when you copy it, you make the cell reference absolute by preceding the columns (B and C) and row (2) with a dollar sign ($). Then, when you copy the formula =$B$4*$C$4 from D4 to D5, the formula stays exactly the same.

Note: The information in this article applies to Excel versions 2019, 2016, 2013, Excel for Mac, Excel Online, and Google Sheets.



Less often, you may want to mixed absolute and relative cell references by preceding either the column or the row value with a dollar sign—which fixes either the column or the row (for example, $B4 or C$4).

To change the type of cell reference:

1 Select the cell that contains the formula.

2 In the formula bar, select the reference that you want to change.

3 Press F4 to switch between the reference types.

The table below summarizes how a reference type updates if a formula containing the reference is copied two cells down and two cells to the right.

The Different Types of Cell References

The three types of references that can be used in Excel and Google Sheets are easily identified by the presence or absence of dollar signs ($) within the cell reference. A dollar sign tells the program to use that value every time it runs a formula.

- Relative cell references contain no dollar signs (i.e., A1).
- Mixed cell references have dollar signs attached to either the letter or the number in a reference but not both (i.e., $A1 and A$1).
- Absolute cell references have dollar signs attached to each letter or number in a reference (i.e., $A$1).

You'll typically use an absolute or mixed cell reference if you set up a formula. For example, if you have a number in Cell A1, more numbers in Column B, and Column C contains the sums of A1 and each of the values in B, you'll use "$A$1" in the SUM formula so that when you auto fill, the program knows to always use the number in A1 instead of the empty cells below it.

A cell is one of the boxlike structures that fill a worksheet, and you can locate one by its references, such as A1, F26, or W345. A cell reference consists of the column letter and row number that intersect at the cell's location. When listing a cell reference, the column letter always appears first. Cell references appear in formulas, functions, charts, and other Excel commands.

How Cell References Use Automatic Updating

One advantage of using cell references in spread sheet formulas is that, normally, if the data located in the referenced cells changes, the formula or chart automatically updates to reflect the change.

If a workbook has been set not to update automatically when you make changes to a worksheet, you can carry out a manual update by pressing the F9 key on the keyboard.

You Can Reference Cells From Different Worksheets

Cell references are not restricted to the same worksheet where the data is located. Other worksheets in the same file can reference each other by including a notation that tells the program which sheet to pull the cell from.

You don't need a sheet notation if you're referring to a cell in the same worksheet.

Similarly, when you reference data in a different workbook, the name of the workbook and the worksheet are included in the reference along with the cell location.

To reference a cell on a different sheet, preface the cell reference with "Sheet [number]" with an exclamation point after it, and then the name of the cell. So if you want to pull info from Cell A1 in Sheet 3, you'll type, "Sheet3! A1."

A notation referring to another workbook in Excel also includes the name of the book in brackets. To use the information contained in Cell B2 in Sheet 2 of Workbook 2, you'll type, "[Book2] Sheet2!B2."

Cell Range: A Quick Primer

While references often refer to individual cells, such as A1, they can also refer to a group or range of cells. You identify ranges of cells by the starting and ending cells. In the case of ranges that occupy multiple rows and columns, you'll use the cell references of the cells in the upper left and lower right corners of the range.

Separate the limits of a cell range with a colon (:), which tells Excel or Google Sheets to include all the cells between these start and end points. So to grab everything between Cell A1 and D10, you'd type, "A1:D10."

To capture an entire row or column, you still use the cell range notation, but you only use the column numbers or row letters. To include everything in Column A, the range will be "A:A." To use Row 8, you'll type, "8:8." For everything in Columns B through D, you'll type, "B:D."

Copying Formulas and Different Cell References

Another advantage of using cell references in formulas is that they make it easier to copy formulas from one location to another in a worksheet or workbook.

Relative cell references change when copied to reflect the new location of the formula. The name relative comes from the fact that they change relative to their location when copied. This is usually a good thing, and it is why relative cell references are the default type of reference used in formulas.

At times, cell references need to stay static when formulas are copied. Copying formulas is the other major use of an absolute reference such as =$A$2+$A$4. The values in those references don't change when you copy them.

At other times, you may want part of a cell reference to change, such as the column letter, while having the row number stay static or vice versa when you copy the formula. In this case, you'll use a mixed cell reference such as =$A2+A$4. Whichever part of the reference has a dollar sign attached to it stays static, while the other part changes when copied.

So for $A2, when it is copied, the column letter is always A, but the row numbers change to $A3, $A4, $A5, and so on.

The decision to use the different cell references when creating the formula is based on the location of the data that the copied formulas will use.

Toggling Between Types of Cell References

The easiest way to change cell references from relative to absolute or mixed is to press the F4 key on the keyboard. To change existing cell references, Excel must be in edit mode, which you enter by double-clicking on a cell with the mouse pointer or by pressing the F2 key on the keyboard.

To convert relative cell references to absolute or mixed cell references:

- Press F4 once to create a cell reference fully absolute, such as $A$6.
- Press F4 a second time to create a mixed reference where the row number is absolute, such as a$6.
- Press F4 a third time to create a mixed reference where the column letter is absolute, such as $A6.
- Press F4 a fourth time to make the cell reference relative again, such as A6.

Row and Column cell references (R1C1)

Excel normally displays cell references in the A1 style i.e with rows each given a number and columns given an alphabetic character.

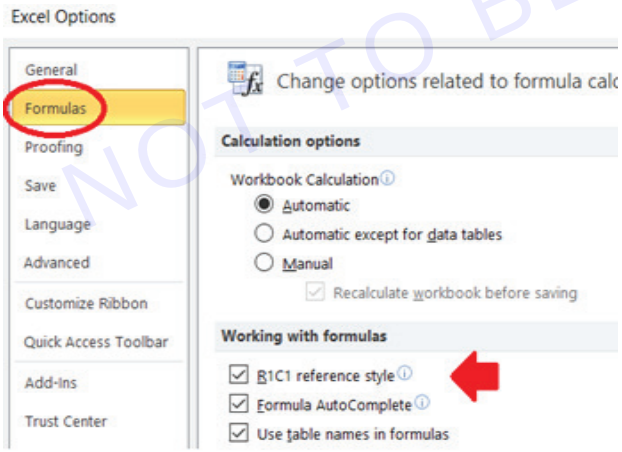
Absolute or Relative cell references determine whether ranges are incremented as they are copied to other cells. Placing the dollar symbol ($) in front of a column letter or row number will fix it if the formula is copied to other cells. If the relative formula SUM(A5:A10) is copied to the adjacent cell, it will update to SUM(B5:B10). The absolute formula SUM($A$5:$A$10) will refer to the same range, regardless of where it is copied to.

Excel can also handle cell references using a different format or notation. The R1C1 notation describes a cell location in terms of a row number and a column number. It can also distinguish between absolute and relative references. R5C1 is an absolute reference (for cell $A$5) whereas R[4]C[1] is four rows below and one column to the right, relative to the current cell.

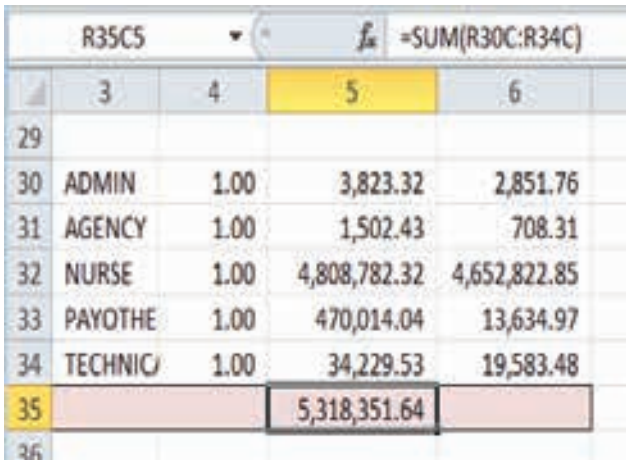
	A	B	C	D	E	F	G
1	**Expenditure**						
2			Month number	3			Total
3			April	May	June	July	to M03
4		Ward 10	89,752	44,324	81,193	65,063	215,269
5		Ward 11	35,324	87,338	56,219	91,235	178,881
6		Ward 12	63,290	15,295	59,293	86,468	137,878
7		Ward 13	60,277	69,067	81,691	2,068	211,035
9		Totals	248,643	216,024	278,397	244,833	
10			[C9] =SUM($C$4:$C$7) OR =SUM(R4C3:R7C3)				
11				[D9] =SUM(D4:D8) OR =SUM(R[-5]C:R[-1]C)			
12					[E9] =SUM(E4:E8) OR =SUM(R[-5]C:R[-1]C)		

In the above example the formulas in Cells C9, D9 and E9 are shown in both A1 and R1C1 notation. The first formula contains absolute references. The next two are relative formulas. Note however that in R1C1 format, these two are identical.

Select File | Options and Formulas to modify the worksheet appearance and display references as R1C1. Formulas change and column labels will switch from letters to numbers.



Use Excel options to toggle the Style

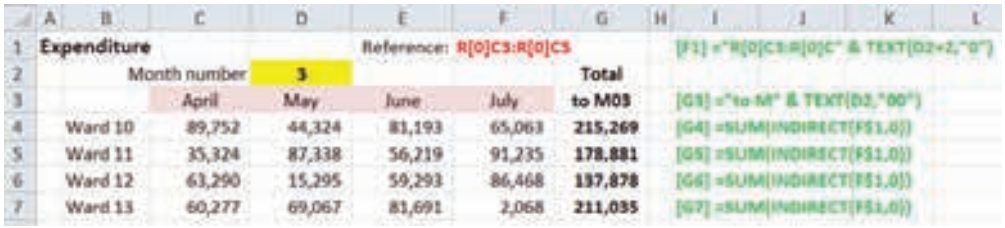


Style Worksheet with R1C1 references ON

These alternative R1C1 references can be applied using the Indirect function without switching Excel's formula properties. e.g. =SUM(INDIRECT("R[-5]C[0]:R[-1]C[0]",0)) will sum the five cells above the current cell. Note that an extra argument must be supplied to the Indirect function. =INDIRECT(TextRng, Type_TF). If TextRng is an R1C1 reference, the optional Type_TF argument must be set to FALSE or zero. If it is any other value or omitted the TextRng is assumed to be in normal A1 notation.

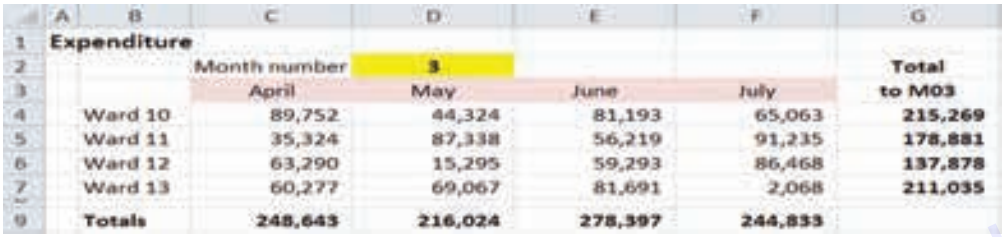
The next example uses the Indirect function to read the R1C1 style reference in cell F1. That string in turn reflects the month number in cell D2. The reference currently represents the first 3 columns of values on the current row.

The nature of R1C1 references means that the exact same range string can be applied (using INDIRECT) against all of the cells in G4:G7. With the A1 notation a different INDIRECT reference would be required for each row. This example has been created to demonstrate INDIRECT() and R1C1 references. If I was trying to construct a table like this with dynamic formulas I would probably use INDEX() or OFFSET() in preference to this method.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Expenditure				Reference:	R[0]C3:R[0]C5			[F1] ="R[0]C3:R[0]C" & TEXT(D2+2,"0")			
2			Month number	3			Total					
3			April	May	June	July	to M03		[G3] ="to-M" & TEXT(D2,"00")			
4		Ward 10	89,752	44,324	81,193	65,063	215,269		[G4] =SUM(INDIRECT(F$1,0))			
5		Ward 11	35,324	87,338	56,219	91,235	178,881		[G5] =SUM(INDIRECT(F$1,0))			
6		Ward 12	63,290	15,295	59,293	86,468	137,878		[G6] =SUM(INDIRECT(F$1,0))			
7		Ward 13	60,277	69,067	81,691	2,068	211,035		[G7] =SUM(INDIRECT(F$1,0))			

In the following (embedded web app) example you can edit the month number value in cell D3 and thereby change the R1C1 string in cell F1. This string is used in each row in column G to supply the INDIRECT function with a dynamic range.



	A	B	C	D	E	F	G
1	Expenditure						
2			Month number	3			Total
3			April	May	June	July	to M03
4		Ward 10	89,752	44,324	81,193	65,063	215,269
5		Ward 11	35,324	87,338	56,219	91,235	178,881
6		Ward 12	63,290	15,295	59,293	86,468	137,878
7		Ward 13	60,277	69,067	81,691	2,068	211,035
9		Totals	248,643	216,024	278,397	244,833	

R1C1 references are most useful when writing Visual Basic code. If you wish to use VBA to write cell formulas, it is much easier to write code to increment 1, 2, 3 rather than A, B, C. Also, a single R1C1 formula with relative references can be copied (using VBA) to a range of cells and will adjust to the appropriate rows and columns.

Named Ranges in Excel – An Introduction

How to Create Named Ranges in Excel

What's in the name?

If you are working with Excel spreadsheets, it could mean a lot of time saving and efficiency.

If someone has to call me or refer to me, they will use my name (instead of saying a male is staying in so and so place with so and so height and weight).

Right?

Similarly, in Excel, you can give a name to a cell or a range of cells.

Now, instead of using the cell reference (such as A1 or A1:A10), you can simply use the name that you assigned to it.

For example, suppose you have a data set as shown below.

	A	B	C
1	Date	Sales Rep	Sales
2	16/05/2018	Joe	899
3	29/12/2017	Tom	735
4	14/08/2017	Kim	572
5	21/02/2018	Marie	663
6	27/03/2018	Josh	638
7	07/09/2017	Martha	550
8	09/08/2017	Jessica	593
9	22/05/2018	Alvin	857
10	16/05/2018	Brad	684
11	11/06/2017	Mike	566

In this data set, if you have to refer to the range that has the Date, you will have to use A2:A11 in formulas. Similarly, for Sales Rep and Sales, you will have to use B2:B11 and C2:C11.

While it's alright when you only have a couple of data points, in case you have huge complex data sets, using cell references to refer to data could be time-consuming.

Excel Named Ranges makes it easy to refer to data sets in Excel.

You can create a named range in Excel for each data category, and then use that name instead of the cell references. For example, dates can be named 'Date', Sales Rep data can be named 'SalesRep' and sales data can be named 'Sales'.

You can also create a name for a single cell. For example, if you have the sales commission percentage in a cell, you can name that cell as 'Commission'.

Benefits of Creating Named Ranges in Excel

Here are the benefits of using named ranges in Excel.

Use Names instead of Cell References

When you create Named Ranges in Excel, you can use these names instead of the cell references.

For example, you can use =SUM(SALES) instead of =SUM(C2:C11) for the above data set.

Have a look at the formulas listed below. Instead of using cell references, I have used the Named Ranges.

- Number of sales with value more than 500: =COUNTIF(Sales,">500″)
- Sum of all the sales done by Tom: =SUMIF(SalesRep,"Tom",Sales)
- Commission earned by Joe (sales by Joe multiplied by commission percentage):

=SUMIF(SalesRep,"Joe",Sales)*Commission

You would agree that these formulas are easy to create and easy to understand (especially when you share it with someone else or revisit it yourself.

No Need to Go Back to the Dataset to Select Cells

Another significant benefit of using Named Ranges in Excel is that you don't need to go back and select the cell ranges.

Named Ranges Make Formulas Dynamic

By using Named Ranges in Excel, you can make Excel formulas dynamic.

For example, in the case of sales commission, instead of using the value 2.5%, you can use the Named Range.

Now, if your company later decides to increase the commission to 3%, you can simply update the Named Range, and all the calculations would automatically update to reflect the new commission.

How to Create Named Ranges in Excel

Here are three ways to create Named Ranges in Excel:

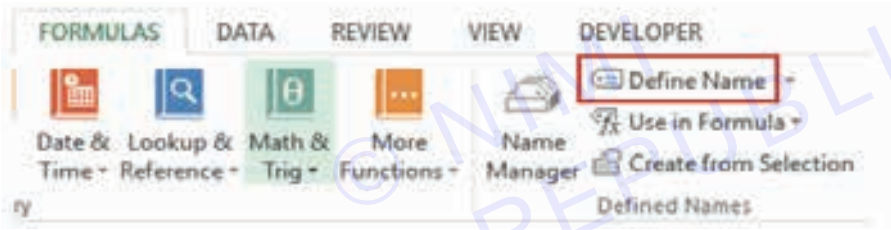
Method #1 – Using Define Name

Here are the steps to create Named Ranges in Excel using Define Name:

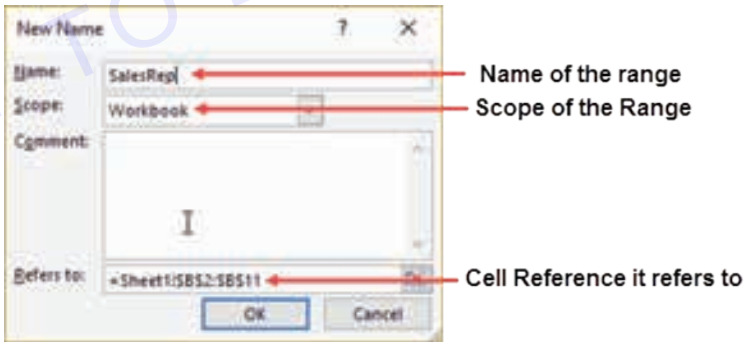
- Select the range for which you want to create a Named Range in Excel.

	A	B	C
1	Date	Sales Rep	Sales
2	16/05/2018	Joe	899
3	29/12/2017	Tom	735
4	14/08/2017	Kim	572
5	21/02/2018	Marie	663
6	27/03/2018	Josh	638
7	07/09/2017	Martha	550
8	09/08/2017	Jessica	593
9	22/05/2018	Alvin	857
10	16/05/2018	Brad	684
11	11/06/2017	Mike	566

- Go to Formulas –> Define Name.



In the New Name dialogue box, type the Name you wish to assign to the selected data range. You can specify the scope as the entire workbook or a specific worksheet. If you select a particular sheet, the name would not be available on other sheets.



- Click OK.

This will create a Named Range SALES REP.

Method #2: Using the Name Box

- Select the range for which you want to create a name (do not select headers).
- Go to the Name Box on the left of the Formula bar and Type the name of the with which you want to create the Named Range.
- Note that the Name created here will be available for the entire Workbook. If you wish to restrict it to a worksheet, use Method 1.

Method #3: Using Create From Selection Option

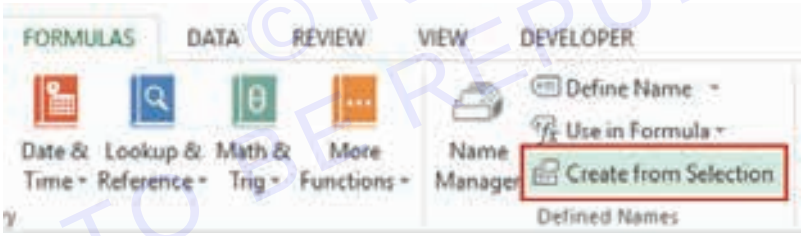
This is the recommended way when you have data in tabular form, and you want to create a named range for each column/row.

For example, in the dataset below, if you want to quickly create three named ranges (Date, Sales_Rep, and Sales), then you can use the method shown below.

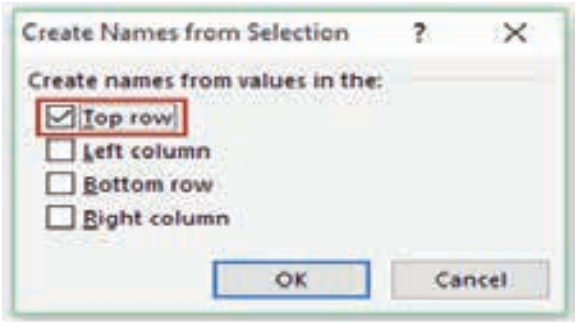
	A	B	C
1	Date	Sales Rep	Sales
2	16/05/2018	Joe	899
3	29/12/2017	Tom	735
4	14/08/2017	Kim	572
5	21/02/2018	Marie	663
6	27/03/2018	Josh	638
7	07/09/2017	Martha	550
8	09/08/2017	Jessica	593
9	22/05/2018	Alvin	857
10	16/05/2018	Brad	684
11	11/06/2017	Mike	566

Here are the steps to quickly create named ranges from a dataset:

- Select the entire data set (including the headers).
- Go to Formulas –> Create from Selection (Keyboard shortcut – Control + Shift + F3). It will open the 'Create Names from Selection' dialogue box.



- In the Create Names from Selection dialogue box, check the options where you have the headers. In this case, we select the top row only as the header is in the top row. If you have headers in both the top row and left column, you can choose both. Similarly, if your data is arranged when the headers are in the left column only, then you only check the Left Column option.



This will create three Named Ranges – Date, Sales_Rep, and Sales.

Note that it automatically picks up names from the headers. If there are any space between words, it inserts an underscore (as you can't have spaces in named ranges).

Naming Convention for Named Ranges in Excel

There are certain naming rules you need to know while creating Named Ranges in Excel:

- The first character of a Named Range should be a letter and underscore character(_), or a backslash(\). If it's anything else, it will show an error. The remaining characters can be letters, numbers, special characters, period, or underscore.
- You can not use names that also represent cell references in Excel. For example, you can't use AB1 as it is also a cell reference.
- You can't use spaces while creating named ranges. For example, you can't have Sales Rep as a named range. If you want to combine two words and create a Named Range, use an underscore, period or uppercase characters to create it. For example, you can have Sales_Rep, SalesRep, or SalesRep.
- While creating named ranges, Excel treats uppercase and lowercase the same way. For example, if you create a named range SALES, then you will not be able to create another named range such as 'sales' or 'Sales'.
- A Named Range can be up to 255 characters long.

What Are Excel Errors

Excel is a powerful tool for working with data, but it can also be frustrating when errors occur. While some errors are easy to fix, others can be more difficult to troubleshoot. By understanding the different types of Excel errors, you can learn how to avoid them and quickly fix them when they occur. This will help you save time and ensure that your Excel workbooks are accurate and error-free.

Below is the list of some most common errors that we can find in the Excel formula

1 #NAME? Error: This Excel error usually occurs because of the non-existent function.

2 #DIV/0! Error: This Excel error because if we try to divide the number by zero or vice-versa.

3 #REF! Error: This error arises due to a reference missing.

4 #NULL! Error: This error comes due to unnecessary spaces inside the function.

5 #N/A Error: The function cannot find the required data. Maybe the wrong reference is given.

6 ###### Error: This is not a true Excel formula error, but it occurs because of a formatting issue. Probably, the value in the cell is more than the column width.

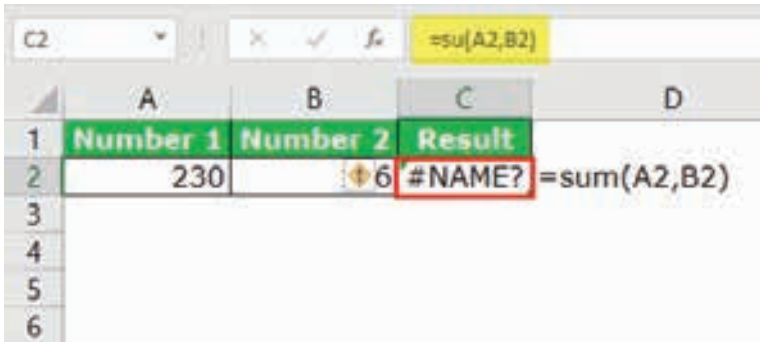
7 #VALUE! Error: This is one of the common Excel formula errors we see in Excel. It occurs due to the wrong data type of the parameter given to the function.

8 #NUM! Error: Excel formula error because the number we have supplied to the formula is not proper.

#1 #NAME? Error

The #NAME? Excel formula error is when Excel cannot find the supplied function or the parameters we gave do not match the standards of the function.

Just because of the display value of #NAME? It does not mean Excel is asking your name. Rather, it is due to the wrong data type for the parameter.



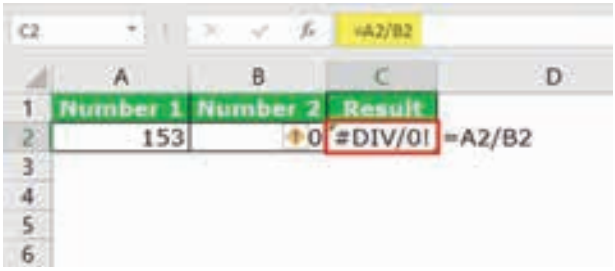
#2 #DIV/0! Error

The #DIV/0! Error is due to the wrong calculation method, or one of the operators is missing. This error occurs in calculations. For example, if we have a "Budgeted" number in cell A1 and the "Actual" number in cell B1, we must divide the B1 cell by the A1 cell to calculate the efficiency. If any cells are empty or zero, we may get this error.

If we notice in the formula bar, instead of the Sum formula in excel, we have misspelt the formula as su. So the result of this function is #NAME?.

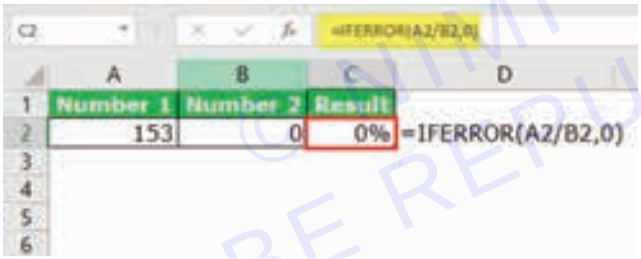
How to Fix the #Name? ERROR Issue?

To fix the Excel formula error, we must look for the formula bar and check if the formula we have inserted is valid. If the formula spelling is inaccurate, we must change that to the correct wording.



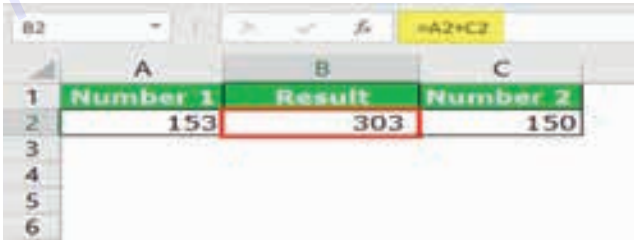
How to Fix the #DIV/o! ERROR Issue?

To fix this Excel formula error, we must use one formula to calculate the efficiency level. Therefore, we can use the IFERROR function in excel and calculate inside the function.

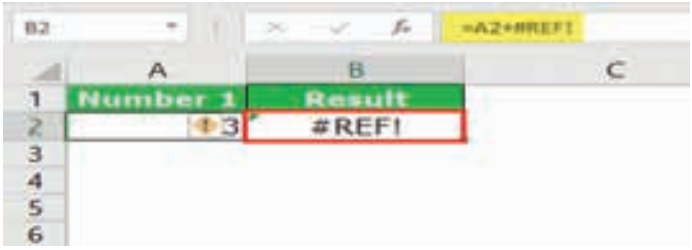


#3 #REF! Error

The #REF! Error is due to the supply of the reference missing suddenly. Look at the below example in cell B2. We have applied the formula "A2 + C2."



Now, we will delete column C2 and see the result in cell B2.



How to Fix the #REF! ERROR Issue?

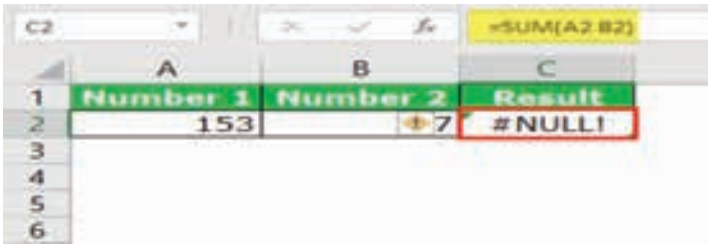
Before adding or deleting anything to the existing data set, we must ensure all the formulas are pasted as values. If the deleting cell is referred to any of the cells, we may encounter this error.

We must always undo the action if we accidentally delete any rows, columns, or cells.

#4 #NULL! Error

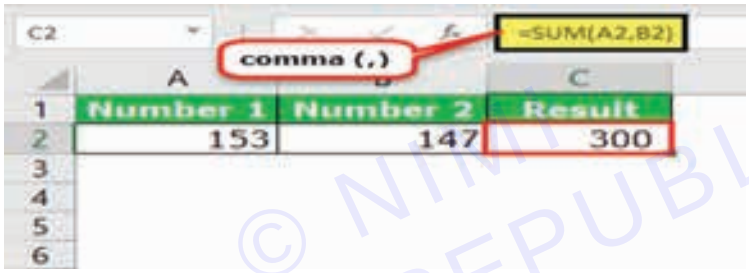
The #NULL! Error is due to the wrong value supplied to the required parameters; for example, after the improper supply of range operator, incorrect mention of parameter separator.

Look at the below image. We have applied the SUM formula to calculate the sum of the values in cells A2 and B2. The mistake we made here is after the first argument. We should give a comma (,) to separate the two arguments. Instead, we have provided space.

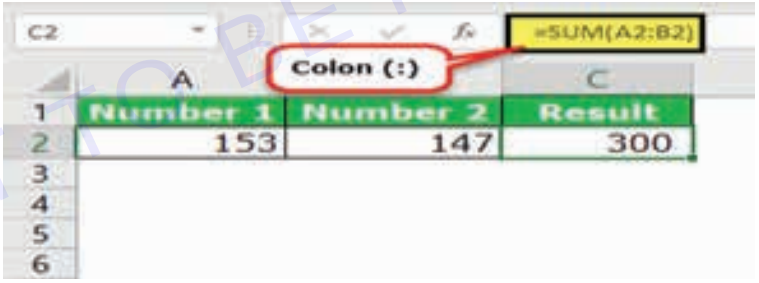


How to Fix the #NULL! ERROR Issue?

In these cases, we need to mention the exact argument separators. For example, we should use the comma (,) after the first argument in the above image.



In the case of a range, we need to use a colon (:).



#5 #VALUE! Error

The #VALUE! error occurs if the formula cannot find the specified result. It is due to non-numerical values or wrong data types to the argument.

Look at the image below. We have calculated the commission amount based on the sales value and commissions.

	A	B	C	D	E
1	ID	Sales Value	Commission %	Commission Amt	
2	153	2179	5.00%	108.95	=C2*B2
3	154	1677	5.50%	92.235	=C3*B3
4	155	2322	6.00%	139.32	=C4*B4
5	156	2076	7.00%	145.32	=C5*B5
6	157	1538	Not Decided	#VALUE!	=C6*B6
7	158	3246	5.00%	162.3	=C7*B7
8	159	3249	Not Decided	#VALUE!	=C8*B8
9	160	2448	5.00%	122.4	=C9*B9
10					

If we notice the cells D6 and D8, we get an error as #VALUE!. The reason we got #VALUE! Error because there is no commission percentage in the cells C6 and C8.

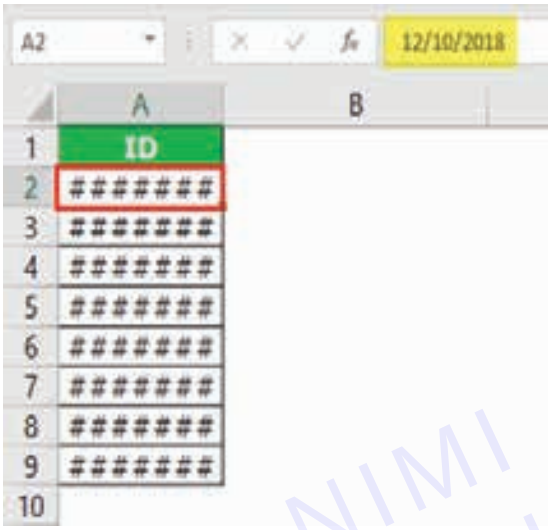
We cannot multiply the text value with numerical values.

How to Fix the #VALUE! ERROR Issue?

In these cases, we must replace all the text values with zero until we get further information.

#6 ###### Error

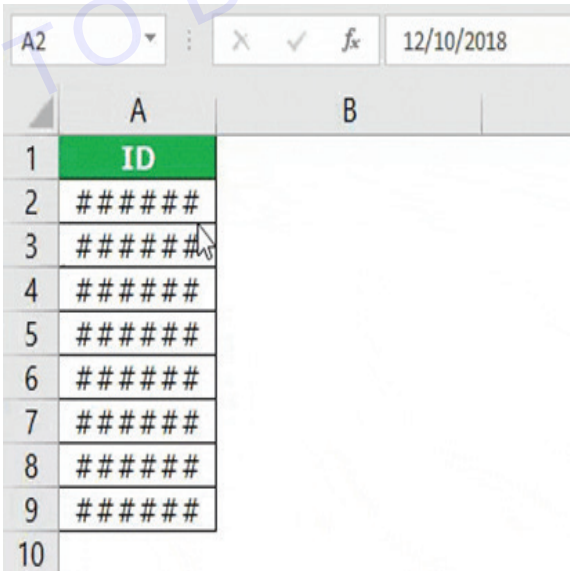
The ###### error, is not an error. Rather, it is just a formatting issue. For example, look at the below image in cell A1 that has entered the date values.



It is due to the length issue of the characters. The values in the cell are more than the column width. In simple terms, the column width is not wide enough.

How to Fix the ###### ERROR Issue?

We must double-click on the column to adjust the column width to get the full values visible.

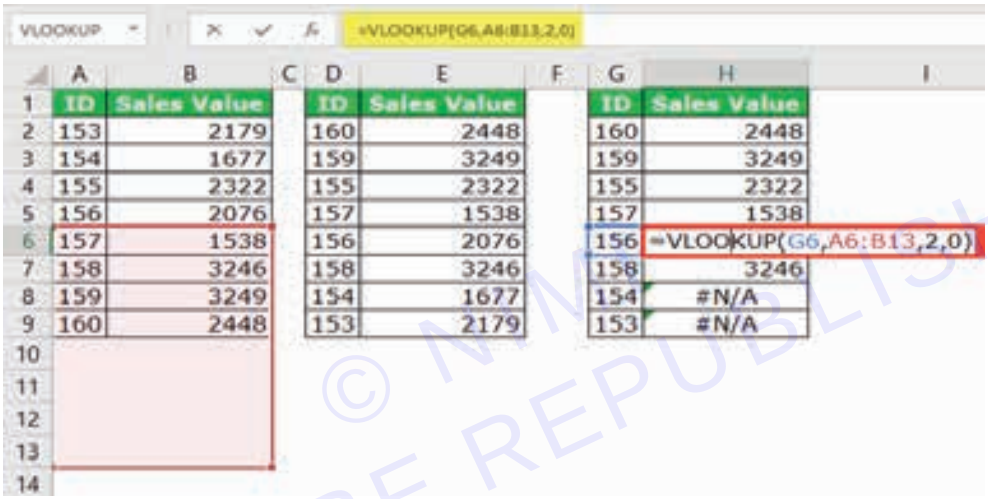


#7 #N/A! Error

The #N/A! error is because the formula cannot find the value in the data. It usually occurs when the Excel lookup functions like VLOOKUP, HLOOKUP, MATCH, and VLOOKUP cannot find the value you are looking for within the formula in the specified range.

	A	B	C	D	E	F	G
1	ID	Sales Value		ID	Sales Value		
2	153	2179		160	2448	=VLOOKUP(D2,A2:B9,2,0)	
3	154	1677		159	3249	=VLOOKUP(D3,A3:B10,2,0)	
4	155	2322		155	2322	=VLOOKUP(D4,A4:B11,2,0)	
5	156	2076		157	1538	=VLOOKUP(D5,A5:B12,2,0)	
6	157	1538		156	#N/A	=VLOOKUP(D6,A6:B13,2,0)	
7	158	3246		158	3246	=VLOOKUP(D7,A7:B14,2,0)	
8	159	3249		154	#N/A	=VLOOKUP(D8,A8:B15,2,0)	
9	160	2448		153	#N/A	=VLOOKUP(D9,A9:B16,2,0)	
10							

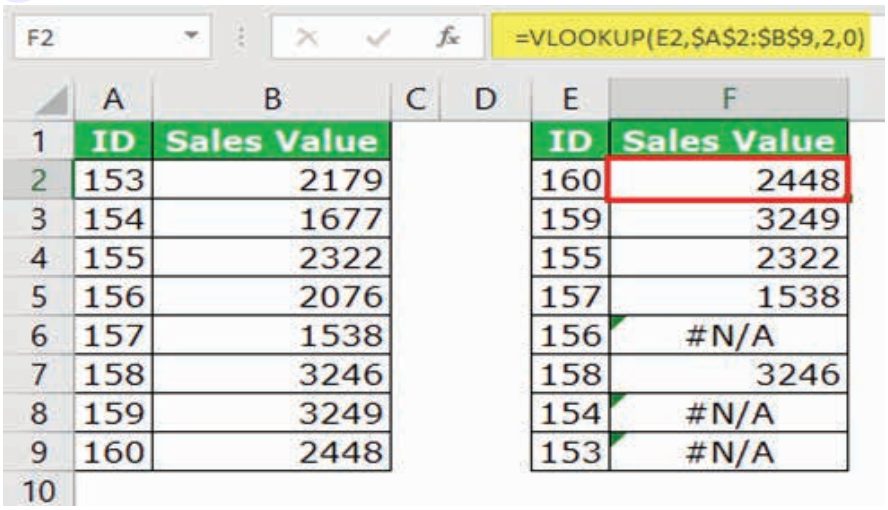
We got some errors as #N/A in the cells E6, E8, and E9. If we look at the ranges for these cells, it does not include the values against those IDs.



ID 156 is not in the range from A6 to B13; we got an error. Similarly, for the remaining two cells, we have the same issue.

How to Fix the #N/A! ERROR Issue?

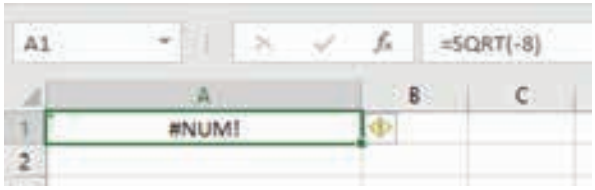
We need to make this an absolute reference when referencing a table range.



We must press the F4 key to make it an absolute reference.

#8 #NUM! Error

Excel error occurs when Excel usually cannot display the result of a mathematical operation. This type of error can occur for two reasons:



For example, a formula or function contains numeric values that aren't valid. Calculating the square root of a negative number =SQRT(-8). This is because, in Excel, imaginary numbers are not considered.

Or it can also be because the result of an operation is too large or small for Excel to display it. For example: Calculating the power of 1000 raised to 300 =POWER(1000,300) will give us a result Excel error type #NUM!.

How to Fix the #NUM! ERROR Issue?

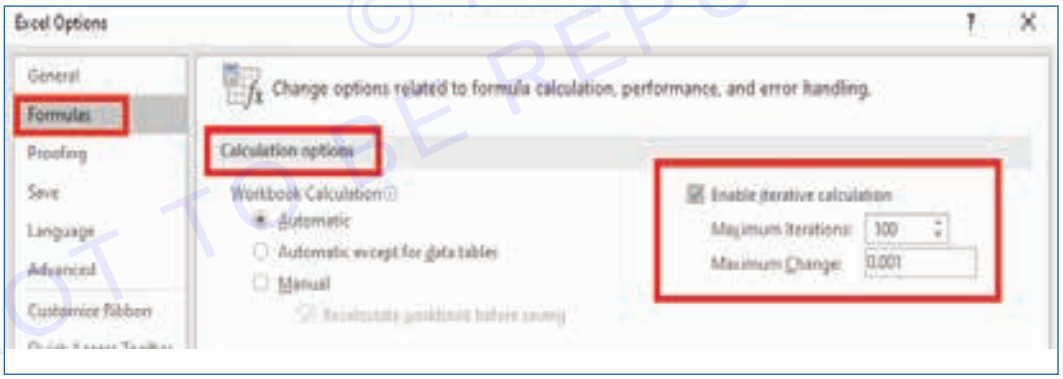
To fix this error in Excel, change the number of times Excel iterates the formula –

1 Go to File < Options.

2 The dialogue box will open. Go to Formulas and then Calculation options. Check the Enable iterative calculation box.

3 You will see two options – Maximum Iterations and Maximum Change. In the Maximum Iterations box, mention the number of times you want Excel to recalculate. The number of iterations is directly proportional to the time Excel takes to calculate a worksheet.

4 Now type the amount of change you will accept between calculation results in the Maximum Change box. The smaller the number, the less time Excel calculates a worksheet.

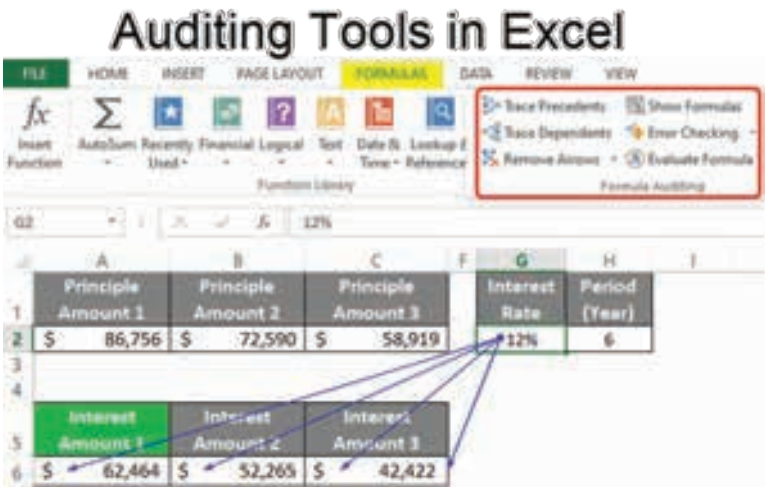


Excel Auditing Tools

Excel auditing tools provide a suite of functionalities that will enable you to review, validate, and troubleshoot formulas and data. Suppose we have a dataset of some projects' Principal Amount, Interest Amount, Monthly Amount, and Interest Rate. Now we will apply all the auditing tools using this data table.

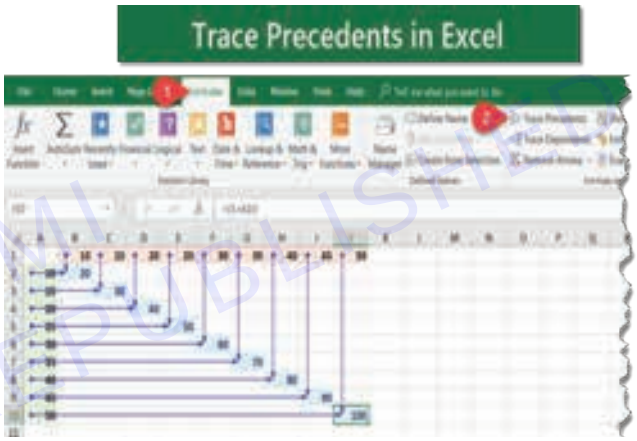
Advantages of Using Excel Auditing Tools

- Auditing tools like "Trace Dependents" and "Trace Precedents" help identify cells that contribute to a specific formula, making it easier to locate errors.
- The "Show Formulas" feature allows you to view the actual formulas in cells, aiming toward understanding complex calculations.
- Tools like "Error Checking" automatically detect and highlight common errors like division by zero or incorrect cell references, aiding in quick error resolution.



Trace Precedents

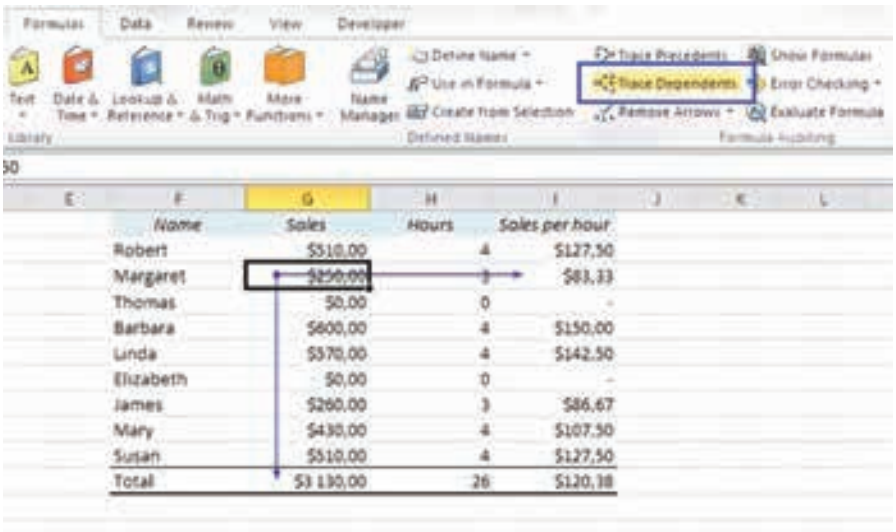
Trace Precedents is a dynamic tool to sort out the complex chain of cell relationships in your spreadsheets. Trace Precedents provide you the ability to understand formula dependencies by highlighting the connections that have an impact on a chosen cell. As we have calculated the interest amount by multiplying the principal amount with the interest rate, let's see what the trace precedent tool show for this calculation.



- To do so, choose a cell (D5), and visit the Trace Precedents option from the Formulas tab.
- As a result, we will see two arrows from cell (C5) and cell (C11) indicating towards cell (D5) as the interest amount is calculated using the principal amount and interest rate.

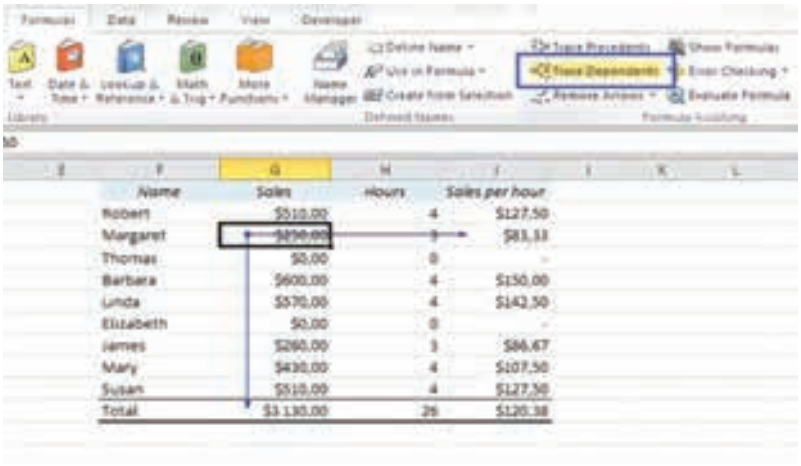
Trace Dependents

In order to visually highlights the cells that depend on the value of a selected cell, you can try the Trace Dependents feature in Excel. This is a powerful tool for understanding the relationship between cells. Here, let's see how the interest rate is dependent on the cells.



- Simply, choose a cell (G22), and then visit the Trace Dependents feature from the Formulas tab.

Finally, you will see the arrows from the cell (G11) to other cells indicating the cells that are dependent on the value of the selected cell.



Remove Arrows

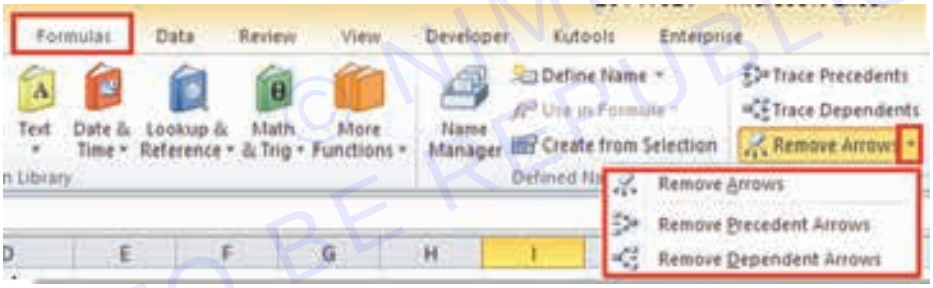
After inserting arrows using the above features, you can also delete them by utilizing the Remove Arrows option.

- Start with, selecting the cell (C11) and clicking the

Remove Arrows option from the Formulas tab.

- Within a blink of an eye, the arrows will be removed.

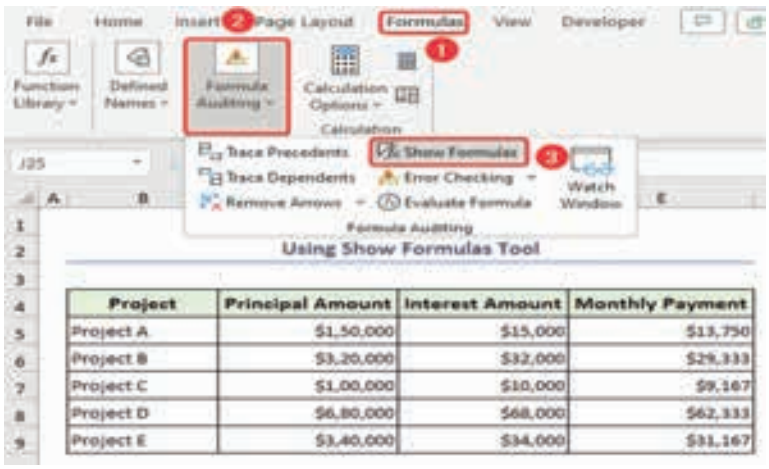
It's that simple.



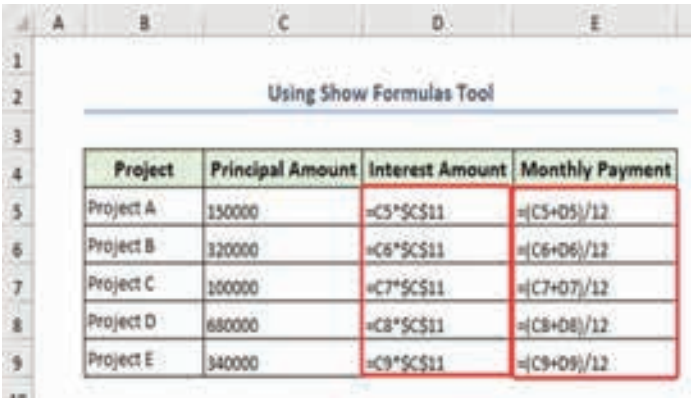
Show Formulas

Show Formulas in Excel is a helpful tool that allows you to view the actual formulas within cells instead of their calculated results. This feature offers transparency into complex calculations, aiding in formula debugging and verification.

- While the worksheet is open, visit the Formulas tab and press the Show Formulas option



Immediately, all the cells with formulas will represent the formulas inside the cells.



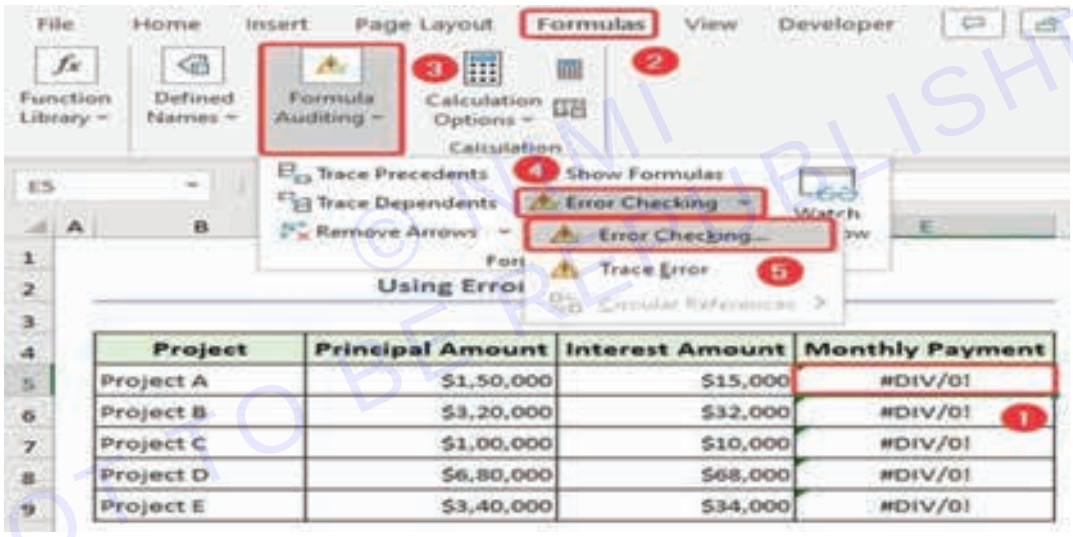
Using Show Formulas Tool

Project	Principal Amount	Interest Amount	Monthly Payment
Project A	150000	=C5*$C$11	=(C5+D5)/12
Project B	320000	=C6*$C$11	=(C6+D6)/12
Project C	100000	=C7*$C$11	=(C7+D7)/12
Project D	680000	=C8*$C$11	=(C8+D8)/12
Project E	340000	=C9*$C$11	=(C9+D9)/12

Error Checking (Includes Error Checking, Trace Error, Circular References)

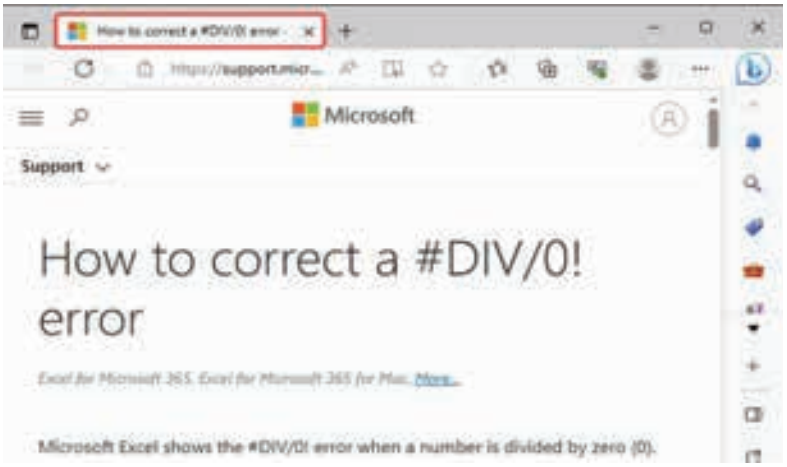
Sometimes while applying formulas, you will get errors like

#DIV/0!, #VALUE!, #NAME? Errors. To check why it's happening you can visit Error Checking option from the Formula tab.

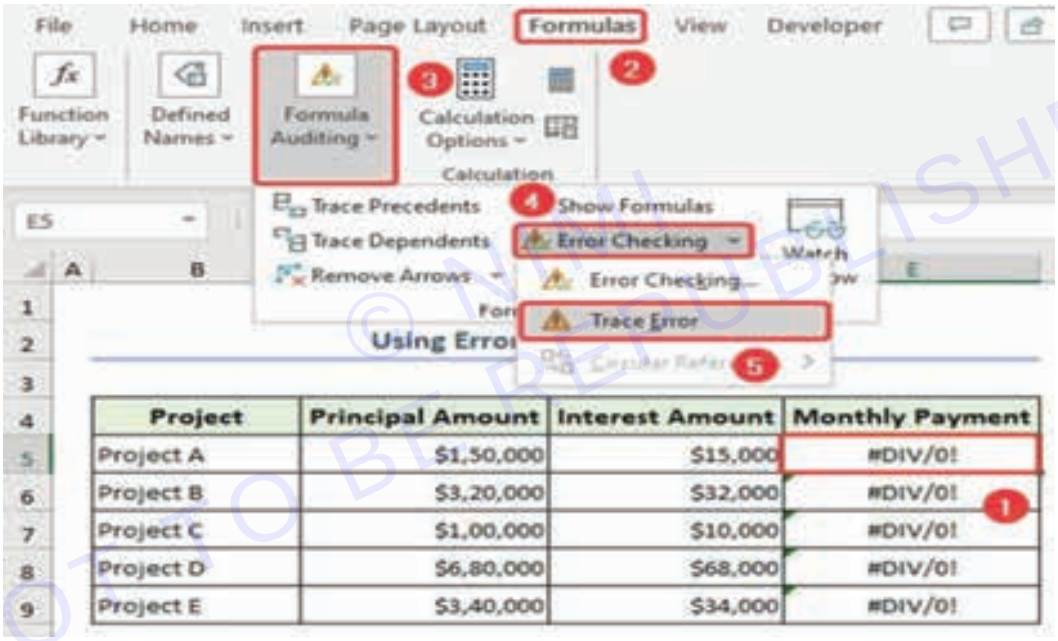


- Select a cell (E5), and hit the Error Checking option from the Formulas tab.
- Now, you can click the Help on this Error option to check in detail about the error.

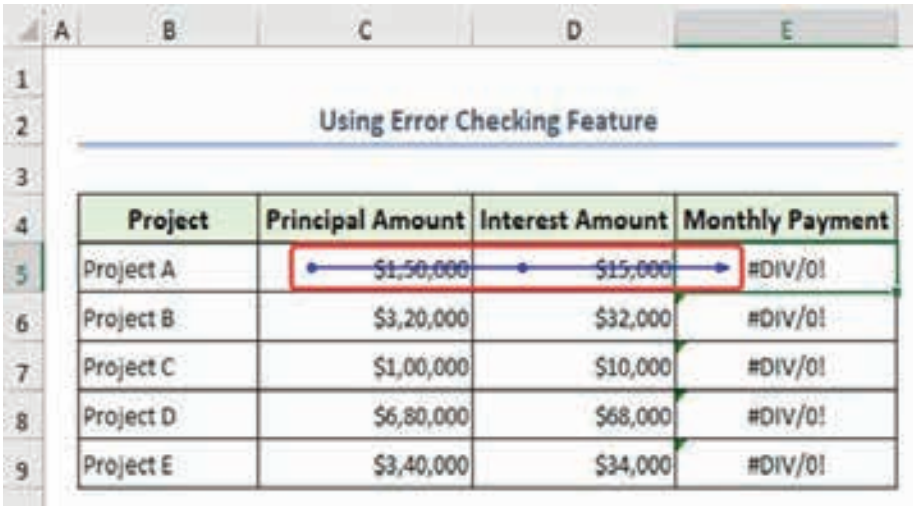
- As a result, a new window will open in your browser providing you details about the error and its solution.



- In order to trace from which cells these are happening, click the Trace Error option from the Formulas tab.



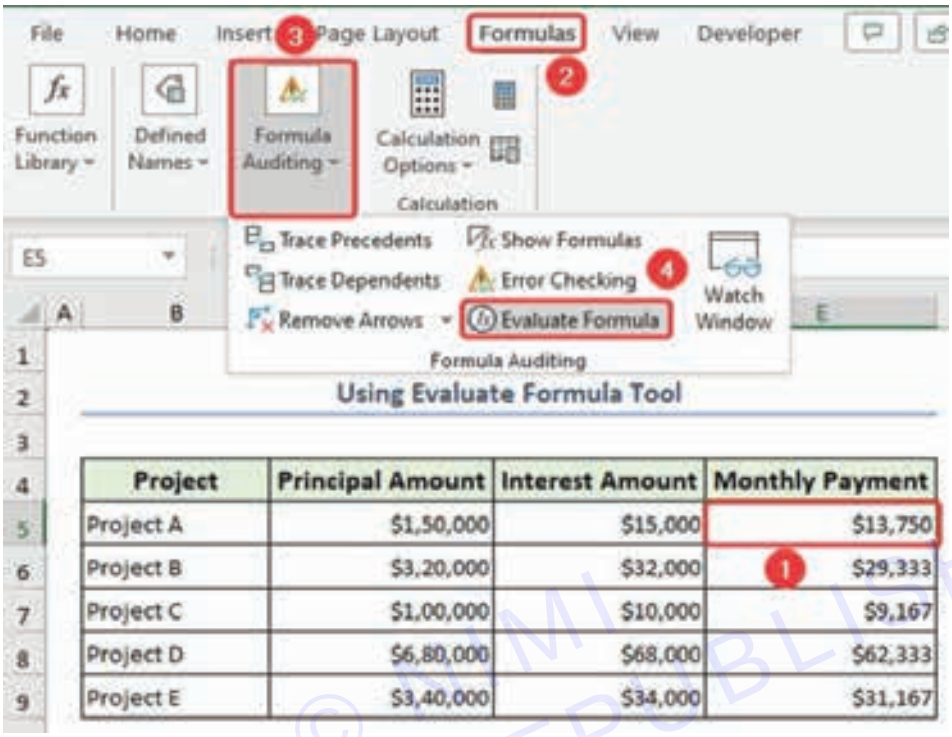
- Immediately, an arrow will appear to indicate the error and its corresponding cells.



Formula evaluator

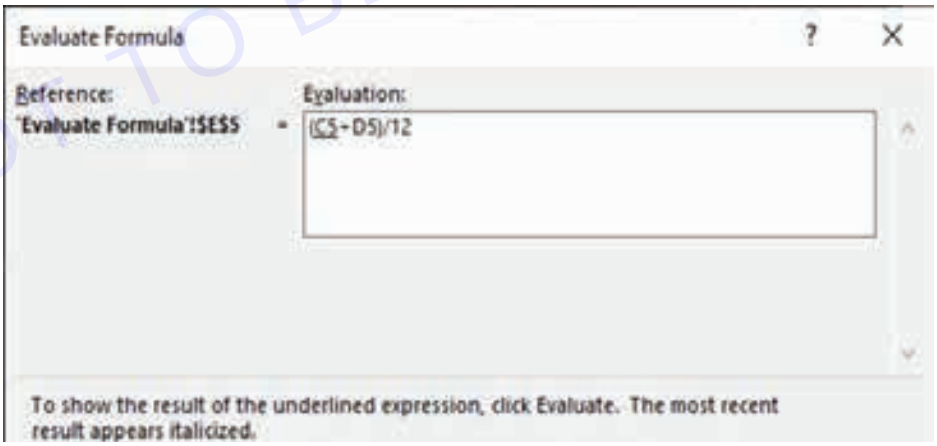
When you are dealing with complex formulas and you are having trouble understanding the formulas, at that time you can visit the Evaluate Formula option to have a better understanding.

- Simply, choose a cell (E5) consisting of the formula and hit the Evaluate Formula option.



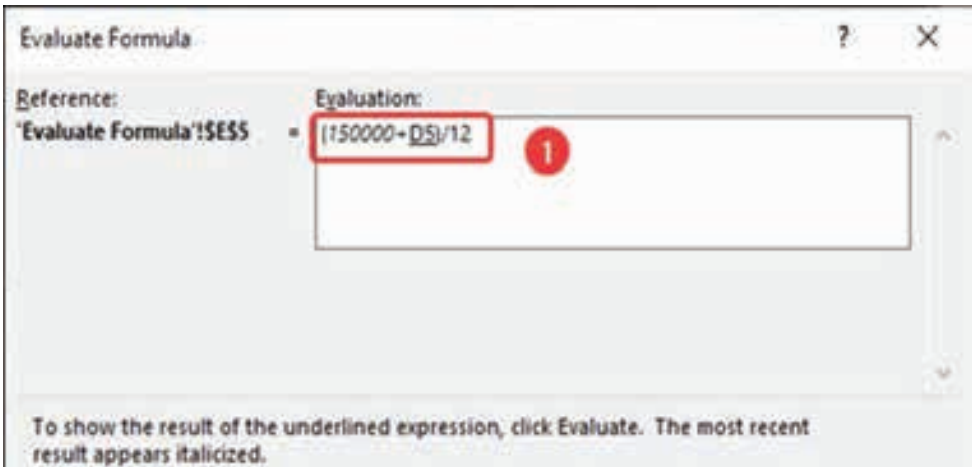
Project	Principal Amount	Interest Amount	Monthly Payment
Project A	$1,50,000	$15,000	$13,750
Project B	$3,20,000	$32,000	$29,333
Project C	$1,00,000	$10,000	$9,167
Project D	$6,80,000	$68,000	$62,333
Project E	$3,40,000	$34,000	$31,167

- Immediately, a window will open evaluating the formula.
- From there to evaluate more deeply click the Evaluate option.

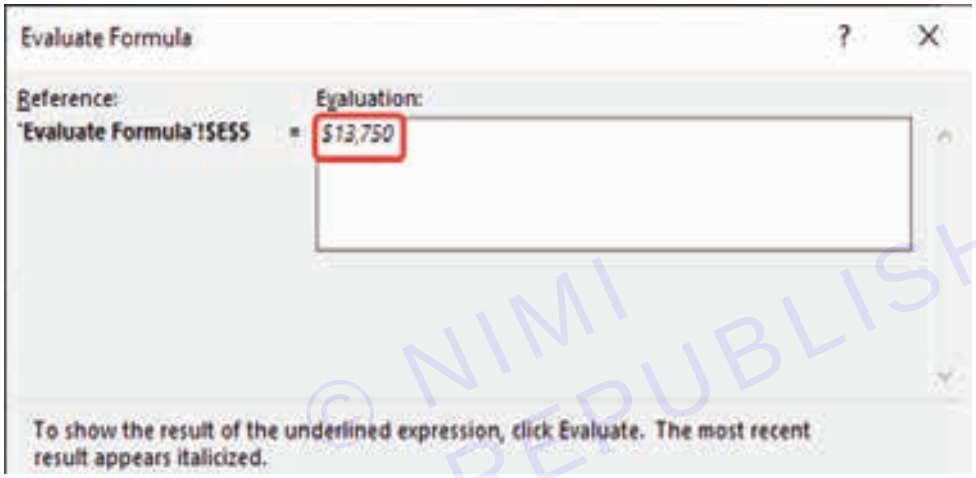


Another update will come describing the formula. Click Evaluate again to get the result.

After completing the evaluation, you will get the cell value in the window.



After completing the evaluation, you will get the cell value in the window.

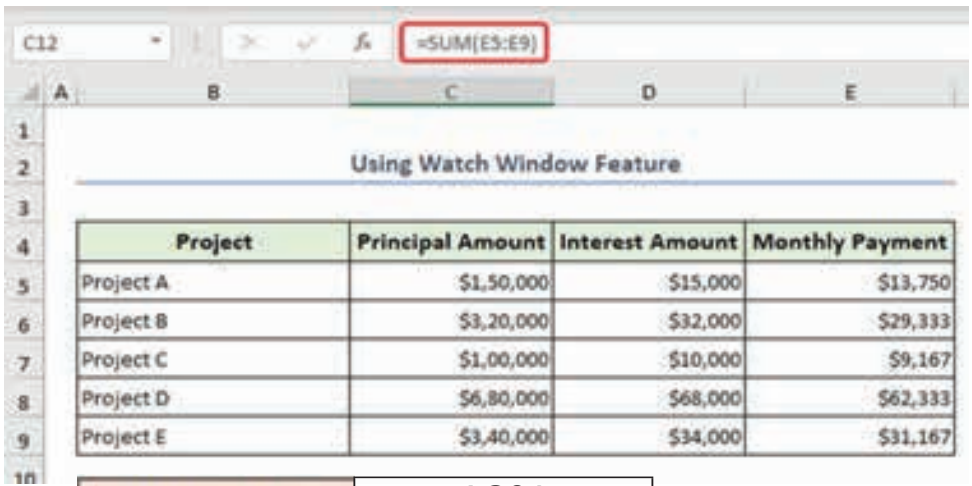


Watch Window

While working with a large dataset sometimes you might need to look over some cell values immediately and all time in a specific space. For that, you can add a watch window at the top of the spreadsheet. Here, we have calculated the total monthly payment amount using the SUM function in Excel.

Now we will add a watch window for this specific cell.

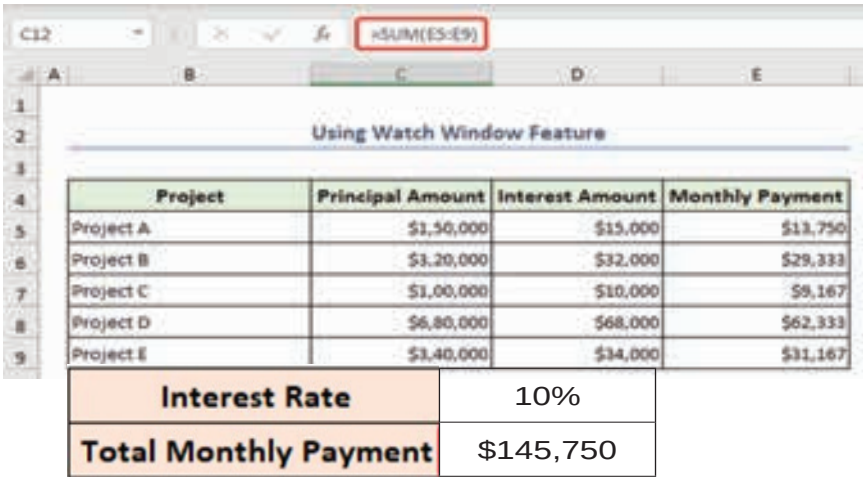
- Choose a cell (C12), apply the below formula, and hit ENTER.

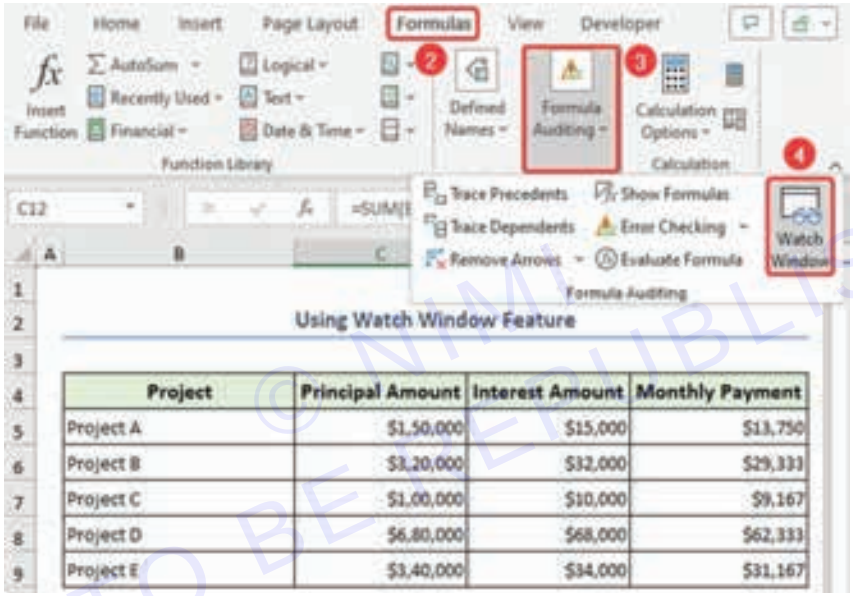


Project	Principal Amount	Interest Amount	Monthly Payment
Project A	$1,50,000	$15,000	$13,750
Project B	$3,20,000	$32,000	$29,333
Project C	$1,00,000	$10,000	$9,167
Project D	$6,80,000	$68,000	$62,333
Project E	$3,40,000	$34,000	$31,167

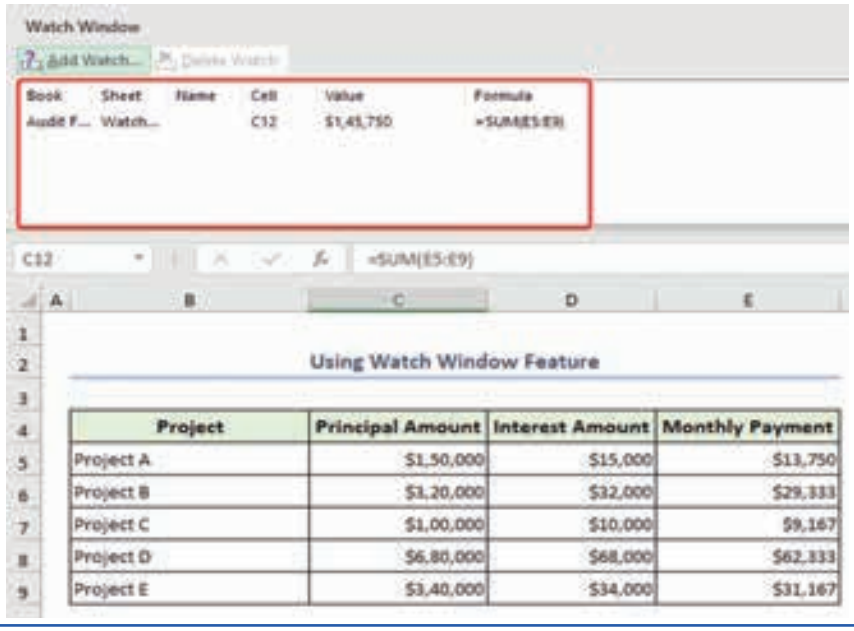
Interest Rate	10%
Total Monthly Payment	$145,750

- Now choosing the cell (C12), hit the Watch Window feature from the Formulas tab.





- Within a glimpse, a window will open at the top of the spreadsheet showing cell value and formula. This watch window is really a helpful tool for making a summary of your dataset. And if you scroll or jump to another sheet the watch window will always be visible on that place.



Array Formulas in Excel

Excel is among the most powerful and widely used spreadsheet tools which eases organizing numbers and data within the sheets with pre-built formulas and functions. In this article, we are going to study Array Formulas. Arrays are one of them. Using Excel Array Formulas, make it super easy to handle a lot of data. Before understanding and using the Array formulas, let's first understand what is Array in Excel?

Array in Excel

An Array in Excel is the structure that holds a collection of data. These data can be in the form of numbers or texts. There could be an array of 1-D or 2-D as well in Excel. For example, the array of fruits in excel would be written something like this:-

{"Apple", "Banana", "Grapes", "Guava"}

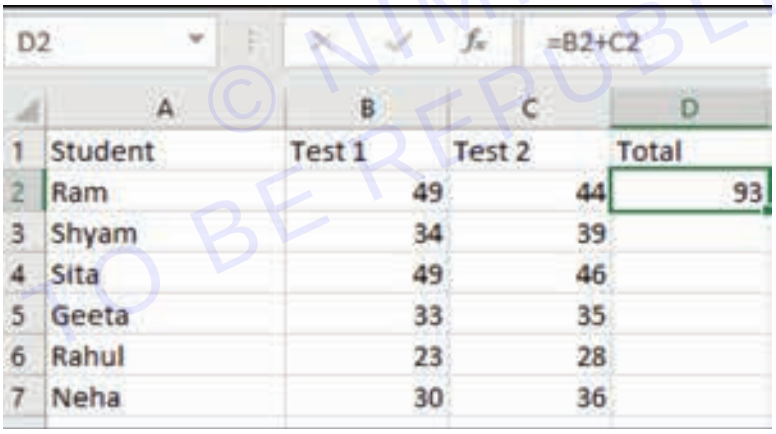
We can directly enter it into the sheet by selecting the cells where you want to enter the data and then writing the array preceded by an "=" sign. After these press Ctrl+Shift+Enter. For example, let's enter these values in cells A1 to D1.

Array-in-excel

The array created here is a horizontal array. You can also create a vertical array by changing the selected cells. You can create an array by specifying the starting and the ending positions, for example, if you wish to make an array of elements between A1 to F1 then you can write A1:F1. Now. let's move to array formulas.

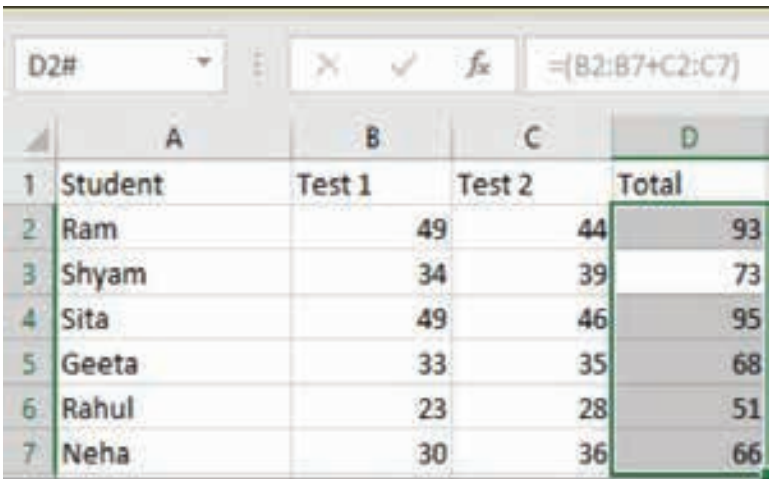
Array formulas

Array formulas enable to process of several values and give several outputs. In a simple world, it can do multiple calculations and reduce a lot of human efforts. In other words, we can say that it solves array calculations and give an array as output too. Let's understand more of them with an example, of students with their test marks,



	A	B	C	D
1	Student	Test 1	Test 2	Total
2	Ram	49	44	93
3	Shyam	34	39	
4	Sita	49	46	
5	Geeta	33	35	
6	Rahul	23	28	
7	Neha	30	36	

Here, we have used a formula to evaluate the result in the 'Total' field. For inserting the formula, click on the cell where you want to store the answer and write the formula. Now, let's try to calculate the same for all students



	A	B	C	D
1	Student	Test 1	Test 2	Total
2	Ram	49	44	93
3	Shyam	34	39	73
4	Sita	49	46	95
5	Geeta	33	35	68
6	Rahul	23	28	51
7	Neha	30	36	66

For calculating the total for all the students, select the field where you want to store them, now select the range, and insert the formula. Array formulas are available in every version of Excel, so you don't worry about your excel version. There are hundreds of operations possible using array formulas, which we will discuss in this article later. But before proceeding let's understand what is the need to use an array formula.

Why use an Array Formula?

Array formulas are to most powerful yet most easy-to-use calculation tools which could be used to perform complex calculations and take place of hundreds of formulas with a single formula. You can further specify conditions for the calculation of data. Not just calculations you can also count the chars in cells, add date, day, or time, and even pick random data.

How to insert an Array Formula?

Before entering the Array formula fir of all we understand some important points about the Array formula:

- After typing the formula you have to press keys CTRL+SHIFT+ENTER together. It will automatically change the normal formula into an array formula.
- If you manually type braces around the formula then it will not convert the formula into an array formula. You have to use CTRL+SHIFT+ENTER keys to convert the formula into an array formula.
- Whenever you edit the array formula the braces will disappear automatically and you have to again press the combination of CTRL+SHIFT+ENTER keys.
- If you forget to press the CTRL+SHIFT+ENTER keys then your formula will work as the normal formula.

Now let's understand inserting an array formula.

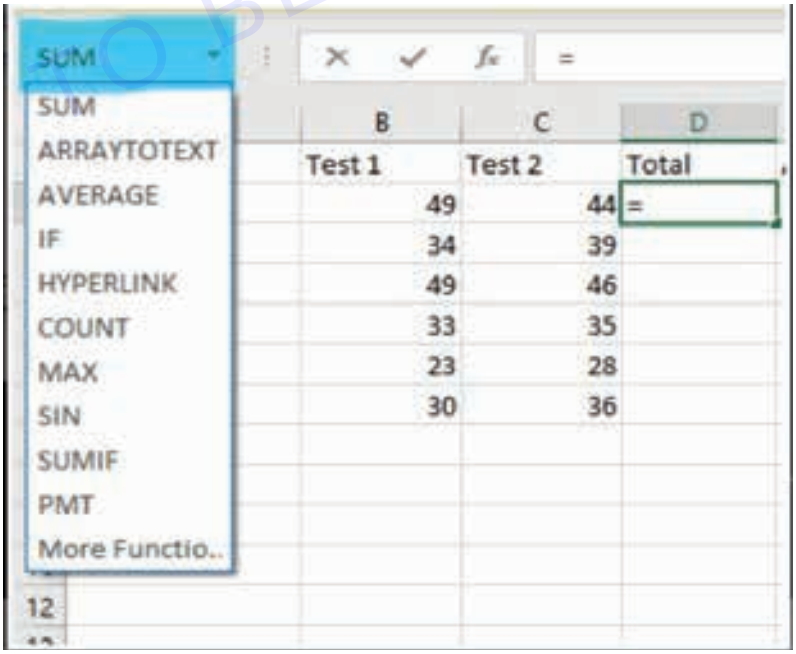
Single Cell Array formula

Every array formula returns the result in either single or in multiple cells. So when an array formula returns its result in the single-cell then such type of formula is known as a single-cell array formula. Such types of formulas are SUM, AVERAGE, AGGREGATE, MAX, MIN, etc. Follow the following steps to use the single-cell array formula:

Step 1: Select the cell where you want to store the answer.

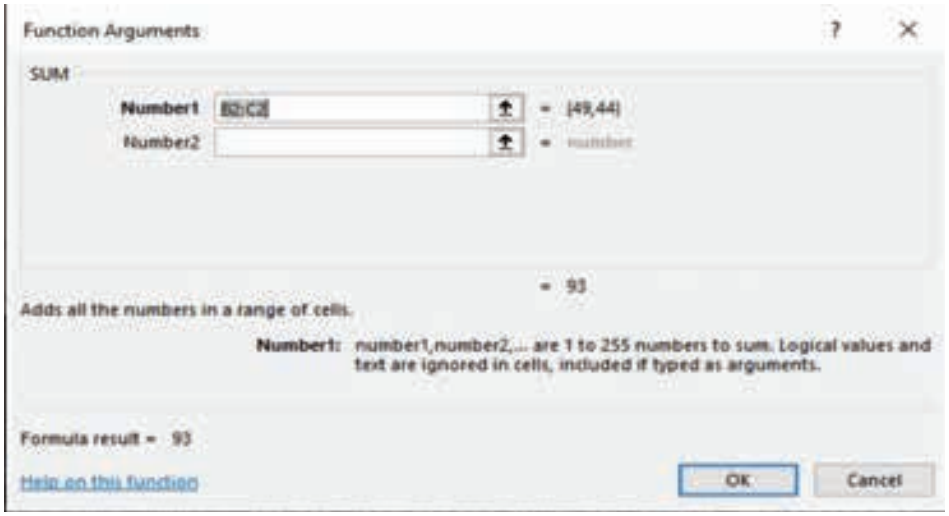
Step 2: Now go to the formula bar and write an "=" sign.

Step 3: Now to formulas (highlighted below), and select the operation you want to perform.



Single-cell-array

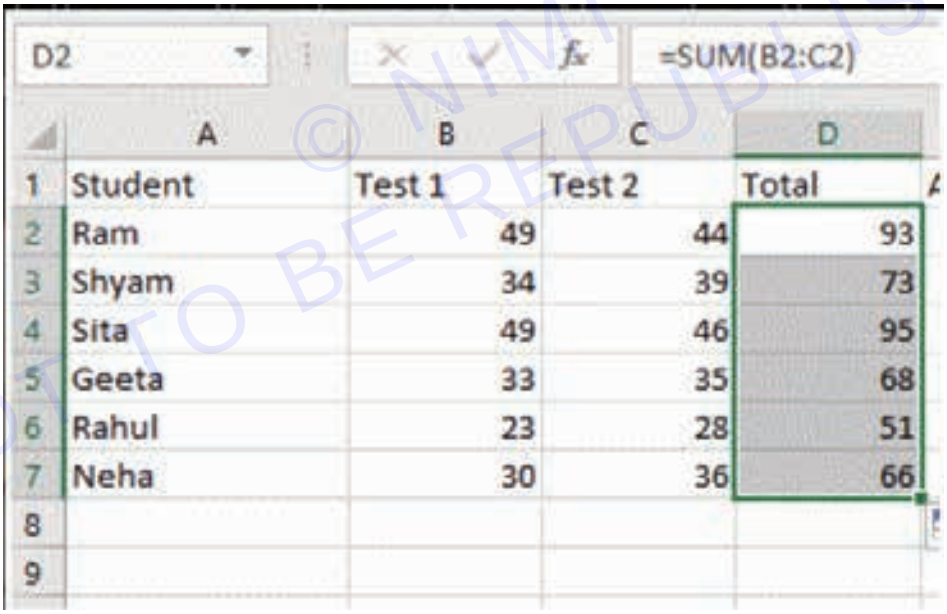
Step 4: Here we want to perform the sum operation. Click on sum and a new popup will open. Write the cells you want to perform the operation with.



Function-arguments

Multi-Cell Array Formula

If you wish to perform the same calculation with more cells then you can use the multi-cell formula. You can implement it in two ways. Functions like TRANSPOSE, TREND, FREQUENCY, LINEST, etc are used for the multi-cell arrays. Implement the formula in single-cell by following the above steps. Now, drag the selection up to the cells you want to implement the formula. Below is the implementation of doing that.



Multi-cell-array-formula

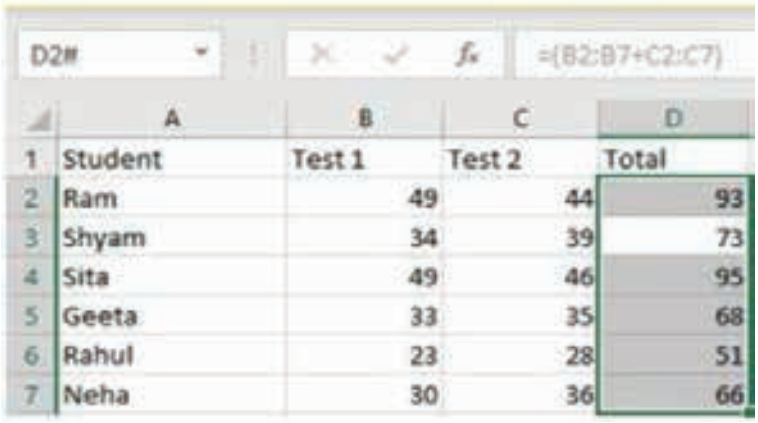
Alternatively,

Specify the range of cells and an operator between them. For example, =(B1:B7 + C2:C7) and press Ctrl+Shift+Enter.

Operator-in-multi-cell

Excel Array Constants

Excel array contacts are a set of static values and can't be changed. These values remain constant regardless of the operations performed on them. Now we will see how to create Array Constants in Excel and will perform different actions on them. There are three types of array constant available:



1 Horizontal or Row Array constant

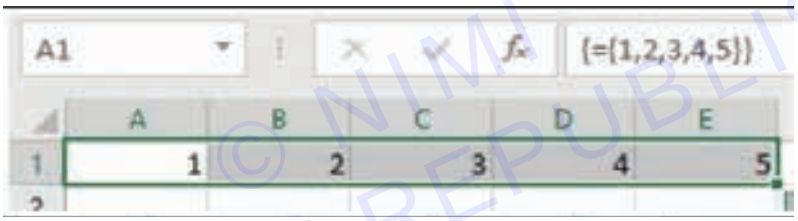
A horizontal constant is present in a row. To create a horizontal constant we have to type multiple values separated by a comma in the enclosed braces. To enter a row array follow these steps below:

Step 1: Select the cells you want to work with.

Step 2: Go to the formula bar and start with an equal sign.

Step 3: Now enter the array you want to create in braces {}, separated by commas(,). Ex:- ={1,2,3,4,5}

Step 4: Now press Ctrl+Shift+Enter.



Horizontal-or-Row-Array-constant

2 Vertical or Column Array

Vertical array constants are generally present in Column Array. To create a vertical constant we have to type multiple values separated by a semicolon in the enclosed braces. To enter a column array follow these steps below:

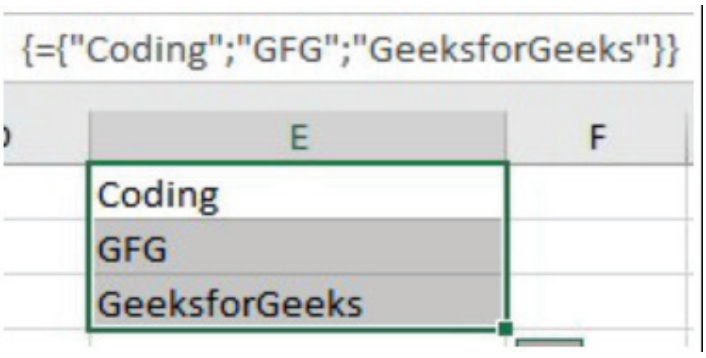
Step 1: Select the cells you want to work with.

Step 2: Go to the formula bar and start with an equal sign.

Step 3: Now enter the array you want to create in braces {} separated with termination sign(;). Ex:- ={“Coding”;” GFG”;”GeeksforGeeks”}

Step 4: Now press Ctrl+Shift+Enter.

Vertical-or-Column-Array



3 2-D array

In Excel, you are allowed to create a 2-D array. So to create a 2-D array constant, we have to type multiple values in which rows are separated by semicolons and d columns are separated by commas. Let's understand the implementation of a 2-d array.

Step 1: Select the cells you want to work with.

Step 2: Go to the formula bar and start with an equal sign.

Step 3: Now enter the array you want to create in braces {} separated with commas(,) for entering the values in same row, termination sign(;) for row change. Ex:- ={1,2,3;"Coding","GFG","GeeksforGeeks"}

Step 4: Now press Ctrl+Shift+Enter.

Working with Constant arrays

In the Excel formula, array constants are the important part. Some of the important concepts of Array constant are:

1 Elements: To create elements of array constant we have to follow the following points:

   1 The elements of an array constant must be of numeric, alphabetic, boolean, or scientific notation and each of them is separated by commas or semicolons.

   2 The elements of an array constant do not contain an array, cell reference, dates functions formulas, defined names, etc.

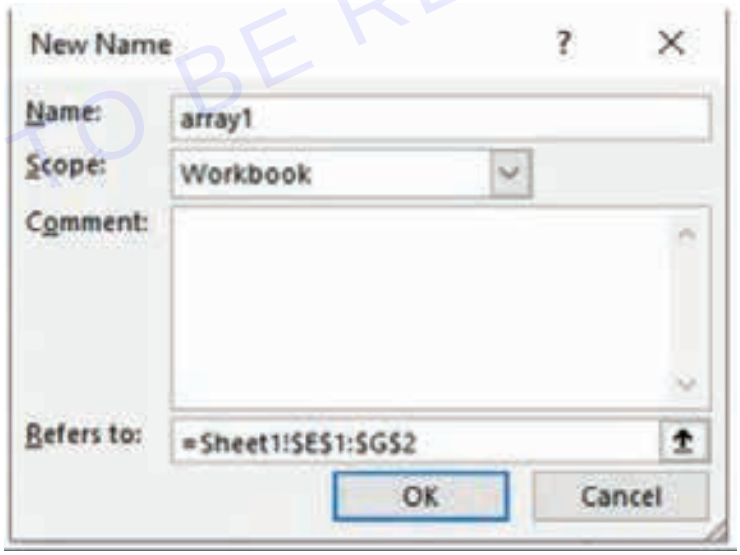
   3 You can also be allowed to use text in the array of the element but the text is surrounded by hyphens("").

2 Naming Array Constants: You can name any array and use them as a variable for further references. For naming an array, follow the steps below:

   1 Go to the formula tab. and select Define Name.

   2 Write the array name in the name. You can also add some comments related to your array.

   3 Click on OK.



3 Common Errors: Following are the common errors that will happen while working with array constants:

   1 The use of delimiters in between the elements.

   2 The selection of range is also an important point to remember.

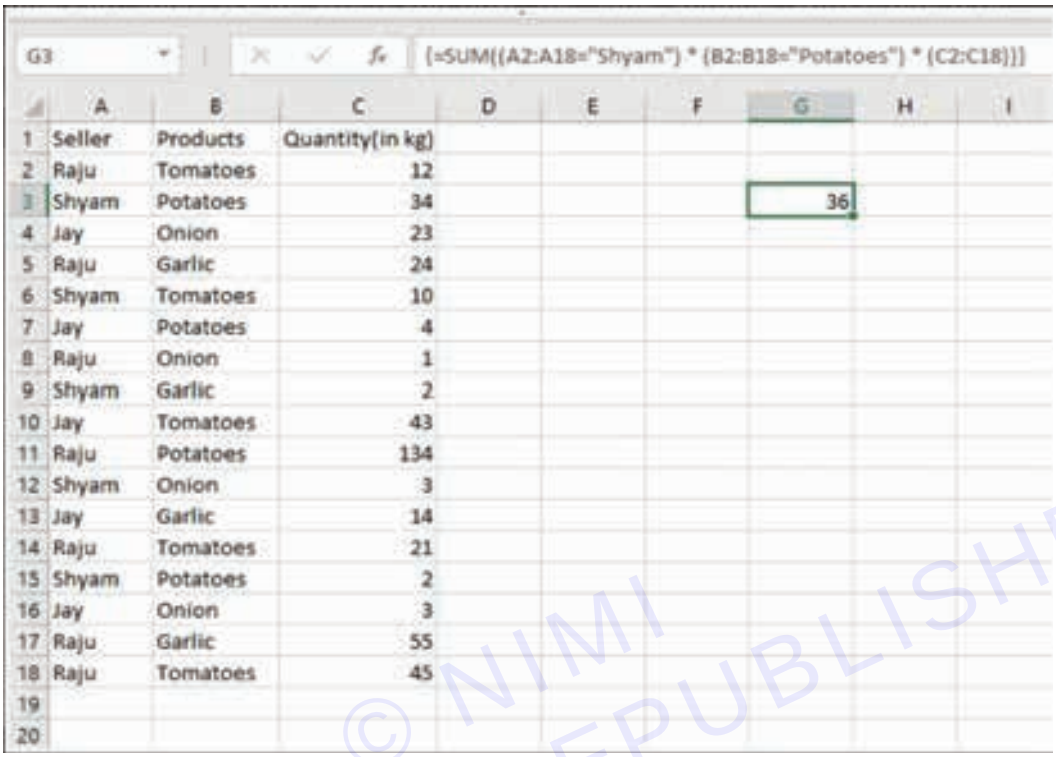
Unary Operator in Excel Array Formulas

Using unary operators we can process the array using AND and OR operators.

1 **AND:** It is denoted by an asterisk(*) symbol. It returns true if and only if all conditions are true,

2 **OR:** It is denoted by a plus(+) symbol. It returns true if any of the statement is true.

These operators are more useful when you have the same variable with a different value. For example, let's consider a case, where data of 3 sellers of different months are given and you have to how much Shyam sold potatoes.



	A	B	C	D	E	F	G	H	I
1	Seller	Products	Quantity(in kg)						
2	Raju	Tomatoes	12						
3	Shyam	Potatoes	34				36		
4	Jay	Onion	23						
5	Raju	Garlic	24						
6	Shyam	Tomatoes	10						
7	Jay	Potatoes	4						
8	Raju	Onion	1						
9	Shyam	Garlic	2						
10	Jay	Tomatoes	43						
11	Raju	Potatoes	134						
12	Shyam	Onion	3						
13	Jay	Garlic	14						
14	Raju	Tomatoes	21						
15	Shyam	Potatoes	2						
16	Jay	Onion	3						
17	Raju	Garlic	55						
18	Raju	Tomatoes	45						
19									
20									

Using-and-operator

Lookup formulasVlookup, Lookup, Index, Hlookup

Before we dive into the arcane twists of Excel Lookup formulas, let's define the key terms to ensure that we are always on the same page.

Lookup - searching for a specified value in a table of data.

Lookup value - a value to search for.

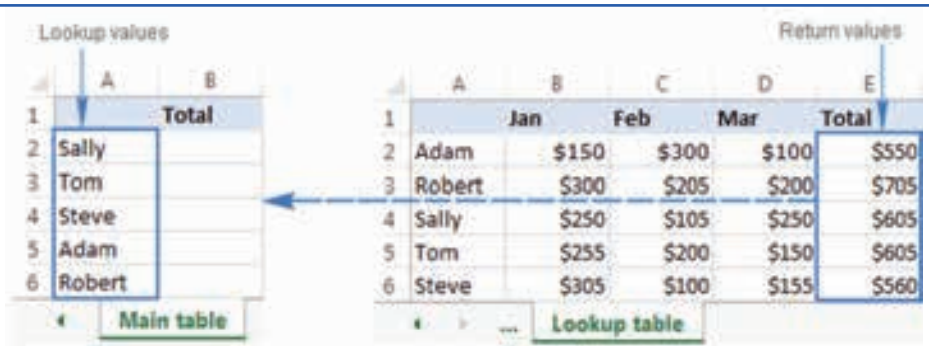
Return value (matching value or match) - a value at the same position as the lookup value but in another column or row (depending on whether you do vertical or horizontal lookup).

Lookup table. In computer science, a lookup table is an array of data, which is generally used to map input values to output values. In terms of this tutorial, an Excel lookup table is nothing else but a range of cells where you search for a lookup value.

Main table (master table) - a table into which you pull matching values.

Your lookup table and main table may have different structure and size, however they should always contain at least one common unique identifier, i.e. a column or row that holds identical data, depending on whether you want to perform a vertical or horizontal lookup.

The following screenshot shows a sample lookup table that will be used in many of the below examples.



Excel Lookup functions

Below is a quick overview of the most popular formulas to perform lookup in Excel, their main advantages and drawbacks.

LOOKUP function

The LOOKUP function in Excel can perform the simplest types of vertical and horizontal lookups.

Pros: Easy-to-use.

Cons: Limited functionality, cannot work with unsorted data (requires sorting the lookup column/row in ascending order).

For more information, please see How to use Excel

LOOKUP function.

VLOOKUP function

It's an improved version of the LOOKUP function specially designed to do vertical lookup in columns.

Pros: Relatively easy to use, can work with exact and approximate match.

Cons: Cannot look at its left, stops working when a column is inserted into or removed from the lookup table, a lookup value cannot exceed 255 characters, requires much processing power on large datasets.

For more information, please see Excel VLOOKUP tutorial for beginners.

HLOOKUP function

It's a horizontal counterpart of VLOOKUP that searches for a value in the first row of the lookup table and returns the value in the same position from another row.

Pros: Easy to use, can return exact and approximate matches.

Cons: Can only search in the topmost row of the lookup table, is affected by the insertion or deletion of rows, a lookup value should be under 255 characters.

For more information, please see How to use HLOOKUP in Excel.

VLOOKUP MATCH / HLOOKUP MATCH

A dynamic column or row reference created by MATCH makes this Excel lookup formula immune to the changes made in the dataset. In other words, with some help from MATCH, the VLOOKUP and HLOOKUP functions can return correct values no matter how many columns/rows have been inserted to or deleted from a lookup table.

Formula for Vertical lookup

VLOOKUP(lookup_value, lookup_table,

MATCH(return_column_name, column_headers, 0), FALSE)

Formula for Horizontal lookup

HLOOKUP(lookup_value, lookup_table,

MATCH(return_row_name, row_headers, 0), FALSE)

Pros: An improvement over regular Hlookup and Vlookup formulas immune to data insertion or deletion.

Cons: Not very flexible, requires a specific data structure (the lookup value supplied to the MATCH function should be exactly equal to the name of the return column), cannot work with lookup values exceeding 255 characters.

For more information and formula examples, please see: Excel Vlookup and Match

Excel Hlookup and Match

OFFSET MATCH

A more complex but a more powerful lookup formula, free of many limitations of Vlookup and Hlookup.

Formula for VLookup

OFFSET(lookup_table, MATCH(lookup_value,

OFFSET(lookup_table, 0, n, ROWS(lookup_table), 1) ,0) -1,

m, 1, 1)

Where:

n - is the lookup column offset, i. e. the number of columns to move from the starting point to the lookup column.

m - is the return column offset, i. e. the number of columns to move from the starting point to the return column.

Formula for HLookup

OFFSET(lookup_table, m, MATCH(lookup_value,

OFFSET(lookup_table, n, 0, 1, COLUMNS(lookup_table)), 0) - 1, 1, 1)

Where:

n - is the lookup row offset, i. e. the number of rows to move from the starting point to the lookup row.

m - is the return row offset, i. e. the number of rows to move from the starting point to the return row.

Formula for matrix lookup (by row and column)

Please pay attention that this is an array formula, which is entered by pressing Ctrl + Shift + Enter keys at the same time.

Pros: Allows performing a left-side Vlookup, an upper Hlookup and two- way lookup (by column and row values), unaffected by changes in the data set.

Cons: Complex and difficult to remember syntax.

For more information and formula examples, please see: Using OFFSET function in Excel

INDEX MATCH

It's the best way to do vertical or horizontal lookup in Excel that can replace most of the above formulas. The Index Match formula is my personal preference and I use it for almost all of my Excel lookups.

Formula for V-Lookup

INDEX (return_column, MATCH (lookup_value, lookup_column, 0))

Formula for H-Lookup

INDEX (return_row, MATCH (lookup_value, lookup_row, 0))

Formula for matrix lookup

An extension of the classic Index Match formula to return a value at the intersection of a specific column and row:

INDEX (lookup_table, MATCH (vertical_lookup_value, lookup_column, 0), MATCH (horizontal_lookup_value, lookup_row, 0))

Cons: Just one - you need to remember the formula's syntax. Pros: The most versatile Lookup formula in Excel, superior to Vlookup, Hlookup and Lookup functions in many respects:

It can do left and upper lookups.

Allows safely extending or collapsing the lookup table by inserting or deleting columns and rows.

No limit to the lookup value's size.

Works faster. Because an Index Match formula references columns/rows rather than an entire table, it requires less processing power and won't slow down your Excel.

For more information, please check out:

INDEX MATCH as a better alternative to VLOOKUP

INDEX MATCH formula for two-dimensional lookup

Excel Lookup comparison table

Formula	Vertical lookup	Left lookup	Horizontal lookup	Upper lookup	Matrix lookup	Allows data insertion/deletion
Lookup	✓		✓			
Vlookup	✓					
Hlookup			✓			
Vlookup Match	✓					✓
Hlookup Match			✓			✓
Offset Match	✓	✓	✓	✓		✓
Offset Match Match					✓	✓
Index Match	✓	✓	✓	✓		✓
Index Match Match					✓	✓

As you see, not all Excel Lookup formulas are equivalent, some can handle a number of different lookups while others can only be used in a specific situation. The table below outlines the capabilities of each Lookup formula in Excel.

Excel Settings in Registry

The Windows Registry is essentially a central hierarchical database that is used by the operating system and by application software. The Registry first appeared in Windows 95 and replaces the old INI files that stored Windows and application settings.

You can use the Registry Editor program to browse the Registry - and even edit its contents if you know what you're doing. The Registry Editor is named regedit.exe. Before beginning your explorations, take a minute to read the upcoming sidebar (titled "Before You Edit the Registry "). Figure 4-9 shows what the Registry Editor looks like.

The Registry Editor lets you browse and make changes to the Registry.

The Registry consists of keys and values, arranged in a hierarchy. The top-level keys are:

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG
- HKEY_DYN_DATA

Excel's settings:

Information used by Excel 2007 is stored in this Registry section:

HKEY_CURRENT_USER\Software\Microsoft\Office.0\Excel

Before You Edit the Registry:

You can use the regedit.exe program to change anything in the Registry, including information that is critical to your system's operation. In other words, if you change the wrong piece of information, Windows may no longer work properly.

Get into the habit of choosing the File ⇨ Export command in Regedit. This command enables you to save an ASCII version of the entire Registry or just a specific branch of the Registry.

If you find that you messed up something, you can always import the ASCII file to restore the Registry to its previous condition (choose the Registry ⇨ Import Registry File command). Refer to the Help file for Regedit for details. In this section of the Registry, you'll find a number of keys that contain specific values that determine how Excel operates.

The Registry settings are updated automatically by Excel when Excel closes.

Note: It's important to understand that Excel reads the Windows Registry only once - when it starts up. In addition, Excel updates the Registry settings only when Excel closes normally. If Excel crashes (unfortunately, not an uncommon occurrence), the Registry information is not updated. For example, if you change one of Excel's settings, such as the visibility of the Formula bar, this setting is not written to the Registry until Excel closes by normal means.

Table 1 lists the Registry sections that are relevant to Excel 2007. You might not find all these sections in your Registry database.

Table 1: EXCEL CONFIGURATION INFORMATION IN THE REGISTRY

Section	Description
Add in manger	Lists add-ins that appear in the add-ins that are included with excel do not appear in this list.If you have an add-in entry in this list box that you no longer use, you can remove it by using the registry editor
Converters	Lists additional (external) file converters that are not built into excel.
Error checking	Holds the settings for formula error checking.
File MRU	Holds information about the most used files (which appears in the recent documents list when you click the office button)
Options	A catch-all section, holds a wide variety of settings.
Recent templates	Stores the names of templates you've used recently
Resiliency	Information used for recovering documents.
Security	Specifies the security options for opening files that contain macros.
Spell checker	Stores information about your spelling checker options.
Statusbar	Stores the user choices for what appears in the status bar.
Userinfo	Stores information about the user.

Although you can change most of the settings via the Excel Options dialog box, a few settings cannot be changed directly from Excel (but you can use the Registry Editor to make changes). For example, when you select a range of cells, you may prefer that the selected cells appear in high contrast white-on-black. There is no way to specify this in Excel, but you can add a new Registry key like this:

1 Open the Registry Editor and locate this section:

HKEY_CURRENT_USER\Software\Microsoft\Office.0\Excel\Options

2 Right-click and select New DWORD Value.

3 Name this value Options6 .

4 Right-click the Options6 key and select Modify.

5 In the Edit DWORD Value dialog box, click the Decimal option and enter 16

Setting a value for a Registry setting

When you restart Excel, range selections will appear with a black background rather than gray. If you don't like this look, just delete the Options6 Registry entry.

Tip If you have trouble starting Excel, it's possible that the Registry keys have become corrupt. You can try using the Registry Editor to delete the entire Excel section:

HKEY_CURRENT_USER\Software\Microsoft\Office.0\Excel

The next time Excel is started, it will rebuild the Registry keys. You will, however, lose all of the customization information that was stored there.

Advanced Excel tips and tricks

Advanced Excel tips and tricks can greatly enhance your productivity and efficiency when working with complex spreadsheets and large datasets. By mastering techniques such as keyboard shortcuts, conditional formatting, data validation, PivotTables, VLOOKUP and INDEX-MATCH functions, Excel tables, named ranges, Power Query, macros, and array formulas, you can streamline your workflow, automate tasks, perform sophisticated data analysis, and save valuable time. These advanced features empower you to work more efficiently, make informed decisions, and unlock the full potential of Excel for your data-driven tasks.

- Advanced Excel Tips and Tricks for Productivity and Efficiency
- Master Keyboard Shortcuts.
- Conditional Formatting.
- Data Validation.
- PivotTables.
- VLOOKUP and INDEX-MATCH Functions.
- Excel Tables.
- Named Ranges.
- Power Query.
- Macros.
- Array Formulas.

Advanced Excel Tips and Tricks for Productivity and Efficiency

Master Keyboard Shortcuts:

Learn commonly used keyboard shortcuts to perform tasks more efficiently. Some essential shortcuts include:

- Ctrl+C for copy, Ctrl+V for paste, Ctrl+Z for undo, and Ctrl+S for save.
- Ctrl+X for cut, Ctrl+B for bold, Ctrl+I for italic, and Ctrl+U for underline.
- Ctrl+Home to go to the beginning of the worksheet and Ctrl+End to go to the last cell with data.
- Ctrl+Page Up to switch between worksheets and Ctrl+Tab to cycle through open workbooks.

Conditional Formatting:

Utilize conditional formatting to visually highlight and format cells based on specific conditions or criteria. This feature helps you identify patterns, outliers, or important data at a glance. Some advanced techniques include:

- Color scales to represent data ranges using different colors.
- Icon sets to display symbols or icons based on data values.
- Data bars to create horizontal bars proportional to the cell values.
- Formulas to create custom formatting rules based on specific criteria.

Data Validation:

Apply data validation to control and validate the type and range of data entered into cells. This ensures data accuracy and consistency. Some advanced techniques include:

- Drop-down lists to provide predefined choices for data input.
- Custom formulas to create complex validation rules based on specific conditions.
- Input messages to provide instructions or guidance to users when entering data.
- Error alerts to display custom error messages when invalid data is entered.

PivotTables:

Become proficient in creating and using PivotTables to analyze and summarize large datasets. PivotTables allow you to:

- Summarize data by grouping and aggregating values based on different criteria.

- Add calculated fields to perform custom calculations within the PivotTable.
- Use slicers to filter data and easily analyze subsets of information.
- Create Pivot Charts to visualize and explore data interactively.
- Refresh data automatically when the source data changes.

VLOOKUP and INDEX-MATCH Functions:

Master the use of VLOOKUP and INDEX-MATCH functions for advanced data retrieval and analysis. These functions allow you to search for a value in one column and return a corresponding value from another column. Some tips for using these functions effectively are:

- Use the INDEX-MATCH combination when the lookup value is not in the leftmost column of the table.
- Combine multiple criteria by nesting INDEX-MATCH functions.
- Utilize approximate match for finding values within a range.

Excel Tables:

Convert your data range into an Excel table to leverage advanced features and enhance data management. Excel tables provide:

- Dynamic range references that automatically expand when new data is added.
- Automatic formatting and styles to make your data visually appealing.
- Easy sorting and filtering options for quick data analysis.
- Structured referencing in formulas for better readability and maintainability.
- Total rows that automatically calculate sums or other aggregations for each column.

Named Ranges:

Assign names to cell ranges to make formulas more readable and easier to manage. By using named ranges, you can:

- Replace complex cell references with meaningful names in formulas.
- Improve formula transparency and reduce the risk of errors.
- Create more flexible and adaptable formulas that can refer to named ranges across worksheets or workbooks.
- Easily update formulas by modifying the named range instead of manually adjusting cell references.

Power Query:

Power Query is a powerful data transformation and cleaning tool available in newer versions of Excel. It allows you to:

- Import and combine data from various sources, such as databases, CSV files, or websites.
- Perform data cleaning operations, such as removing duplicates, filtering rows, or splitting columns.
- Merge or append multiple datasets into a single table.
- Apply transformations and calculations to shape the data before loading it into Excel.
- Establish a connection to external data sources for automatic data refresh.

Macros:

Use macros to automate repetitive tasks in Excel. Macros are recorded sequences of actions that can be played back with a single click or assigned to a keyboard shortcut. With macros, you can:

- Automate formatting tasks, such as applying specific styles, adjusting column widths, or adding headers and footers.
- Perform data manipulations, such as sorting, filtering, or removing blank rows.
- Create custom functions or calculations that are
- not available in built-in Excel functions.

- Generate reports or charts automatically based on predefined templates.
- Combine multiple actions into a single macro to streamline complex tasks.

Array Formulas:

Array formulas allow you to perform calculations on multiple cells simultaneously. By using array formulas, you can:

- Perform complex calculations, such as array multiplication, matrix operations, or advanced statistical calculations.
- Manipulate multiple cells within a single formula, eliminating the need for intermediate columns or helper calculations.
- Apply conditional logic or perform calculations across multiple dimensions.
- Remember to enter array formulas by pressing Ctrl+Shift+Enter instead of just pressing Enter.

By incorporating these advanced Excel tips and tricks into your workflow, you can significantly increase your productivity and efficiency when working with complex data and tasks. These techniques enable you to streamline processes, automate repetitive tasks, and extract valuable insights from your data more effectively.

What is a Pivot Table

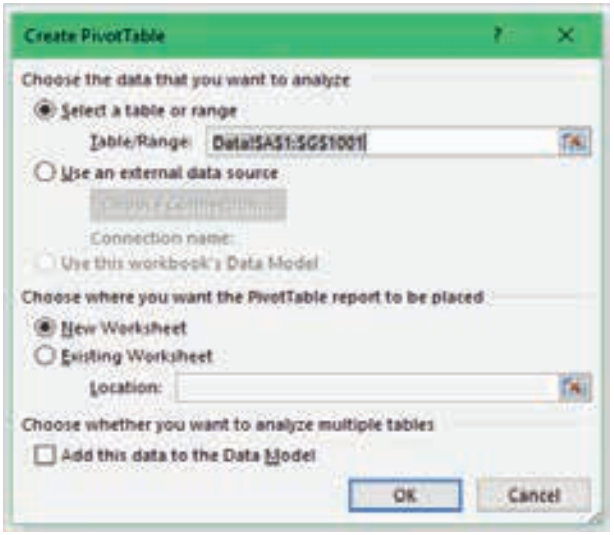
A Pivot Table is a tool in Microsoft Excel that allows you to quickly summarize huge datasets (with a few clicks).

Even if you're absolutely new to the world of Excel, you can easily use a Pivot Table. It's as easy as dragging and dropping rows/columns headers to create reports.

Inserting a Pivot Table in Excel:

Here are the steps to create a pivot table using the data shown above:

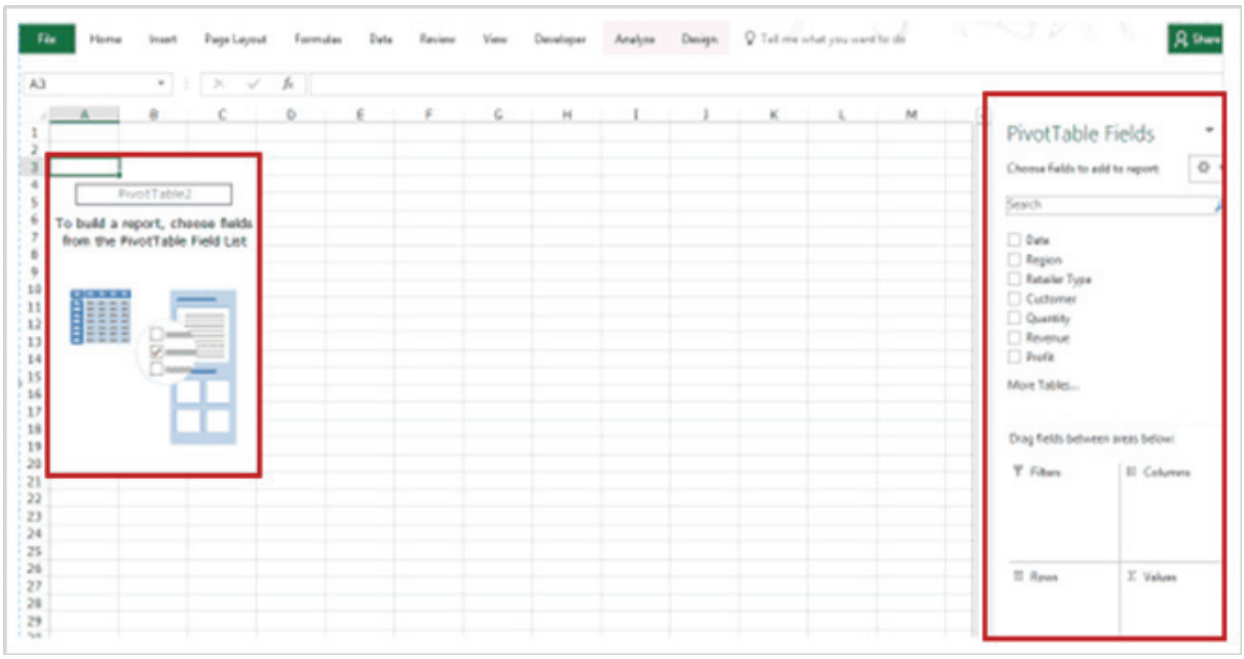
- Click anywhere in the dataset.
- Go to Insert –> Tables –> Pivot Table.
- In the Create Pivot Table dialog box, the default options work fine in most of the cases. Here are a couple of things to check in it:
- Table/Range: It's filled in by default based on your data set. If your data has no blank rows/ columns, Excel would automatically identify the correct range. You can manually change this if needed.
- If you want to create the Pivot Table in a specific location, under the option 'Choose where you want the PivotTable report to be placed', specify the Location. Else, a new worksheet is created with the Pivot Table.



- Click OK.

As soon as you click OK, a new worksheet is created with the Pivot Table in it.

While the Pivot Table has been created, you'd see no data in it. All you'd see is the Pivot Table name and a single line instruction on the left, and Pivot Table Fields on the right.



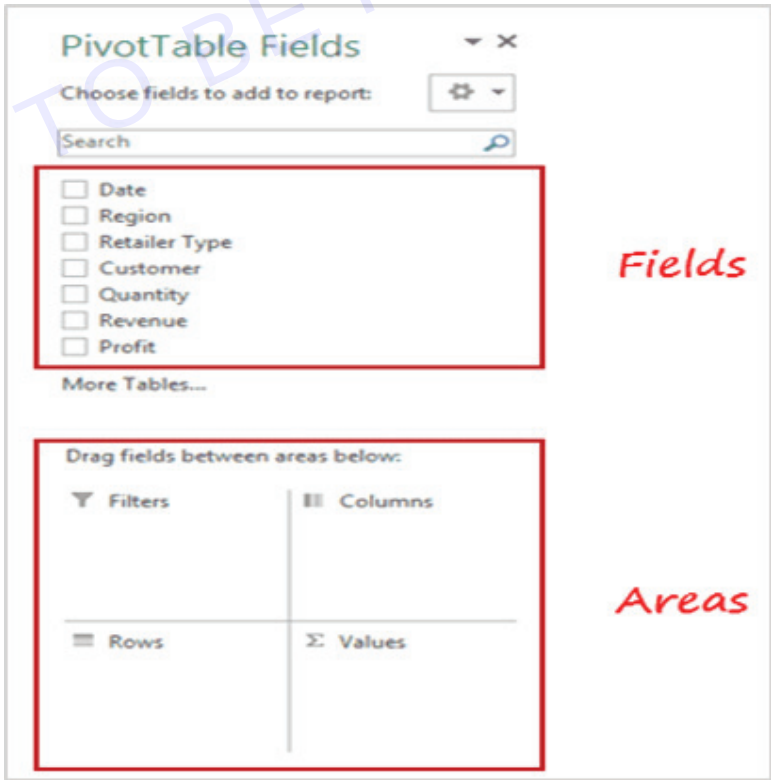
Now before we jump into analyzing data using this Pivot Table, let's understand what are the nuts and bolts that make an Excel Pivot Table.

Analyzing Data Using the Pivot Table:

Now, let's try and answer the questions by using the Pivot

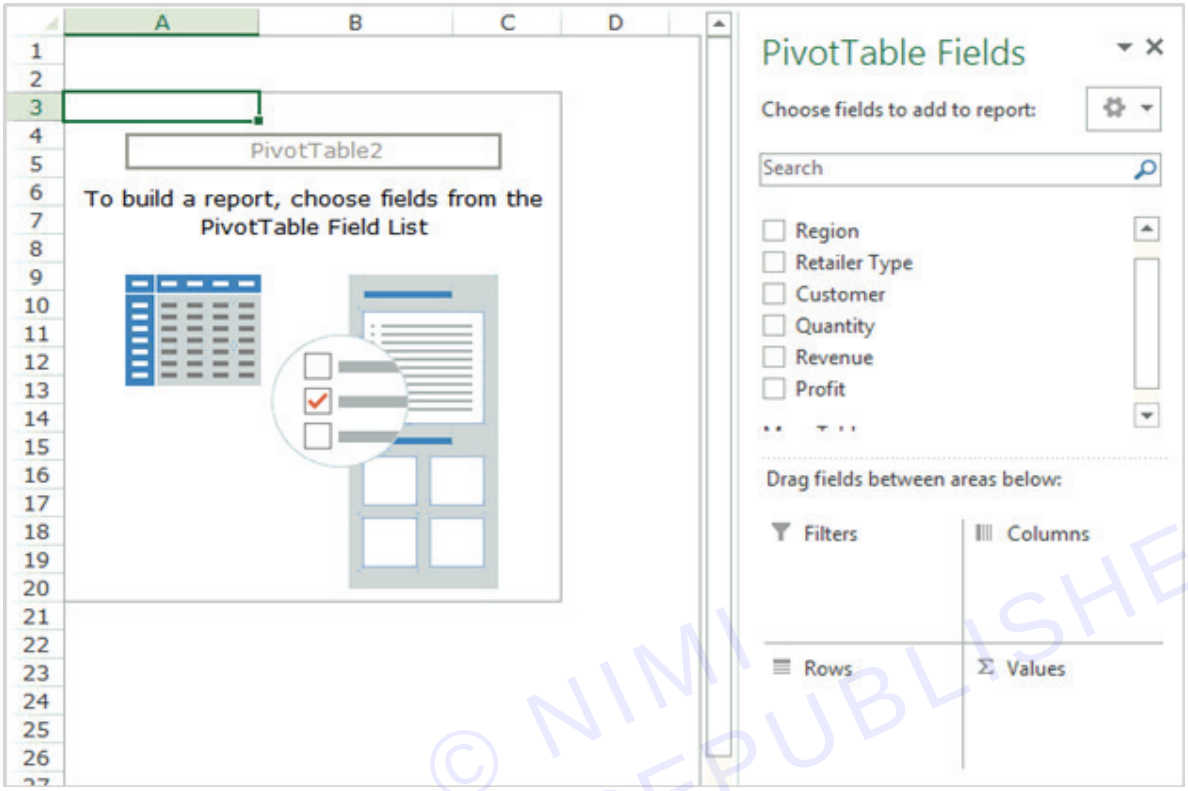
To analyze data using a Pivot Table, you need to decide how you want the data summary to look in the final result. For example, you may want all the regions in the left and the total sales right next to it. You have this clarity in mind, you can simply drag and drop the relevant fields in the Pivot Table.

In the Pivot Table Fields section, you have the fields and the areas (as highlighted below):



The Fields are created based on the backend data used for the Pivot Table. The Areas section is where you place the fields, and according to where a field goes, your data is updated in the Pivot Table.

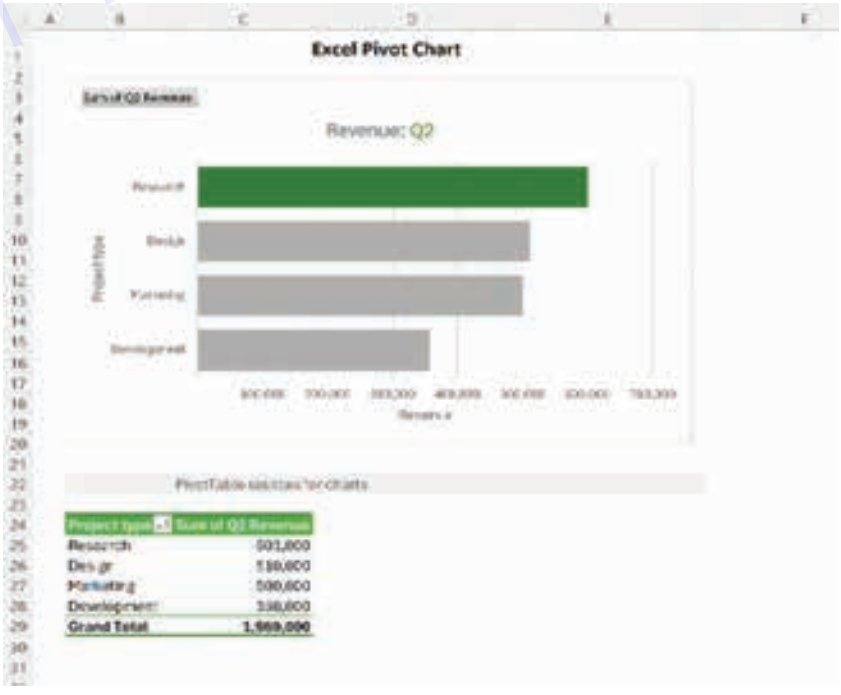
It's a simple drag and drop mechanism, where you can simply drag a field and put it in one of the four areas. As soon as you do this, it will appear in the Pivot Table in the worksheet.



Note that by default, the items (in this case the customers) are sorted in an alphabetical order.

What is pivot chart?

Pivot Chart is a dynamic visualization tool that works together with Excel PivotTables. While PivotTables provide a way to summarize and analyze large datasets, Pivot Charts offer a graphical representation of that summarized data, making trends and patterns easier to spot.



How to make a pivot chart in Excel:

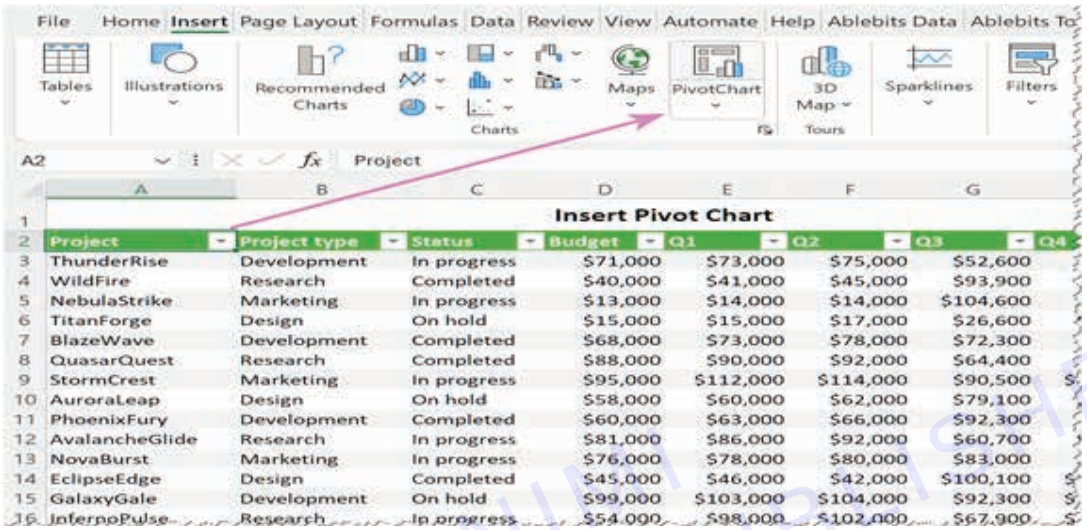
Before you start creating a pivot chart, it's important to make sure your data is well-organized and structured. Each column should represent a different variable or category, and each row should contain a unique record. Don't forget to remove any blank rows or columns and double-check your data for errors or inconsistencies.

Tip. If you want your graph to automatically include new records, then format your source data as a table.

Step 1. Insert a pivot chart

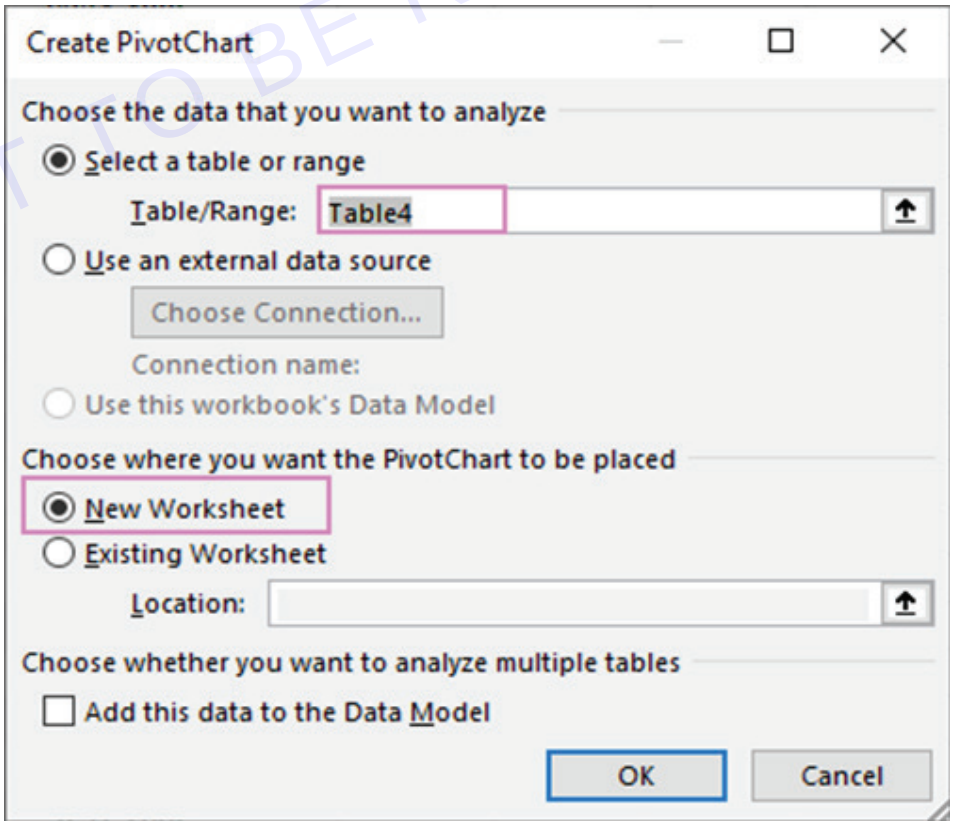
1 Select any cell in your dataset.

2 On the Insert tab, in the Charts group, click PivotChart.



Project	Project type	Status	Budget	Q1	Q2	Q3	Q4
ThunderRise	Development	In progress	$71,000	$73,000	$75,000	$52,600	
WildFire	Research	Completed	$40,000	$41,000	$45,000	$93,900	
NebulaStrike	Marketing	In progress	$13,000	$14,000	$14,000	$104,600	
TitanForge	Design	On hold	$15,000	$15,000	$17,000	$26,600	
BlazeWave	Development	Completed	$68,000	$73,000	$78,000	$72,300	
QuasarQuest	Research	Completed	$88,000	$90,000	$92,000	$64,400	
StormCrest	Marketing	In progress	$95,000	$112,000	$114,000	$90,500	
AuroraLeap	Design	On hold	$58,000	$60,000	$62,000	$79,100	
PhoenixFury	Development	Completed	$60,000	$63,000	$66,000	$92,300	
AvalancheGlide	Research	In progress	$81,000	$86,000	$92,000	$60,700	
NovaBurst	Marketing	In progress	$76,000	$78,000	$80,000	$83,000	
EclipseEdge	Design	Completed	$45,000	$46,000	$42,000	$100,100	
GalaxyGale	Development	On hold	$99,000	$103,000	$104,000	$92,300	
InfernoPulse	Research	In progress	$54,000	$98,000	$102,000	$67,900	

3 The Create PivotChart dialog window will pop up, automatically selecting the entire data range or table. It will then prompt you to choose where to insert your visual - either in a new worksheet or an existing one. Select your preferred location and click OK.



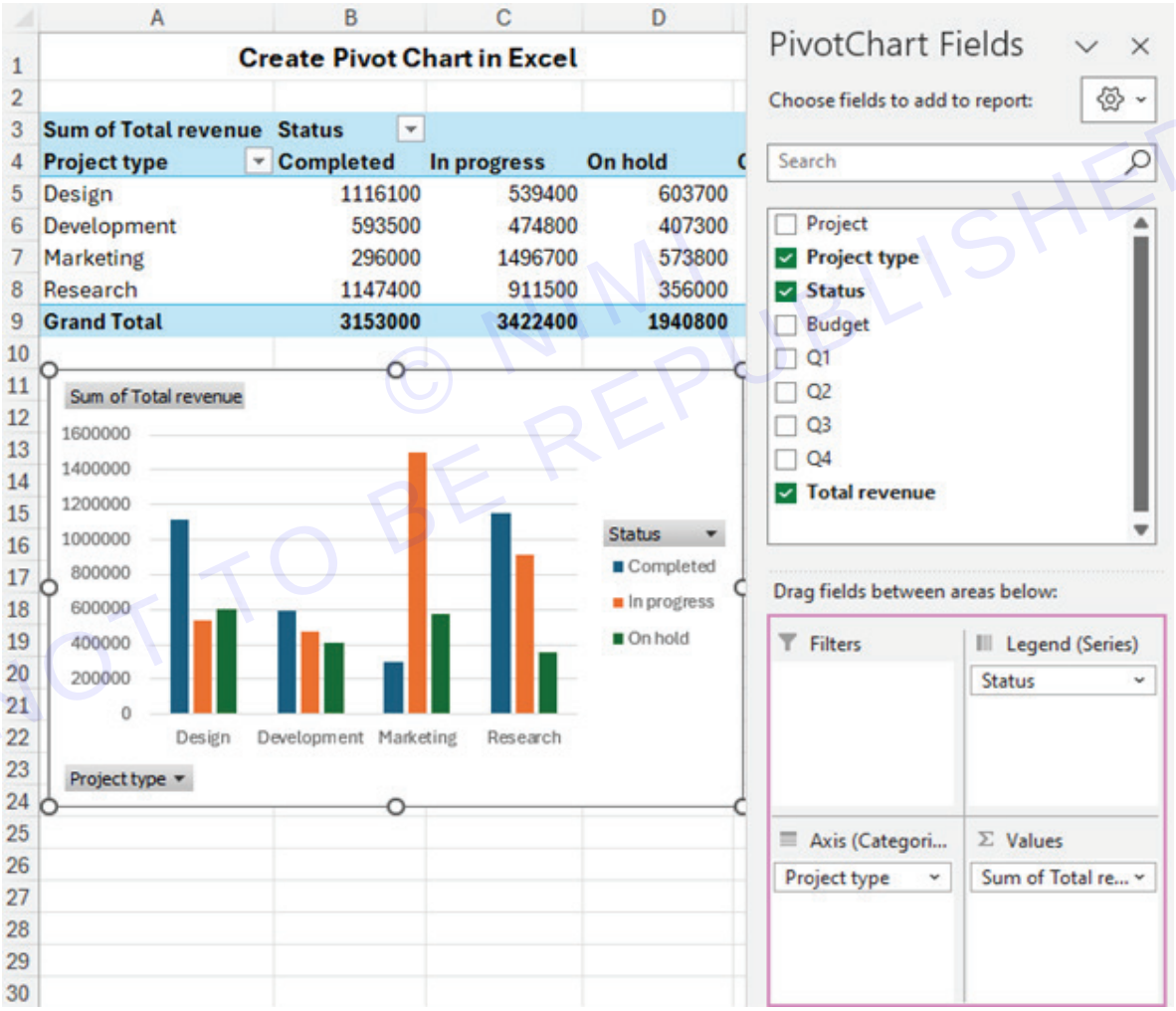
Note. Inserting a pivot chart will automatically include a pivot table alongside it, as these two things are closely related.

Step 2. Configure pivot chart

Now that you have a blank pivot table and pivot chart ready, it's time to set them up to display your data trends. In the PivotChart pane that appears on the left, you'll find a list of all the fields from your dataset. Select the fields you want to display in the graph, and then drag and drop them into the corresponding areas.

- **Filters -** add filters to display or hide certain data.
- **Axis (Categories)** – define what data categories to display along the horizontal axis (placed in pivot table rows).
- **Legend (Series)** – indicate what data series to display (placed in pivot table columns).
- **Values** – define the values that will be depicted in the chart.

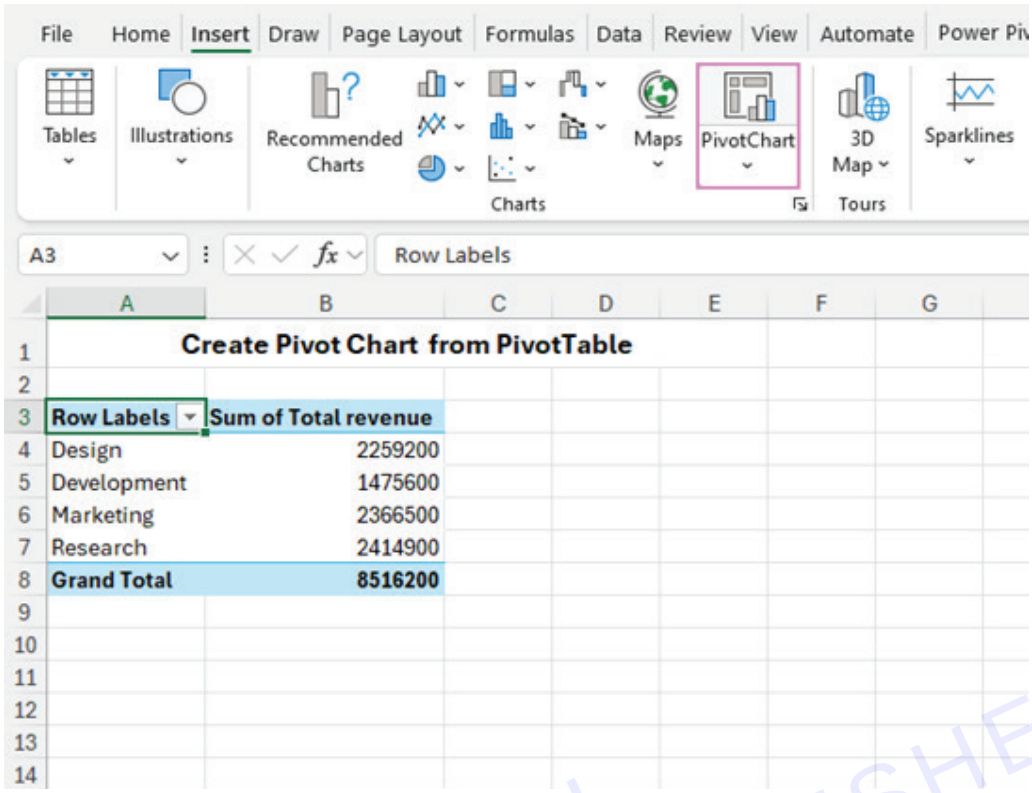
For example, to visualize the total revenue for different project types and statuses, you could drag the "Status" field to the Legend area, "Project type" to the Axis area, and "Total revenue" to the Values area.



How to create a chart from a pivot table

If you already have a pivot table set up, here's how you can easily create a graph from it:

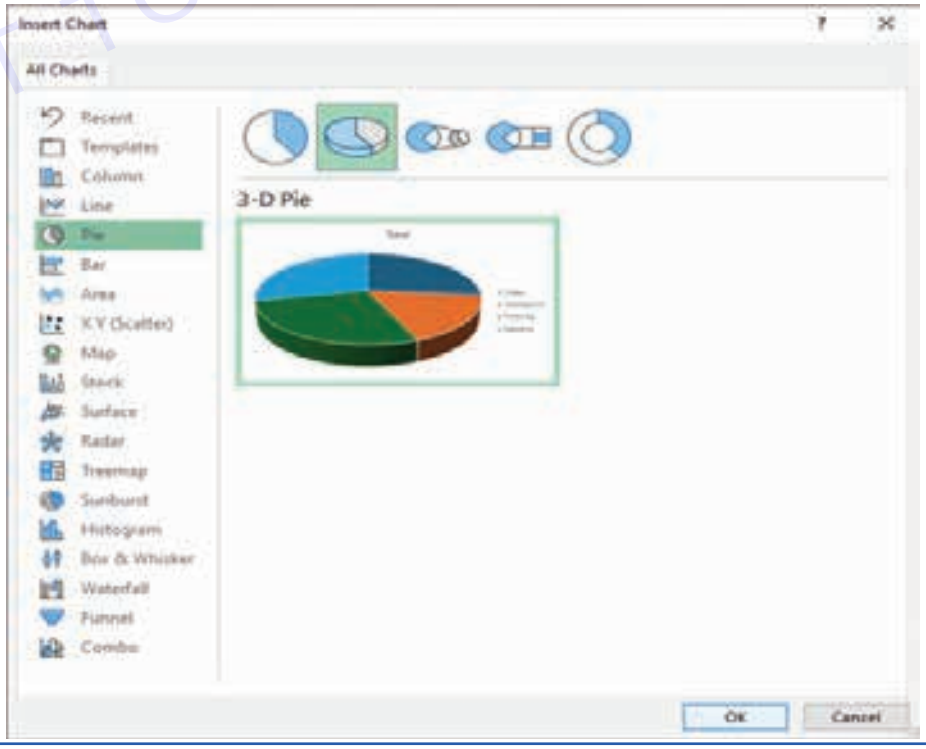
1. Select any cell within your PivotTable.
2. Navigate to the Insert tab on the Excel ribbon and click on the PivotChart button.



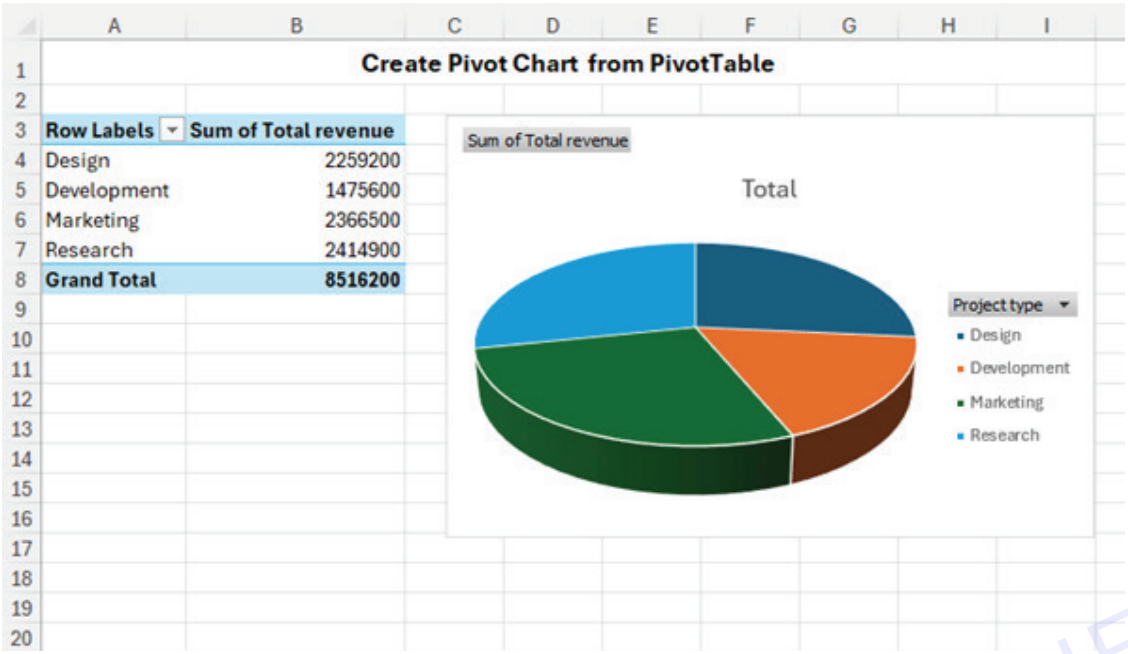
3 In the dialog box that appears, choose the type of graph you want to create. Here are some recommendations:

- Bar charts are ideal for comparing values across different categories or time periods.
- Line charts work well for illustrating trends over time or continuous data series.
- Pie charts are effective for showing the proportion of each category in a dataset.
- Scatter plots are useful for visualizing the relationship between two variables and identifying any correlations or patterns.

4 Once you've selected your desired type, click OK.



5 You will find the visual in the same worksheet next to your PivotTable. You can move it to a different location by dragging and change its size using the resize handles, just as you would with any Other graph in Excel.



Pivot chart shortcut key

To swiftly make a graph based on a pivot table, you can use the following keyboard shortcuts:

1 Select any cell within the pivot table.

2 Press the F11 key to create a graph in a new Chart sheet.

3 Alternatively, use the Alt + F1 shortcut to insert a chart on the active sheet.

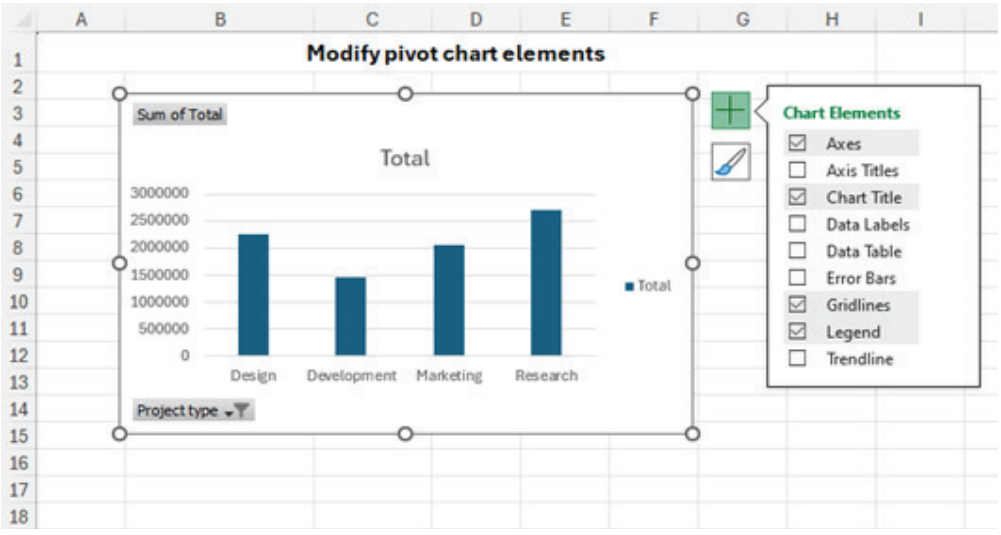
The newly created pivot chart will automatically use the default type. From there, you can easily modify it to suit your preferences by selecting a different chart type.

How to modify a pivot chart

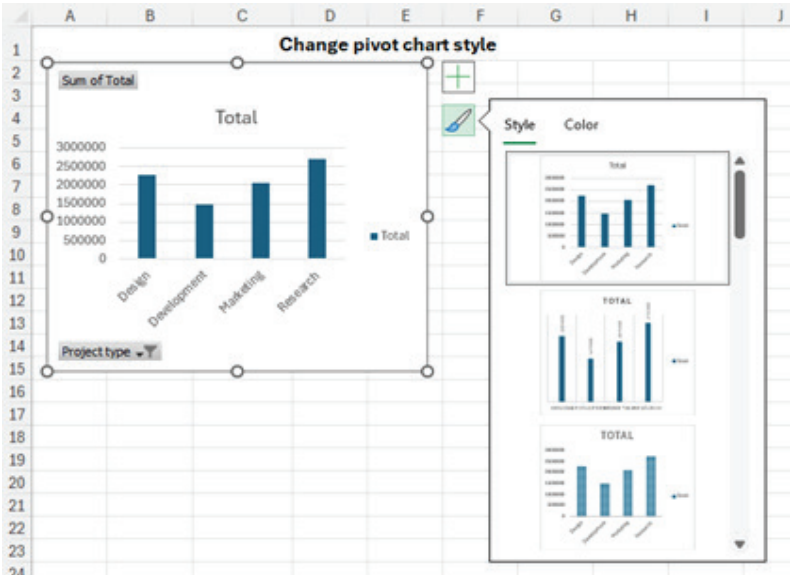
Pivot charts are not just powerful visualization tools; they're also incredibly flexible, allowing you to tweak nearly every aspect to suit your needs.

To show, hide or format graph elements like axes, labels, and legend, simply click the Chart Elements button (a plus sign) at the upper right.

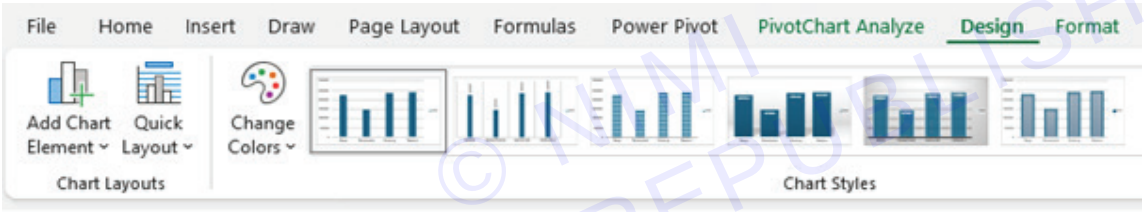
Corner of the graph:



For changing the chart style, click the Style icon at the upper right corner and choose various predefined styles:



For more options to adjust colours and styles, explore the Design and Format tabs on the ribbon. These tabs offer additional settings and tools to fine-tune your pivot chart to perfection.



How to change pivot chart type

You can change the type of your pivot graph at any time after you make it. Here are the steps:

1 Click anywhere within your graph to activate the pivot chats tabs on the ribbon.

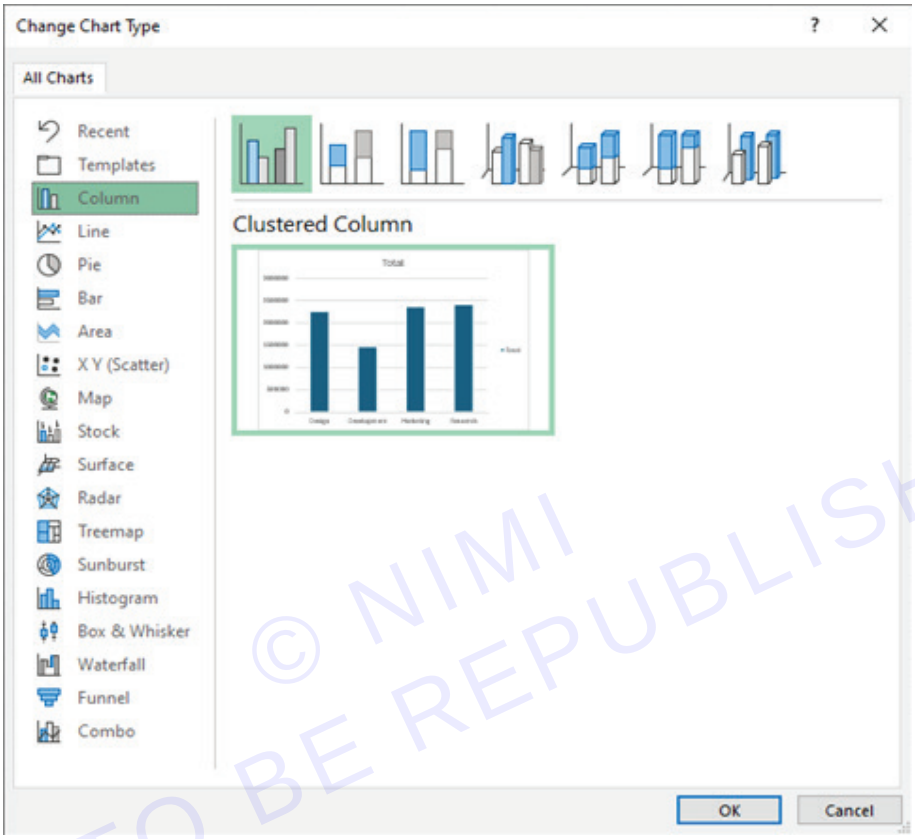
2 On the Design tab, in the Type group, click Change Chart Type.

3 Select the desired chart type from the options provided, and then click OK.

With these simple steps, you can quickly change the appearance of your pivot graph to better suit your data visualization goals.

Conditional Formatting

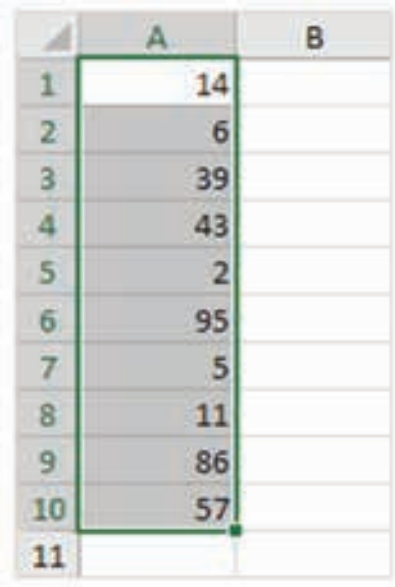
Use conditional formatting in Excel to automatically highlight cells based on their content. Simply apply a rule or use a formula to determine which cells to format.



Highlight Cells Rules

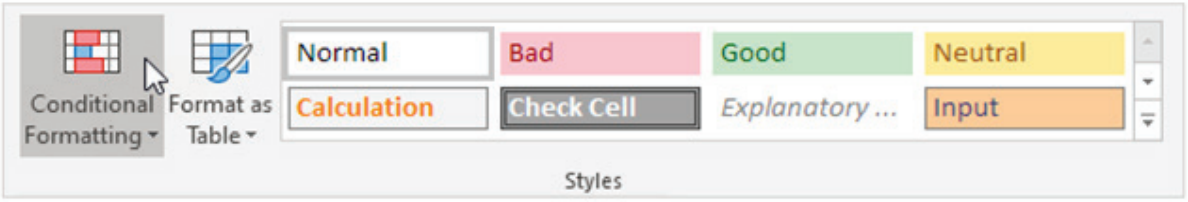
To highlight cells that are greater than a value, execute the following steps.

1 Select the range A1:A10.

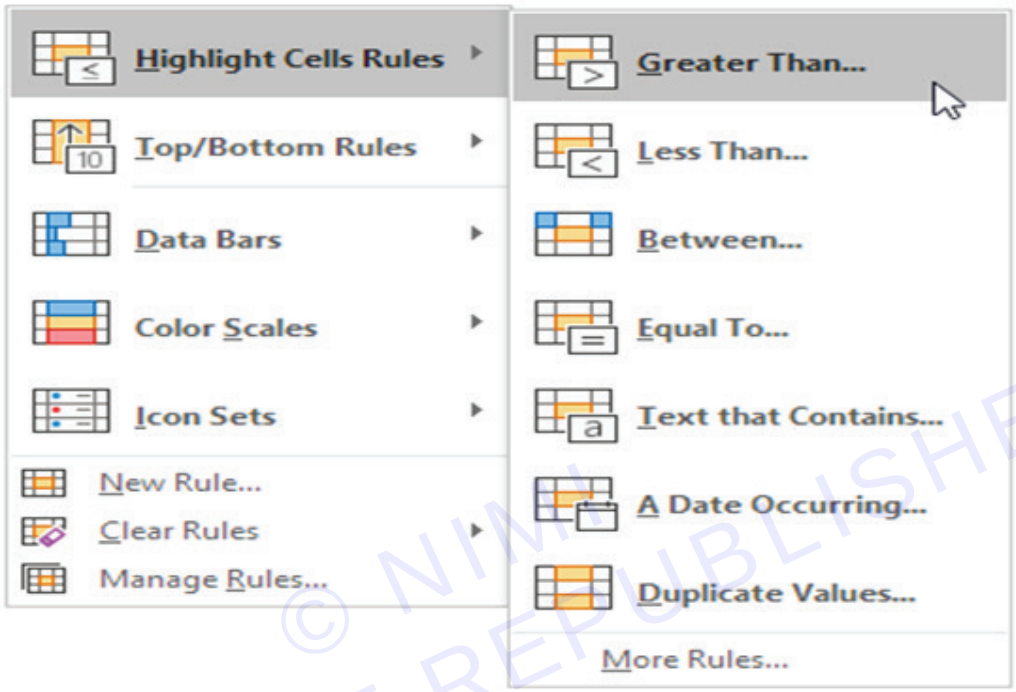


	A	B
1	14	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

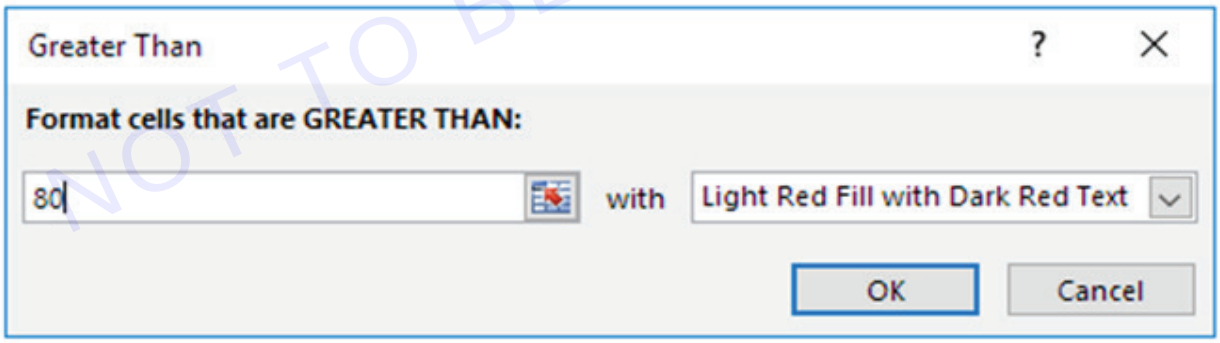
2 On the Home tab, in the Styles group, click Conditional Formatting.



3 Click Highlight Cells Rules, Greater Than.

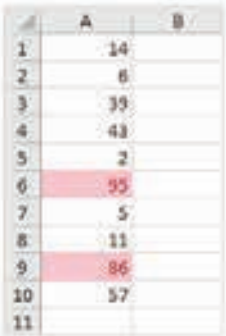


4 Enter the value 80 and select a formatting style.



5 Click OK.

Result. Excel highlights the cells that are greater than 80.



6 Change the value of cell A1 to 81.

Result. Excel changes the format of cell A1 automatically.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

Note: you can also use this category (see step 3) to highlight cells that are less than a value, between two values, equal to a value, cells that contain specific text, dates (today, last week, next month, etc.), duplicates or unique values.

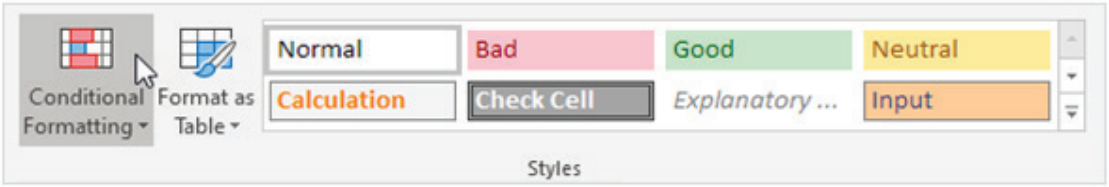
Clear Rules:

To clear a conditional formatting rule, execute the following steps.

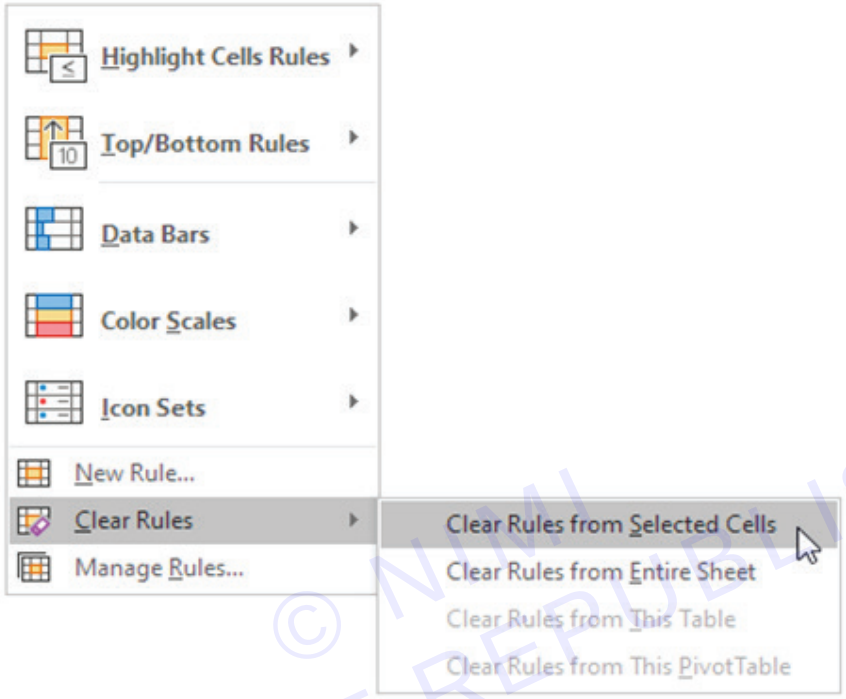
1 Select the range A1:A10.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

2 On the Home tab, in the Styles group, click Conditional Formatting.



3 Click Clear Rules, Clear Rules from Selected Cells.



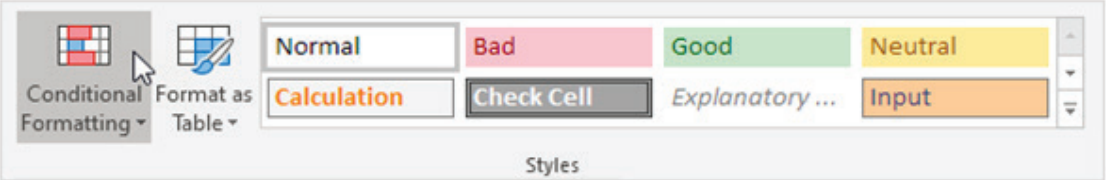
Top/Bottom Rules

To highlight cells that are above average, execute the following steps.

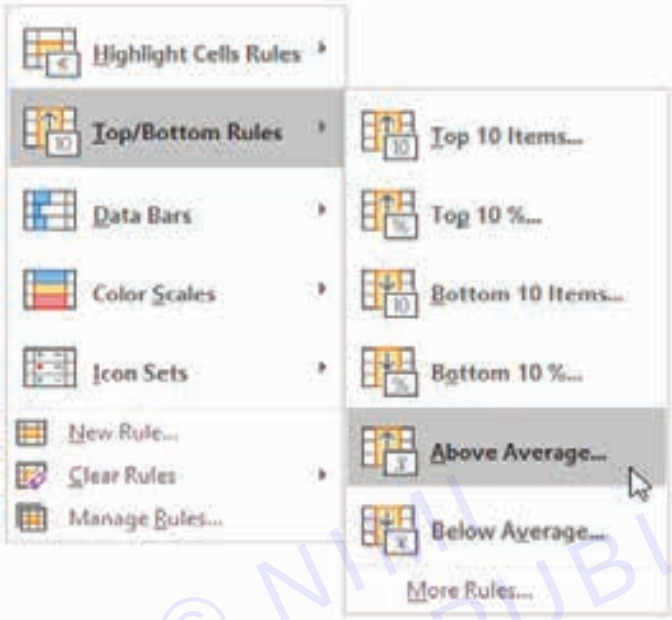
1 Select the range A1:A10.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

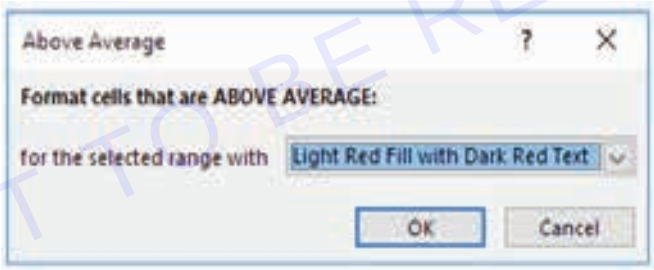
2 On the Home tab, in the Styles group, click Conditional Formatting.



3 Click Top/Bottom Rules, Above Average.



4 Select a formatting style.



5 Click OK.

Result. Excel calculates the average (42.5) and formats the cells that are above this average.

	A	B
1	81	
2	6	
3	39	
4	43	
5	2	
6	95	
7	5	
8	11	
9	86	
10	57	
11		

Note: you can also use this category (see step 3) to highlight the top n items, the top n percent, the bottom n items, the bottom n percent or cells that are below average.

ADVANCE CHART & GRAPH

The excel charts and graphs are the tools used to visualise the data by representing their values. Now, let's discuss the advanced charts and its usage in excel.

What is an advanced chart in Excel ?

Advanced charts are the charts that are beyond the basic charts in excel. If the user has more than one set of data and if the user wants to compare the data values on the same chart, then the advanced charts come in handy. The user can create the basic chart for one set of data and then they can add more datasets to that chart. The user can also format the charts, etc.

The importance of advanced charts in Excel

1 Advanced charts provide more consolidated information in a single chart. It paves way for the user to compare more than one data set and it helps alot to draw decisions.

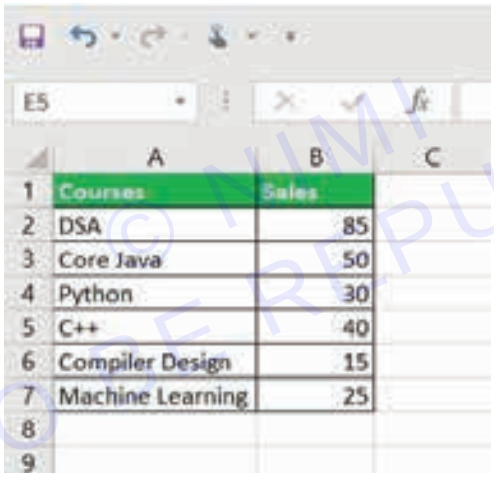
2 Advanced charts allow the user to customize the way it appears.

How to Create an Advance Chart in Excel:

In this example, we will create a column chart and format its various chart elements. We will use random sales data for different courses for this.

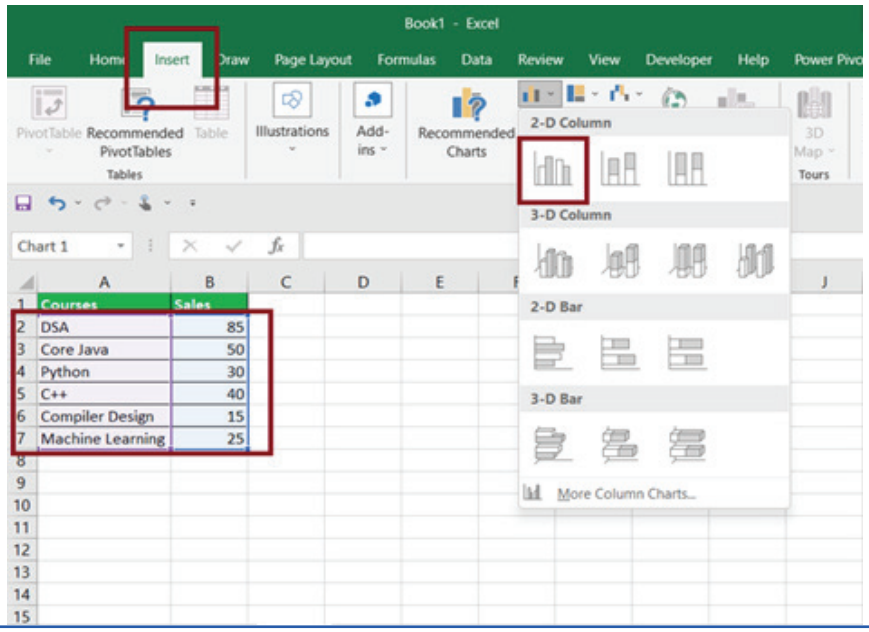
Step 1: Create a Dataset in a workbook

In this step, we will be creating the dataset using the following random sales data for different courses.



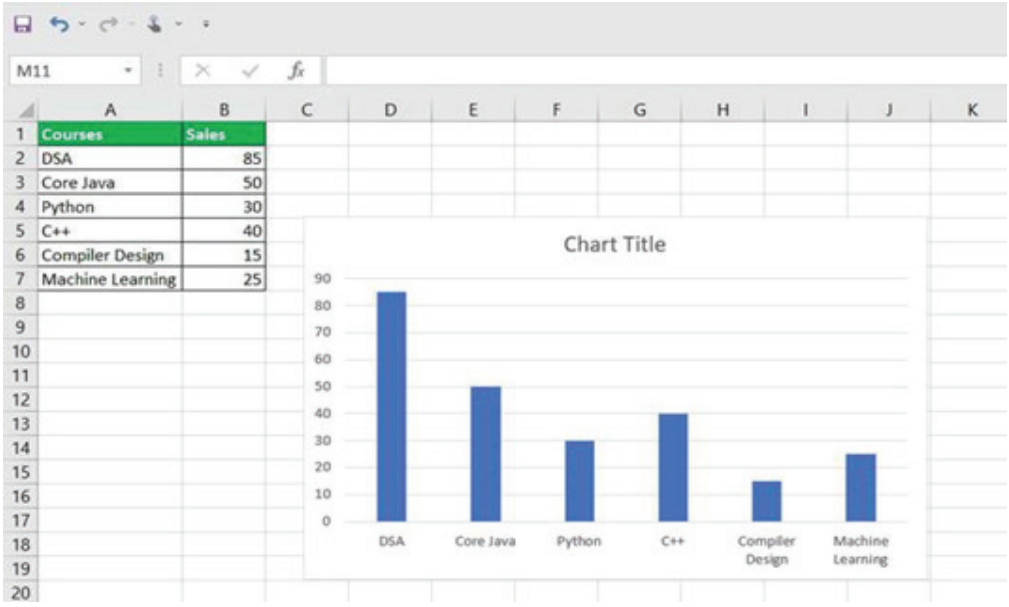
	A	B	C
1	Courses	Sales	
2	DSA	85	
3	Core Java	50	
4	Python	30	
5	C++	40	
6	Compiler Design	15	
7	Machine Learning	25	
8			
9			

Step 2: Insert Chart



Now, we will be inserting the chart for our dataset. To do this, we need to Select Data and then navigate to Insert. Then click on Charts and then select Column Chart there.

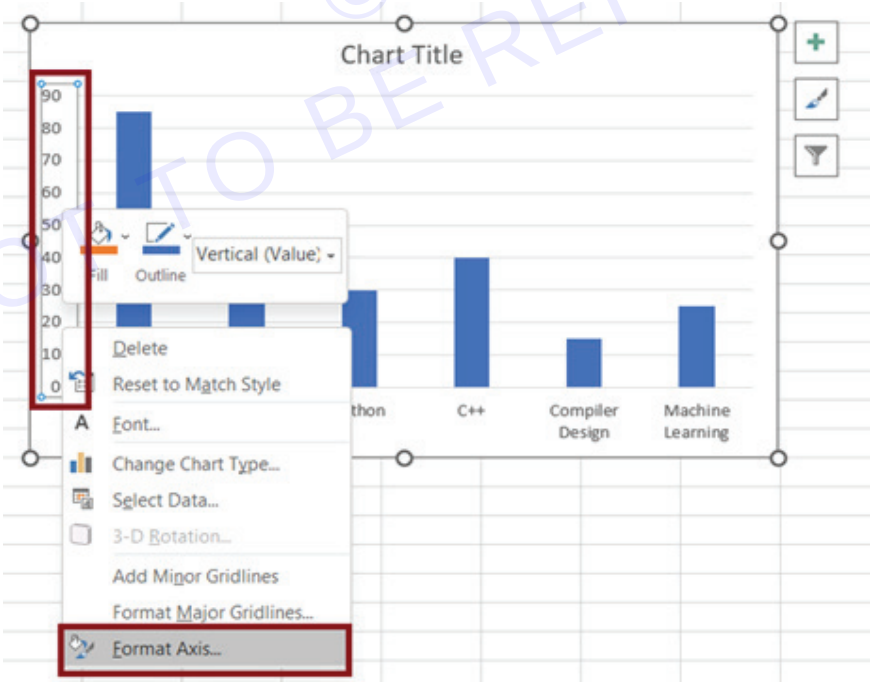
By doing this, Excel will automatically insert a column chart for our dataset.



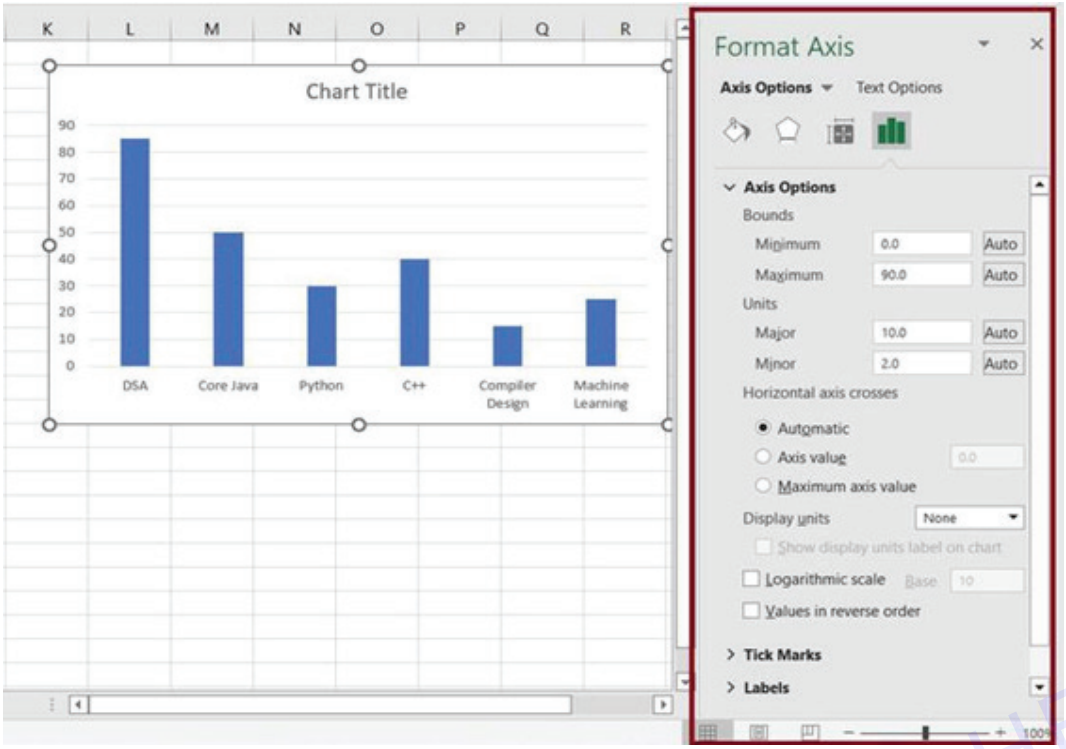
The Format Pane is introduced in Excel 2013 which provides a chart formatting option for various chart elements. To access the formatting pane, we need to Select Chart Element > Right-Click > Format<chart_element> Excel will open a Format Pane for that particular chart element on the right side of the spreadsheet.

Step 3: Format Axis

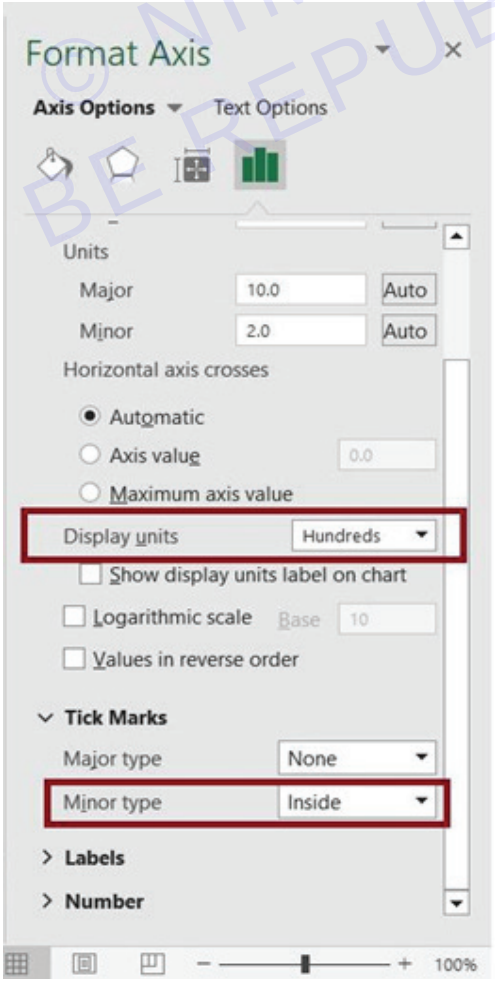
In this step, we will format the chart axis. First, Select Axis, then Right-Click on it, and then select Format Axis.



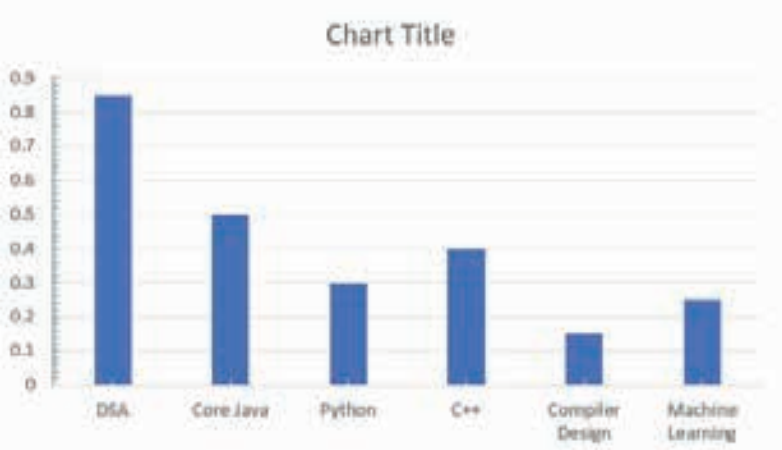
Once we click on the Format Axis option, Excel will automatically open a Format Axis Pane.



Using the format axis pane, we can format our chart axis according to our requirements. Here, we are going to change units to display in Hundreds and Tick Mark as Inside for Minor Type.

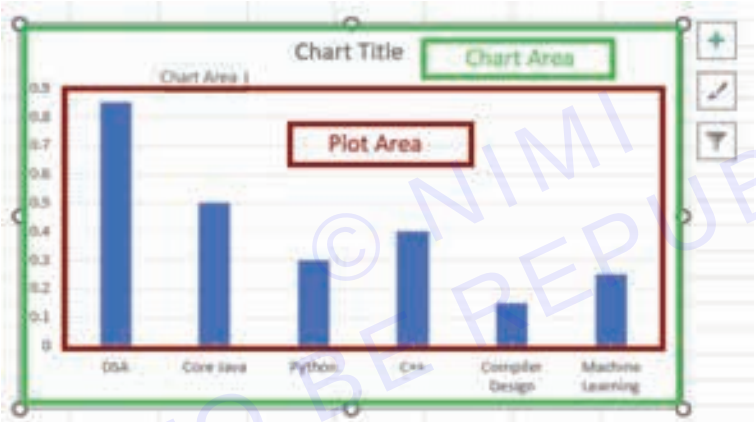


Similarly, we will also change the Tick Marks for Minor Type to Inside for Horizontal-Axis and we will get the following output.

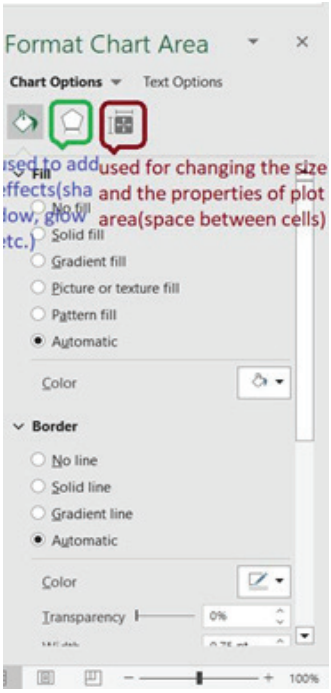


Step 4: Format Chart Area

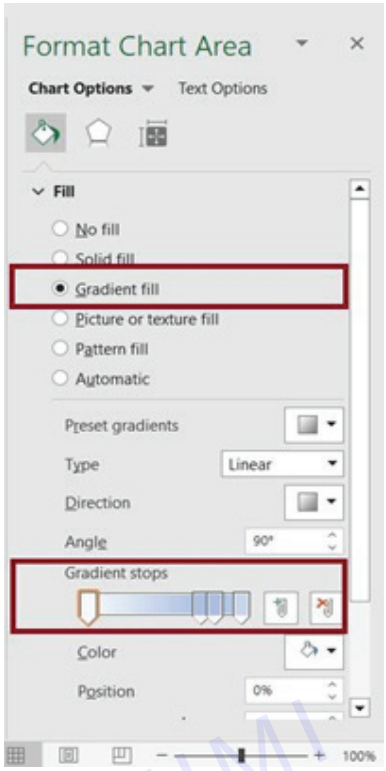
In this step, we will format the Chart Area. To make the chart look more enhanced and cleaner. For this Select Chart Area and then Right Click on it and select Format Chart Area. Please, make sure to click on Chart Area, not the Plot Area.



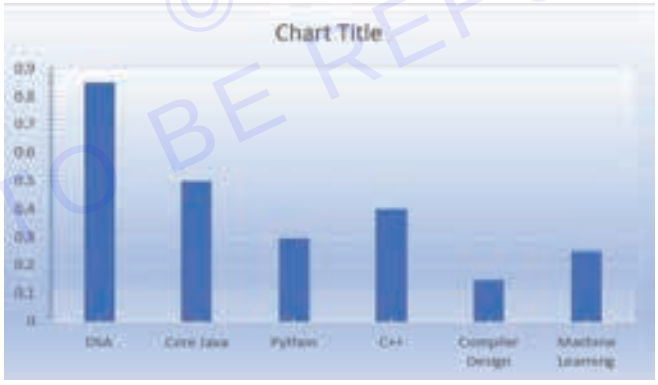
Once we select the chart area and perform the above operation, excel will open the Format Chart Area pane.



To format the chart area pane, we are adding gradient colour to our chart. Go to Format Chart Area, click on Fill & Line, and select Gradient Fill.

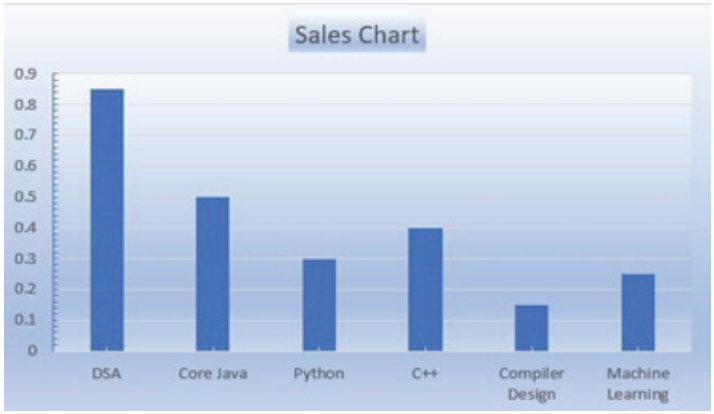


Once we choose the Gradient fill option, excel will automatically fill the gradient colour to our Chart Area. Similarly, we will also add the Gradient Fill to our Plot Area, and we will get the following output.



Step 5: Format Chart Title

In this step, we will format the Chart Title.

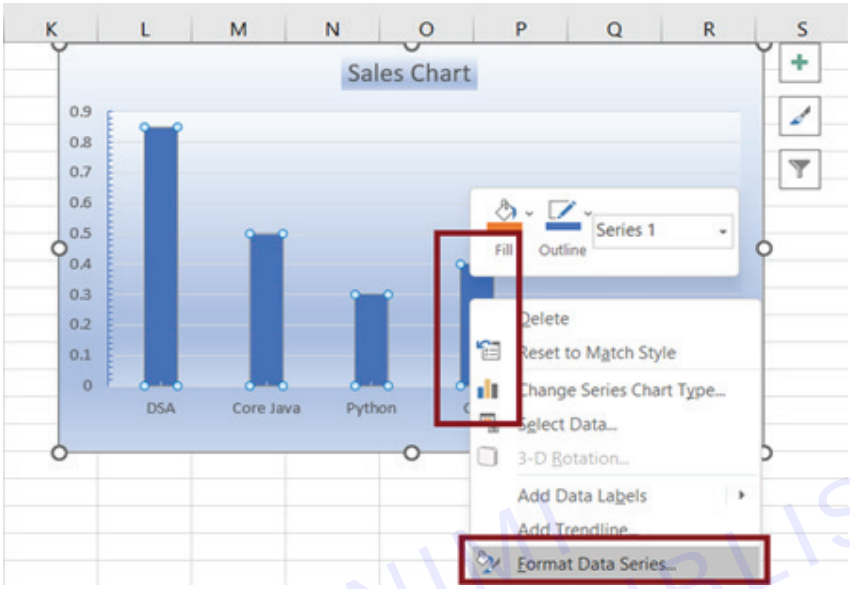


To make the chart look more enhanced and cleaner. For this Select Chart Title and then Right Click on it and select Format Chart Title. This will open the Format Chart Title pane on the right side of the spreadsheet.

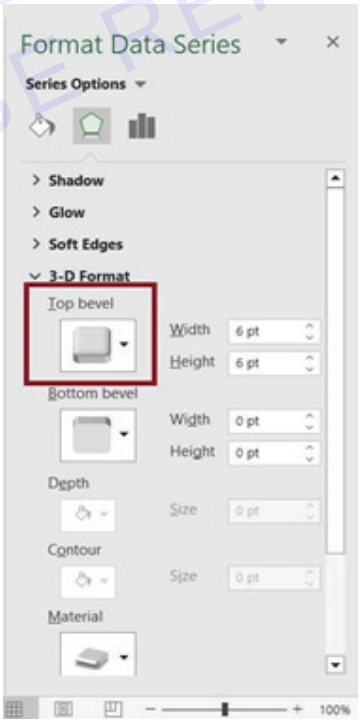
Using Format Chart Title, we are choosing the Gradient Fill option to fill gradient colour to our chart title. Also, we will change the title to Sales Chart. After adding these changes, we will g**et the following output.**

Step 6: Format Data Series

In this step, we will format the Chart Data Series. To make the chart look more enhanced and cleaner. For this Select Chart Data Series and then Right Click on it and select Format Data Series.



This will open the Format Data Series pane on the right side of the Excel spreadsheet. Using the Format Data Series, we will add a 3-D effect to our data series.



Once we add the 3-D effect to our data series we will get the following final output.

FAQs on Chart Formatting?

What is the need of formatting a chart?

Formatting the chart makes the chart easier to read and also qualifies you to explain the data in detail.

What are the best-advanced Graphs in Excel?

- Sankey diagram
- Likert Scale chart
- Comparison Bar Chart
- Gauge Chart
- Multi-Axis Line Chart
- Sunburst Chart
- Radar Chart
- Radial Bar Chart
- Box and Whisker Chart
- Dot Plot Chart

What is Power Query?

Power Query is a business intelligence tool available in Excel that allows you to import data from many different sources and then clean, transform and reshape your data as needed.

It allows you to set up a query once and then reuse it with a simple refresh. It's also pretty powerful. Power Query can import and clean millions of rows into the data model for analysis after. The user interface is intuitive and well laid out so it's really easy to pick up. It's an incredibly short learning curve when compared to other Excel tools like formulas or VBA.

The best part about it, is you don't need to learn or use any code to do any of it. The power query editor records all your transformations step by step and converts them into the M code for you, similar to how the Macro recorder with VBA.

If you want to edit or write your own M code, you certainly can, but you definitely don't need to.

What Can Power Query Do?

Imagine you get a sales report in a text file from your system

on a monthly basis that looks like this.

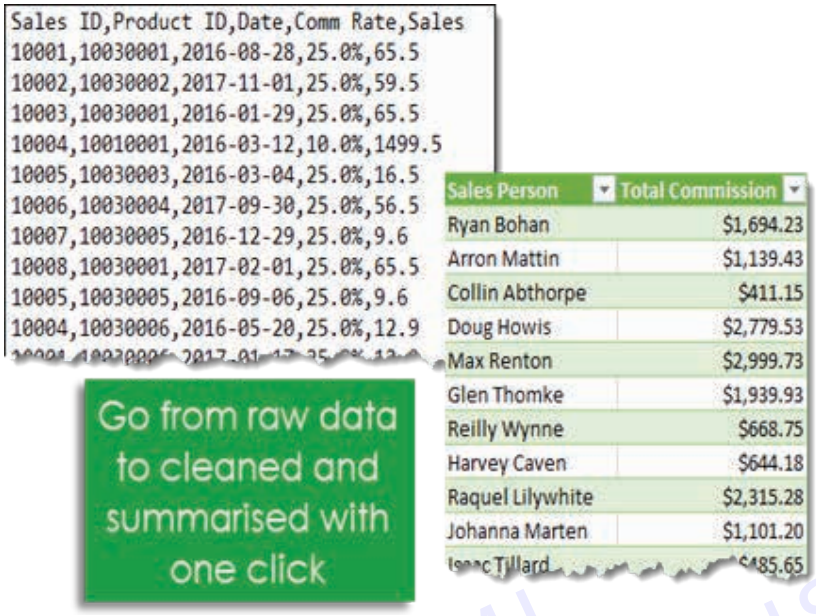
Every month you need to go to the folder where the file is uploaded and open the file and copy the contents into Excel. You then use the text to column feature to split out the data into new columns.

The system only outputs the sales person's ID, so you need to add a new column to the data and use a VLOOKUP to get the salesperson associated with each ID. Then you need to summarize the sales by salesperson and calculate the commission to pay out.

```
Sales ID,Product ID,Date,Comm Rate,Sales
10001,10030001,2016-08-28,25.0%,65.5
10002,10030002,2017-11-01,25.0%,59.5
10003,10030001,2016-01-29,25.0%,65.5
10004,10010001,2016-03-12,10.0%,1499.5
10005,10030003,2016-03-04,25.0%,16.5
10006,10030004,2017-09-30,25.0%,56.5
10007,10030005,2016-12-29,25.0%,9.6
10008,10030001,2017-02-01,25.0%,65.5
10005,10030005,2016-09-06,25.0%,9.6
10004,10030006,2016-05-20,25.0%,12.9
```

You also need to link the product ID to the product category but only the first 4 digits of the product code relate to the product category. You create another column using the LEFT function to get the first 4 digits of the product code, then use a VLOOKUP on this to get the product category. Now you can summarize the data by category.

Maybe it only takes an hour a month to do, but it's pretty mindless work that's not enjoyable and takes away from time you can actually spend analyzing the data and producing meaningful insight.

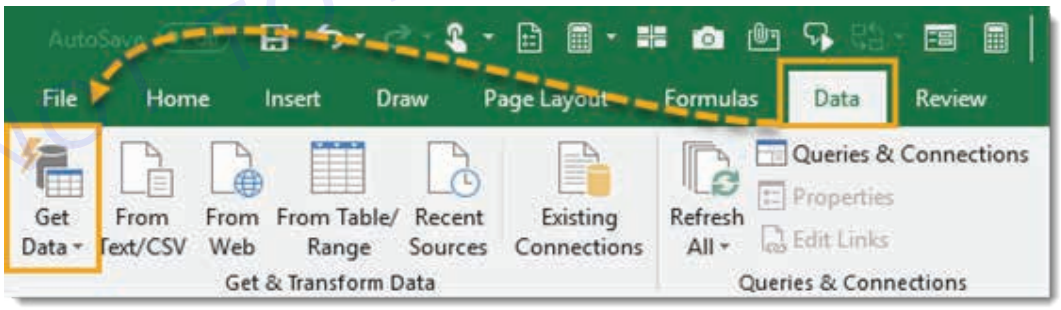


With Power Query, this can all be automated down to a click of the refresh button on a monthly basis. All you need to do is build the query once and reuse it, saving an hour of work each and every month!

Where is Power Query?

Power Query is available as an add-in to download and install for Excel 2010 and 2013 and will appear as a new tab in the ribbon labelled Power Query. In 2016 it was renamed to Get & Transform and appears in the Data tab without the need to install any add-in.

Importing Your Data with Power Query



Importing your data with Power Query is simple. Excel provides many common data connections that are accessible from the Data tab and can be found from the Get Data command.

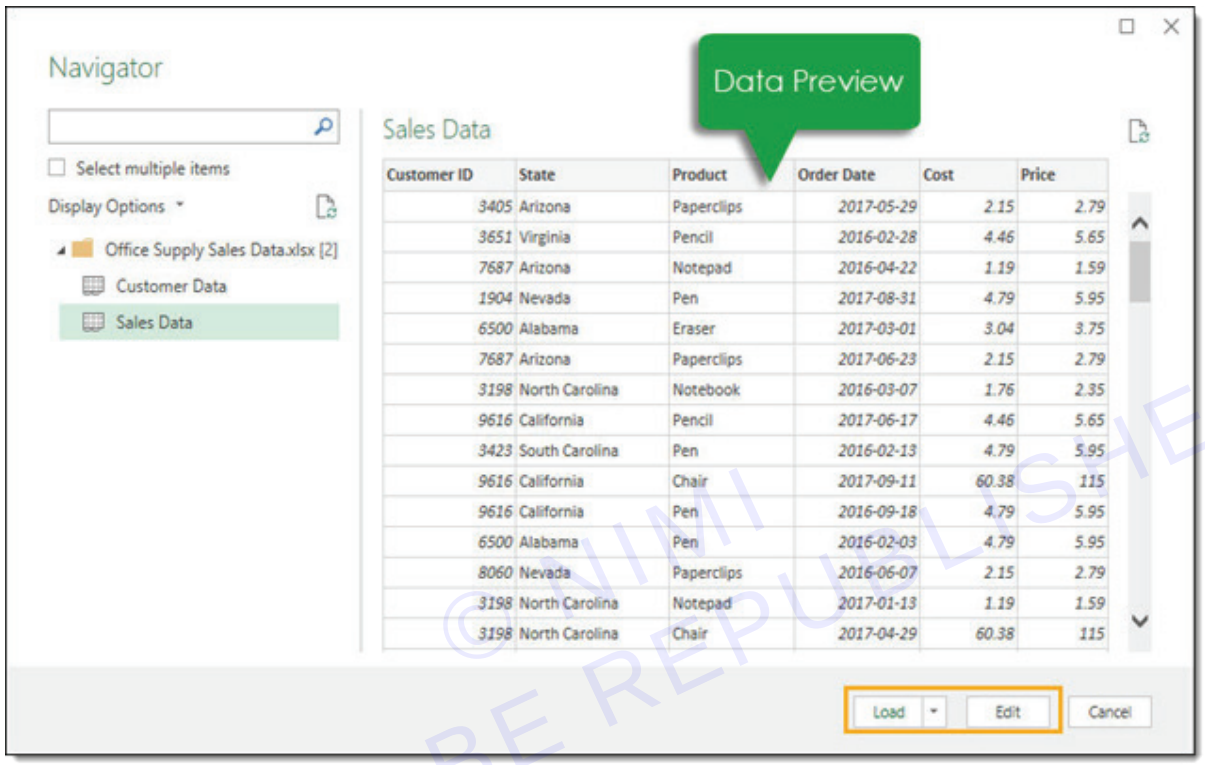
- Get data from a single file such as an Excel workbook, Text or CSV file, XML and JSON files. You can also import multiple files from within a given folder.
- Get data from various databases such as SQL Server, Microsoft Access, Analysis Services, SQL Server Analysis Server, Oracle, IBM DB2, MySQL, PostgreSQL, Sybase, Teradata and SAP HANA databases.
- Get data from Microsoft Azure
- Get data from online services like Sharepoint, Microsoft Exchange, Dynamics 365, Facebook and Salesforce.
- Get data from other sources like a table or range inside the current workbook, from the web, a Microsoft Query, Hadoop, OData feed, ODBC and OLEDB.
- We can merge two queries together similar to joining two queries in SQL.

- We can append a query to another query similar to a union of two queries in SQL.

Note: The available data connection options will depend on your version of Excel.

There are a couple of the more common query types available in the top level of the ribbon commands found in the Get & Transform section of the Data tab. From here we can easily access the From Text/CSV, From Web and From Table/Range queries. These are just duplicated outside of the Get Data command for convenience of use, since you'll likely be using these more frequently.

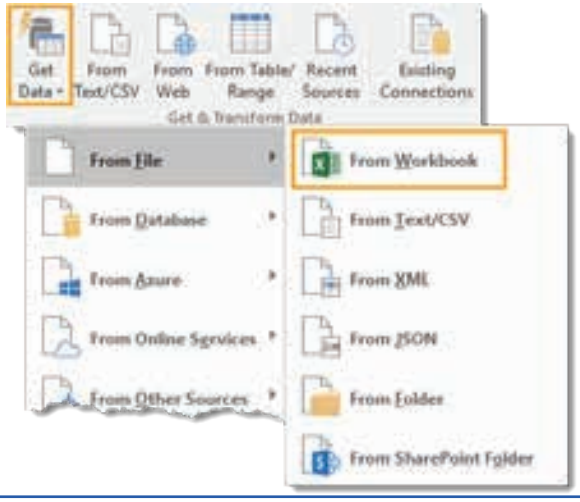
Depending on which type of data connection you choose, Excel will guide you through the connection set up and there might be several options to select during the process.



At the end of the setup process, you will come to the data preview window. You can view a preview of the data here to make sure it's what you're expecting. You can then load the data as is by pressing the Load button, or you can proceed to the query editor to apply any data transformation steps by pressing the Edit button.

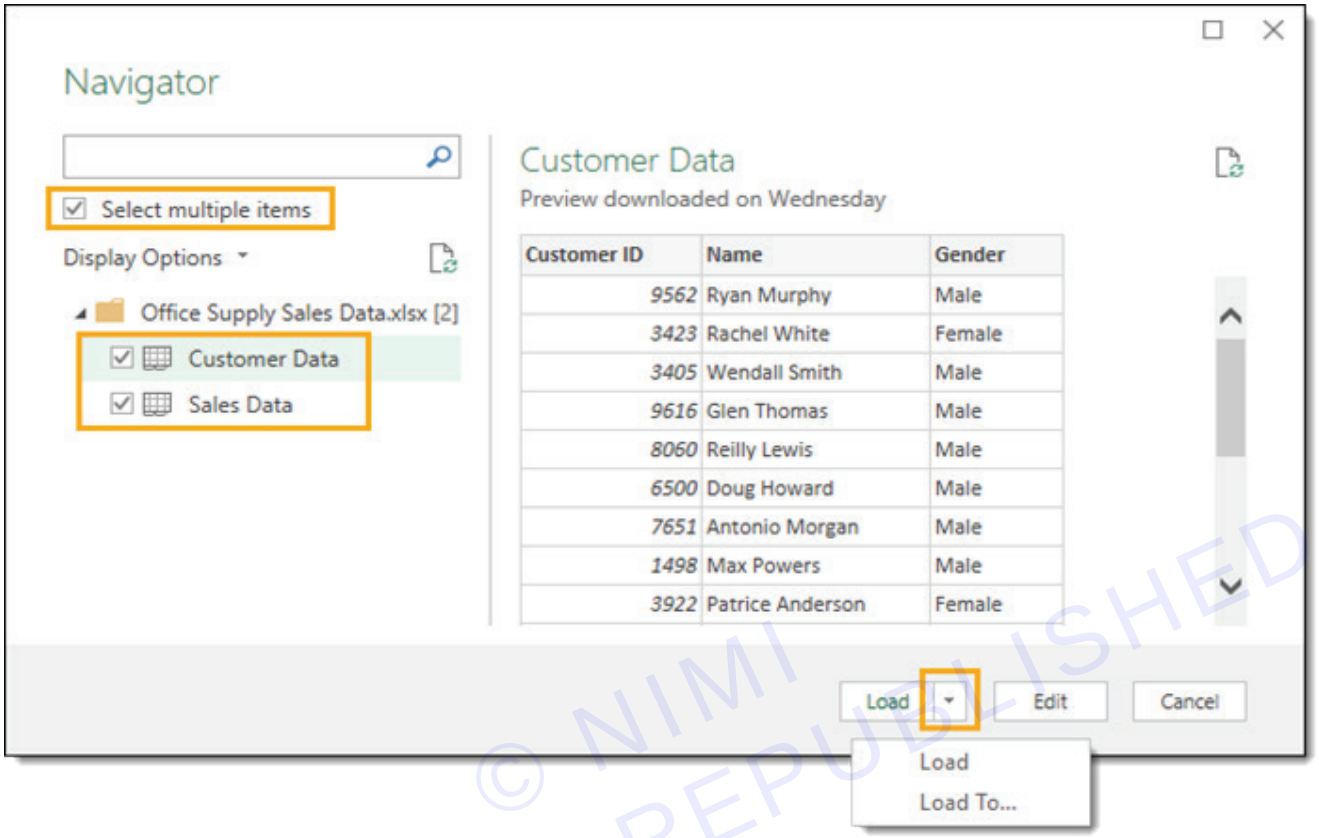
A Simple Example of Importing Data in an Excel File

Let's take a look at importing some data from an Excel workbook in action. We're going to import an Excel file called Office Supply Sales Data.xlsx. It contains sales data on one sheet called Sales Data and customer data on another sheet called Customer Data. Both sheets of data start in cell A1 and the first row of the data contains column headers.



Go to the Data tab and select the Get Data command in the Get & Transform Data section. Then go to From File and choose From Workbook.

This will open a file picker menu where you can navigate to the file you want to import. Select the file and press the Import button.

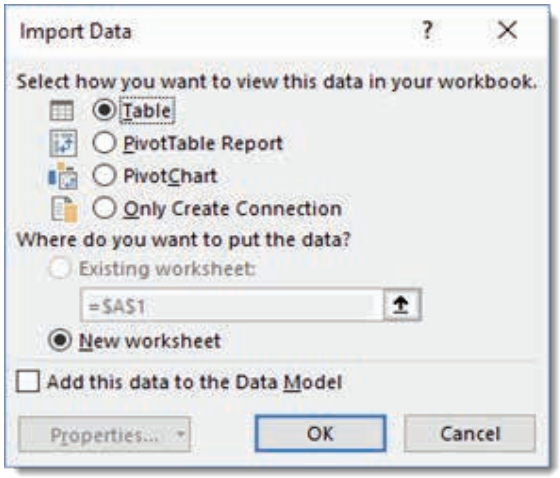


After selecting the file you want to import, the data preview Navigator window will open. This will give you a list of all the objects available to import from the workbook. Check the box to Select multiple items since we will be importing data from two different sheets. Now we can check both the Customer Data and Sales Data.

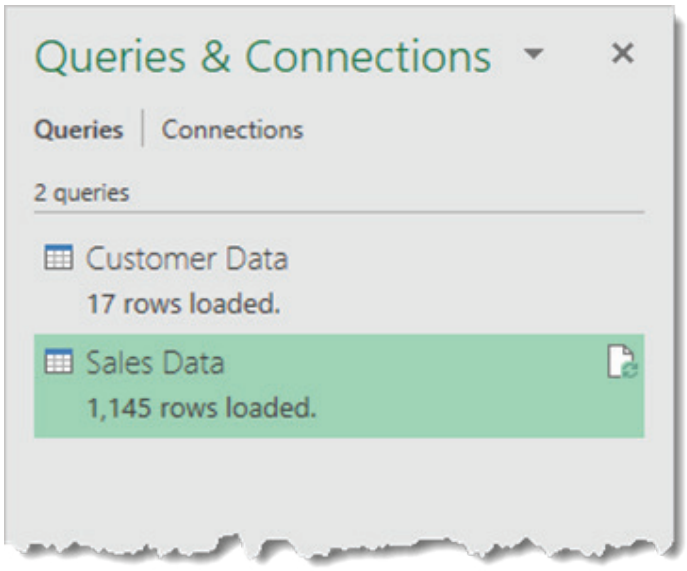
When you click on either of the objects in the workbook, you can see a preview of the data for it on the right hand side of the navigator window. This is great for a sense check to make sure you've got the correct file.

When you're satisfied that you've got everything you need from the workbook, you can either press the Edit or Load buttons. The edit button will take you to the query editor where you can transform your data before loading it. Pressing the load button will load the data into tables in new sheets in the workbook.

In this simple example, we will bypass the editor and go straight to loading the data into Excel. Press the small arrow next to the Load button to access the Load To options. This will give you a few more loading options.

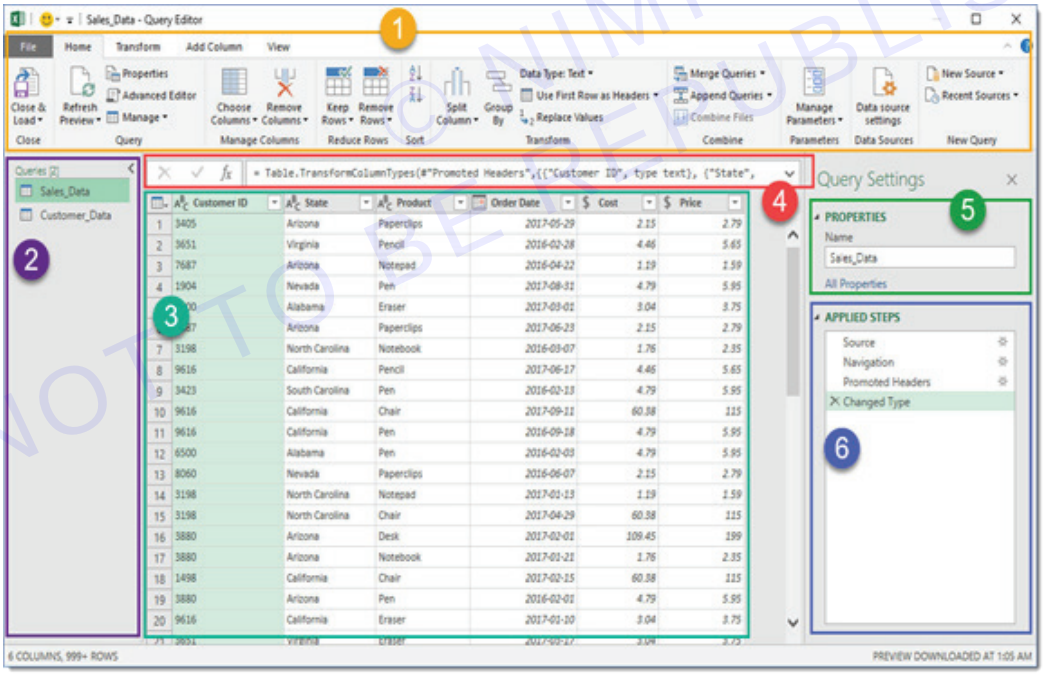


We will choose to load the data into a table in a new sheet, but there are several other options. You can also load the data directly into a pivot table or pivot chart, or you can avoid loading the data and just create a connection to the data.



Now the tables are loaded into new sheets in Excel and we also have two queries which can quickly be refreshed if the data in the original workbook is ever updated.

The Query Editor



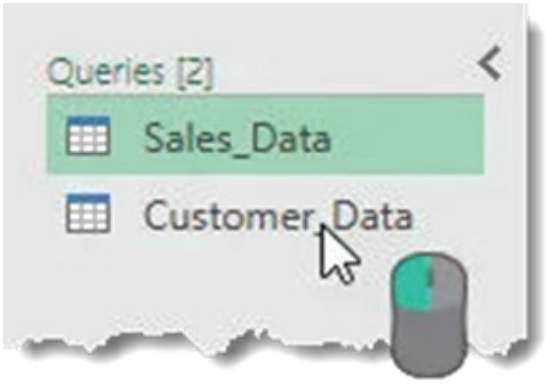
After going through the guide to connecting your data and selecting the Edit option, you will be presented with the query editor. This is where any data transformation steps will be created or edited. There are 6 main area in the editor to become familiar with.

1 **The Ribbon** – The user interface for the editor is quite similar to Excel and uses a visual ribbon style command center. It organizes data transformation commands and other power query options into 5 main tabs.

2 **Query List** – This area lists all the queries in the current workbook. You can navigate to any query from this area to begin editing it.

3 **Data Preview** – This area is where you will see a preview of the data with all the transformation steps currently applied. You can also access a lot of the transformation commands here either from the filter icons in the column headings or with a right click on the column heading.

The Query List:

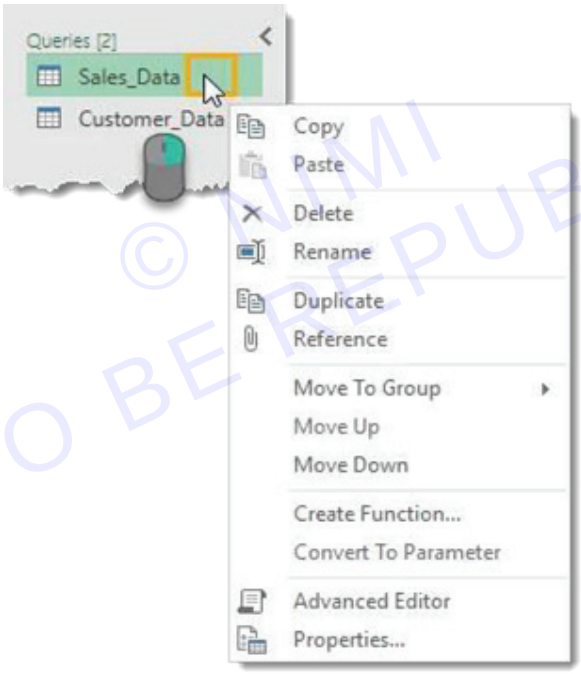


1 **Formula Bar** – This is where you can see and edit the M code of the current transformation step. Each transformation you make on your data is recorded and appears as a step in the applied steps area.

2 **Properties** – This is where you can name your query. When you close and load the query to an Excel table, power query will create a table with the same name as its source query if the table name isn't already taken. The query name is also how the M code will reference this query if we need to query it in another query.

3 **Applied Steps** – This area is a chronological list of all the transformation steps that have been applied to the data.

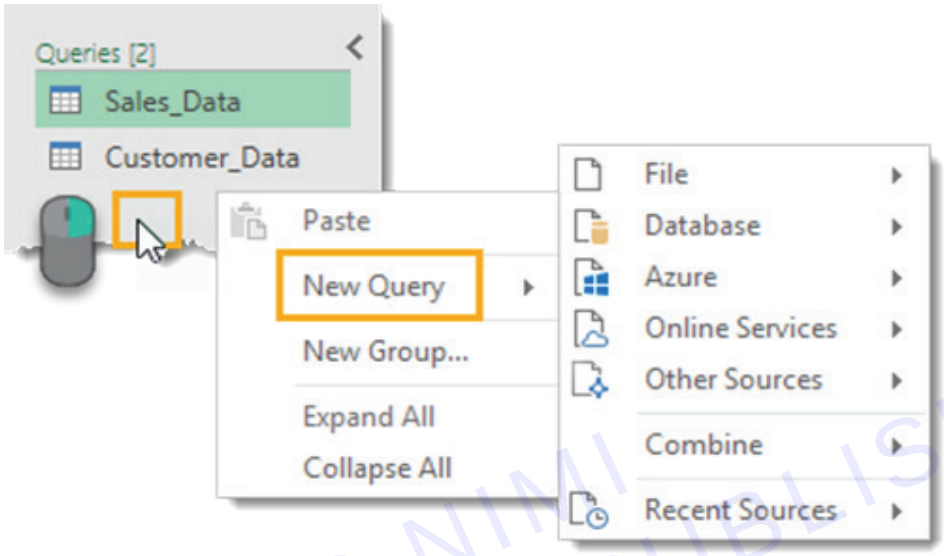
One of the primary functions of the query list is navigation. There's no need to exit the query editor to switch which query you're working on. You can left click on any query to switch. The query you're currently on will be highlighted in a light green colour.



When you do eventually exit the editor with the close and load button, changes in all the queries You can hide the query list to create more room for the data preview. Left click on the small arrow in the upper right corner to toggle the list between hidden and visible. If you right click on any query in the list, there are a variety of options available.

- **Copy and Paste** – Copy and paste a query to make another copy of it.
- **Delete** – Delete the query. If you accidentally delete a query, there's no undo button, but you can exit the query editor without saving via close and load to restore your query.
- **Rename** – Rename your query. This is the same as renaming it from the properties section on the left hand side of the editor.
- **Duplicate** – Make another copy of the query. This is the same as copy and paste but turns the process into one step.
- **Move To Group** – Place your queries into a folder like structure to keep them organised when the list gets large.

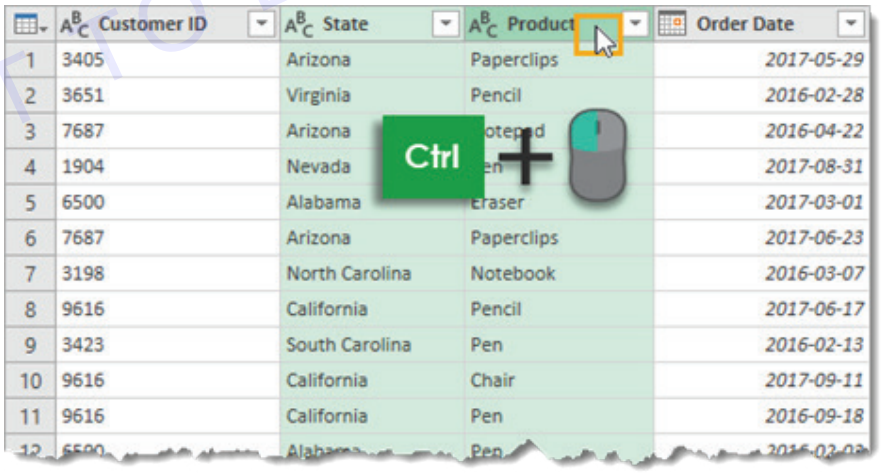
- **Move Up and Move Down** – Rearrange the order your queries appear in the list or within the folder groups to add to your organisational efforts. This can also be done by dragging and dropping the query to a new location.
- **Create Function** – Turn your query into a query function. They allow you to pass a parameter to the query and return results based on the parameter passed.
- **Convert To Parameter** – Allows you to convert parameters to queries or queries to parameters.
- **Advanced Editor** – Open the advanced editor to edit the M code for the query.
- **Properties** – Allows you to change the query name, add a description text and enable Fast Data Load option for the query.



If you right click any empty area in the query list, you can create a new query.

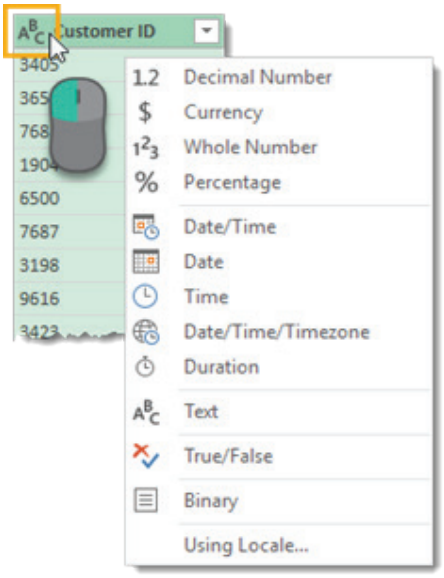
The Data Preview

The main job of the data preview area is to apply transformation steps to your data and show a preview of these steps you're applying



In the data preview area, you can select columns with a few different methods. A column will be highlighted in a light green colour when it's selected.

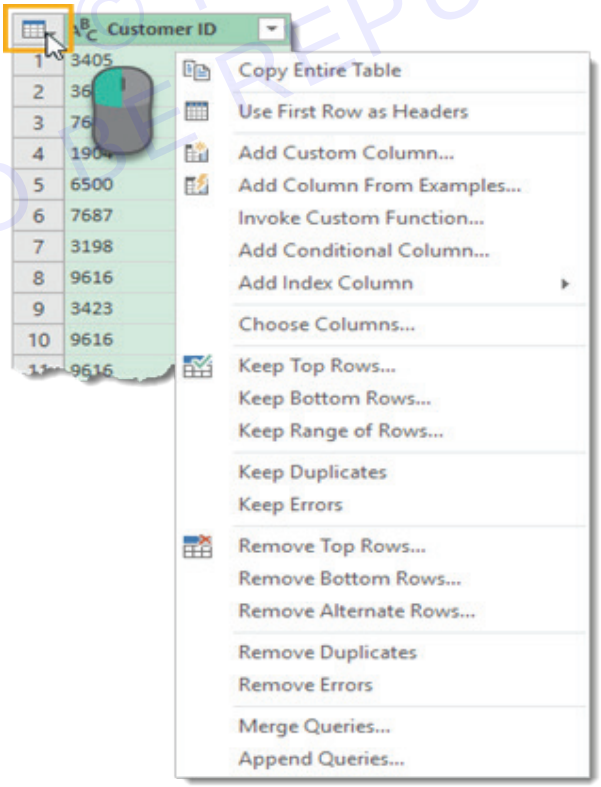
- Select a single column with a left click on the column heading.
- Select multiple adjacent columns with a left click on the first column heading, then hold Shift and left click on the last column heading.
- Select multiple non-adjacent columns by holding Ctrl then left click on any column headings you want to select.



You can then apply any relevant data transformation steps on selected columns from the ribbon or certain steps can be accessed with a right click on the column heading. Commands that are not available to your selected column or Each column has a data type icon on the left hand of the column heading. You can left click on it to change the data type of the column.

You can choose from decimal numbers, currency, whole numbers, percentages, date and time, dates, times, time zone, duration, text, Boolean, and binary.

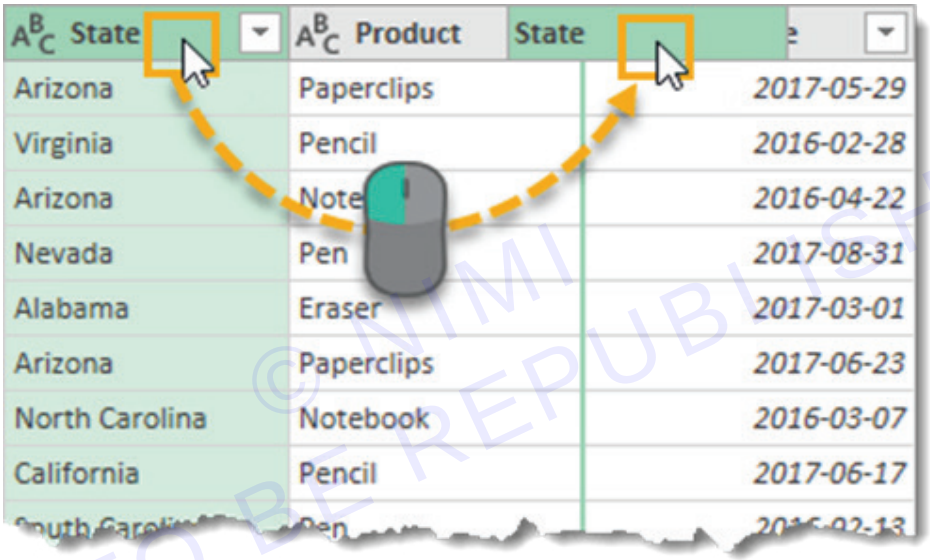
Using the Locale option allows you to set the data type format using the convention from different locations. For example, if you wanted to display the date in the American m/d/yyyy format instead of the usual dd/mm/yyyy then you could select United States as the locale.



There's a small table icon in the top left hand corner of the data preview, you can right click or left click this to access various actions that affect the whole table.

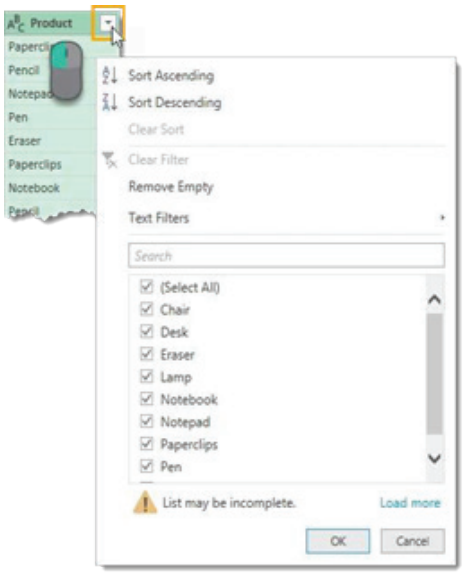
Renaming any column heading is really easy.

Double left click on any column heading then type your new name and press Enter when you're done.

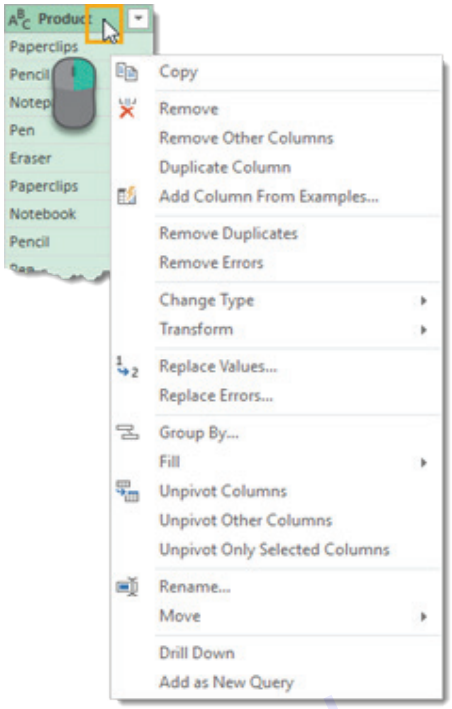


You can change around the order of any of the columns with a left click and drag action. The green border between two columns will become the new location of the dragged column when you release the left click.

Each column also has a filter toggle on right hand side. Left click on this to sort and filter your data. This filter menu is very similar to the filters found in a regular spreadsheet and will work the same way.



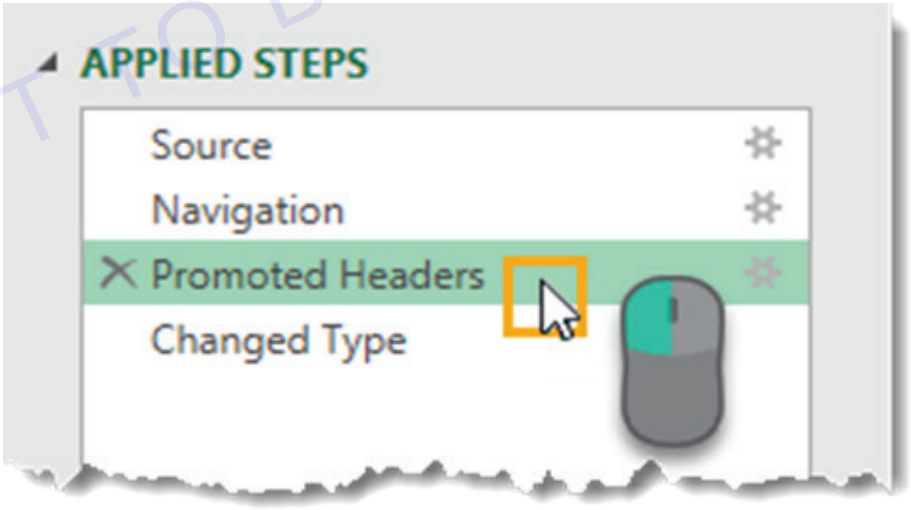
The list of items shown is based on a sample of the data so may not contain all available items in the data. You can load more by clicking on the Load more text in blue.



Many transformations found in the ribbon menu are also accessible from the data preview area using a right click on the column heading. Some of the action you select from this right click menu will replace the current column. If you want to create a new column based, use a command from the Add Column tab instead.

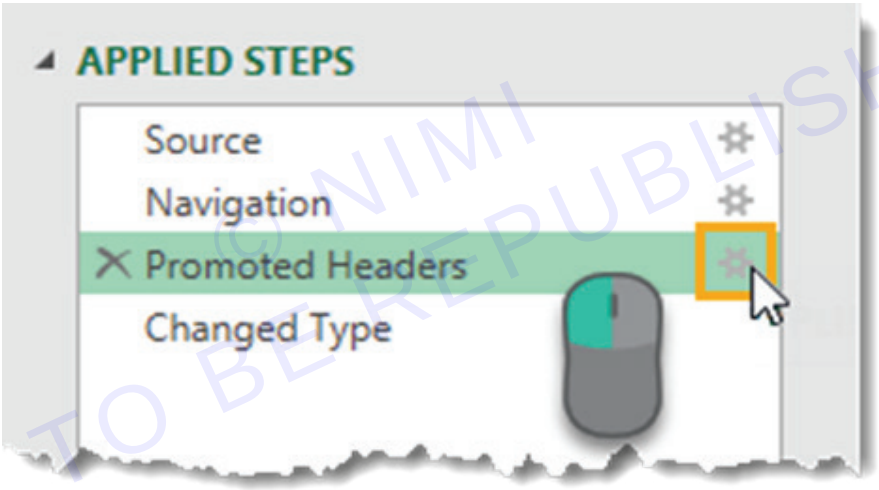
The Applied Steps

Any transformation you make to your data will appear as a step in the Applied Steps area. It also allows you to navigate through your query. Left click on any step and the data preview will update to show all transformations up to and including that step.

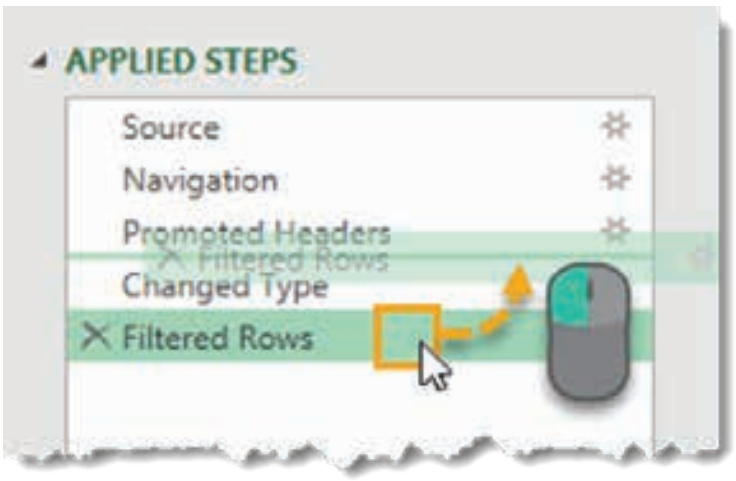


You can insert new steps into the query at any point by selecting the previous step and then creating the transformation in the data preview. Power Query will then ask if you want to insert this new step. Careful though, as this may break the following steps that refer to something you changed.

You can delete any steps that were applied using the X on the left hand side of the step name in the Applied Steps area. Caution is needed though, as if any of the following steps depend on the step you're trying delete, you will break your query. This is where Delete Until End from the right click menu can be handy.

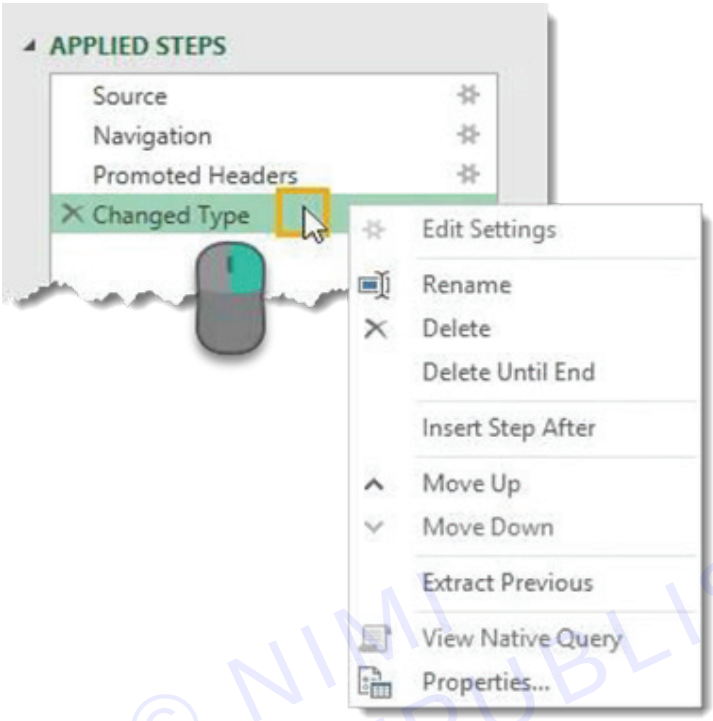


A lot of transformation steps available in power query will have various user input parameters and other setting associated with them. If you apply a filter on the product column to show all items not starting with Pen, you might later decide you need to change this filter step to show all items not equal to Pen. You can make these edits from the Applied Step area.



Some of the steps will have a small gear icon on the right hand side. This allows you to edit the inputs and settings of that step.

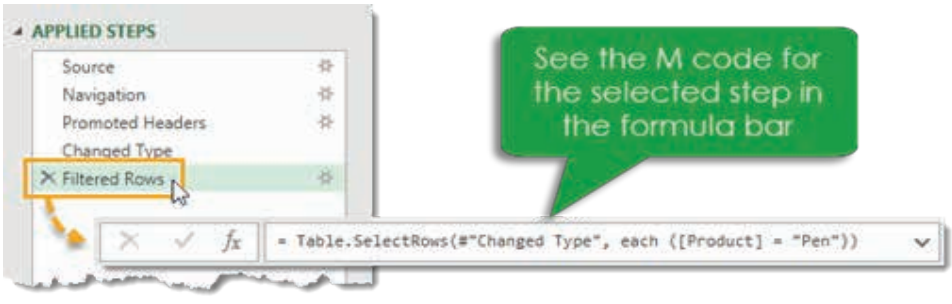
You can rearrange the order the steps are performed in your query. Just left click on any step and drag it to a new location. A green line between steps will indicate the new location. This is another one you'll need to be careful with as a lot of steps will depend on previous steps, and changing ordering can create errors because of this.



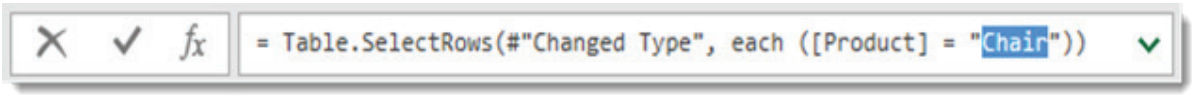
Right click on any step to access a menu of options

- **Edit Settings** – This allows you to edit the settings of the step similar to using the gear icon on the right hand side of the step.
- **Rename** – This allows you to rename the steps label. Instead of the displaying the generic name like "Filtered Rows", you could have this display something like "Filtered Product Rows on Pens" so you can easily identify what the step is doing.
- **Delete** – This deletes the current step similar to the X on the left hand side of the step.
- **Delete Until End** – This allows you to delete the current step plus all steps up until the end. Since steps can depend on previous steps, deleting all steps after a step is a good way to avoid any errors.
- **Insert Step After** – This allows you to insert a new step after the current step.
- **Move Up and Move Down** – This allows you to rearrange the query steps similar to the dragging and dropping method.
- **Extract Previous** – This can be a really useful option. It allows you to create a new copy of the query up to the selected step.

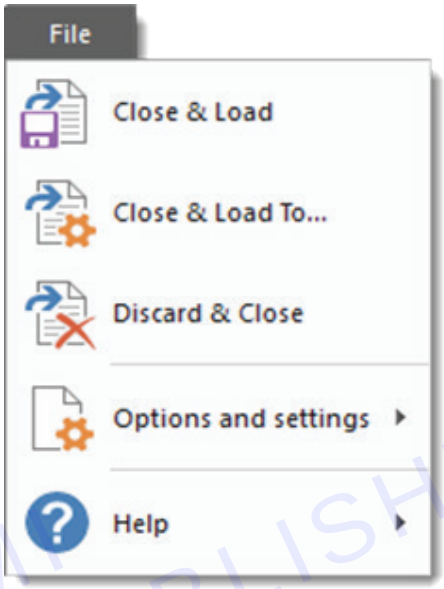
The Formula Bar



When you click on different steps of the transformation process in the Applied Steps area, the formula bar updates to show the M code that was created for that step. If the M code generated is longer than the formula bar, you can expand the formula bar using the arrow toggle on the right hand side.



You can edit the M code for a step directly from the formula bar without the need to open the advanced editor. In this example, we've changed our filter from "Pen" to "Chair" by typing in the formula bar and then pressing Enter or using the check mark on the left to confirm the change. Press Esc or use the X on the left to discard any changes.



The File Tab

The File tab contains various options for saving any changes made to your queries as well as power query options and settings.

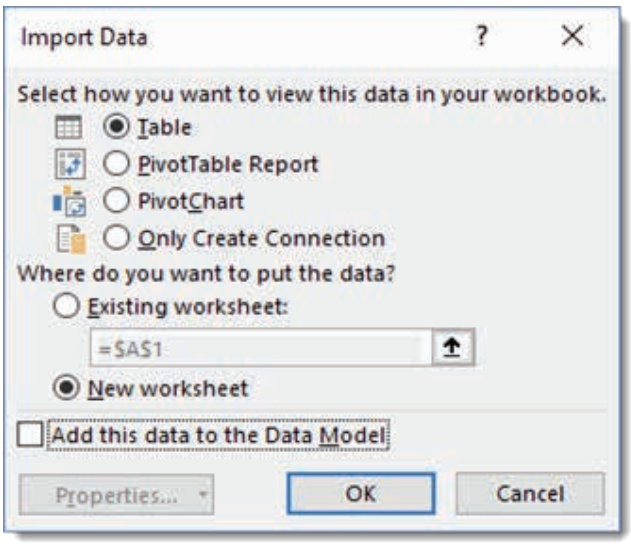
- **Close & Load** – This will save your queries and load your current query into an Excel table in the workbook.
- **Close & Load To** – This will open the Import Data menu with various data loading options to choose from.
- **Discard & Close** – This will discard any changes you made to the queries during your session in the editor and close the editor.

Data Loading Options

When you use the Close & Load To option to exit the editor, this will open the Import Data menu.

You can choose to load the query to a table, pivot table, pivot chart or only create a connection for the query. The connection only option will mean there is no data output to the workbook, but you can still use this query in other queries. This is a good option if the query is an intermediate step in a data transformation process.

You'll also be able to select the location to load to in your workbook if you selected either a table, pivot table or pivot chart in the previous section. You can choose a cell in an existing worksheet or load it to a new sheet that Excel will create for you automatically.



The other option you get is the Add this data to the Data Model. This will allow you to use the data output in Power Pivot and use other Data Model functionality like building relationships between tables. The Data Model Excel's new efficient way of storing and using large amounts of data.

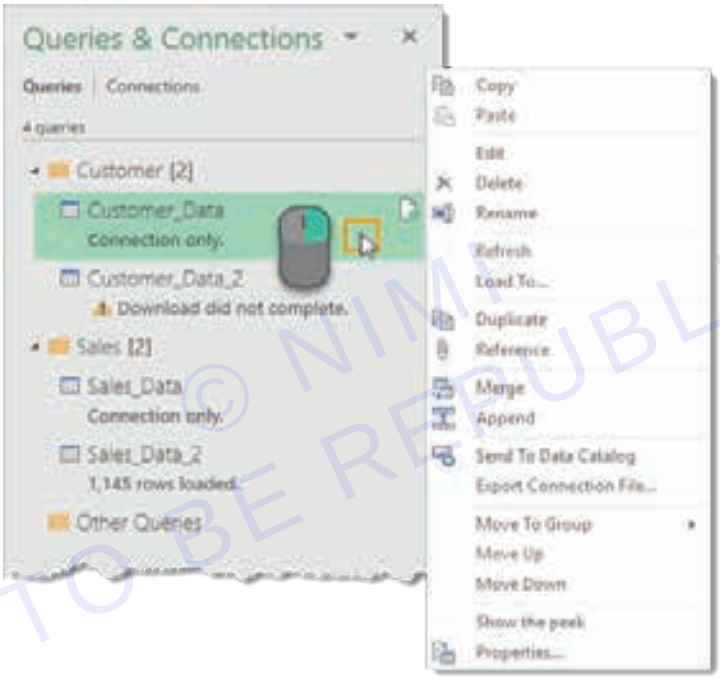
The Queries & Connections Window

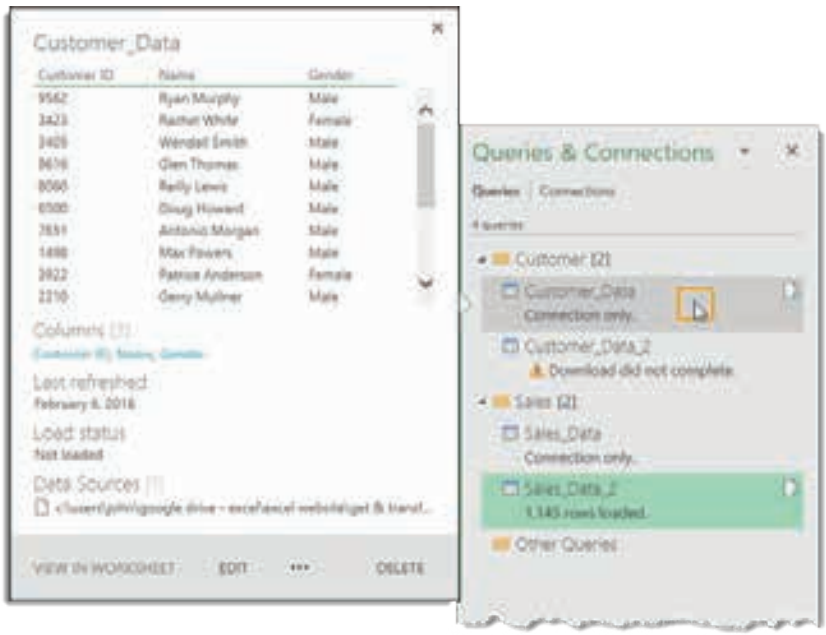
When you're working outside of the power query editor, you can see and interact with all the queries in the workbook through the Queries & Connections window. To open this, go to the Data tab in the regular Excel ribbon, then press the Queries & Connections command button found in the Queries & Connections section.

When opened it will be docked to the right hand side of the workbook. You can undock it by left clicking on the title and dragging it. You can drag it to the left hand side and dock it there or leave it floating. You can also resize the window by left clicking and dragging the edges.

This is very similar to the query list in the editor and you can perform a lot of the same actions with a right click on any query.

One option worth noting that's not in the query list right click menu, is the Load To option. This will allow you to change the loading option for any query, so you can change any Connection only queries to load to an Excel table in the workbook.



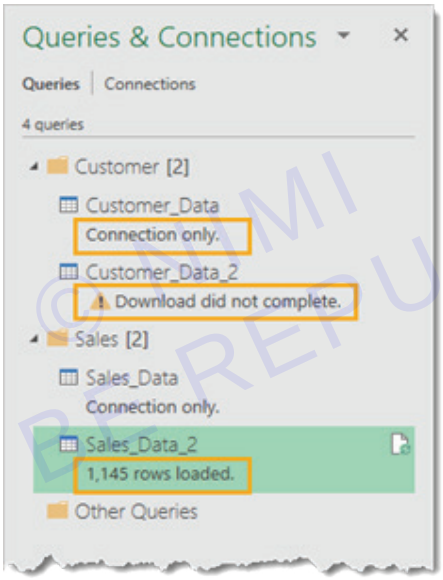


Another thing worth noting is when you hover over a query with the mouse cursor, Excel will generate a Peek Data Preview. This will show you some basic information about the query.

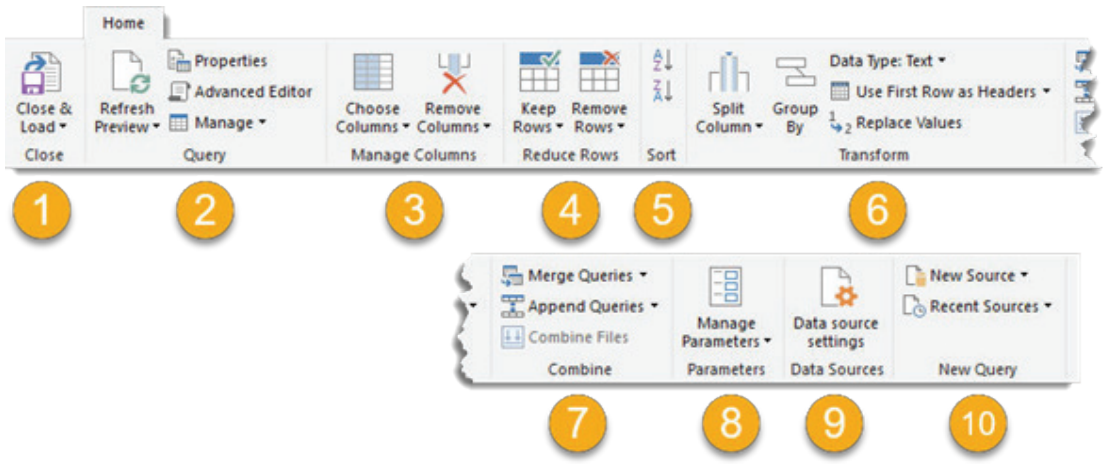
- **Data Preview** – This is a live preview of the data similar to when first setting up a query.
- **Columns** – This will give you a list of all the columns contained in the final results of the query along with a count of how many columns there are. Clicking on any of them will highlight the column in the data preview.
- **Last Refreshed** – This will tell you when the data was last refreshed.
- **Load Status** – This displays whether the data is loaded to a table, pivot table, pivot chart or is a connection only.
- **Data Sources** – This will show you the source of the data along with a count of the number of files if you're it's a from folder query.
- **View in Worksheet** – Clicking on this will take you to the output table if the query is loaded to a table, pivot table or pivot chart.

You can also access this Peek view by right clicking on the query and selecting Show the peek.

There are also some useful messages displayed in the Queries & Connections window for each query. It will show you if the query is a connection only, if there were any errors when the query last ran, or how many rows loaded.



The Home Tab



The Home tab contains all the actions, transformations, and settings that will affect the whole table.

1 **Close** – You can access the Close & Load and Close & Load To options from here. These are also available in the File tab menu.

2 **Query** – You can refresh the data preview for the current query or all query connections. You can also open the properties settings and the advanced editor for the current query and there are options under the Manage button to delete, duplicate or reference the current query.

3 **Manage Columns** – You can navigate to specific columns and choose to keep or remove columns.

4 **Reduce Rows** – You can manage the rows of data from this section. There are lots of options to either keep certain rows or remove certain rows. Keep or remove the top N rows, the bottom N rows, a particular range of rows, alternating rows, duplicate rows or rows with errors. One option only available for removing rows is to remove blank rows.

5 **Sort** – You can sort any column in either ascending or descending order.

6 **Transform** – This section contains a mix of useful transformation options.

- Split Columns – This allows you to split the data in a column based on a delimiter or character length.
- Group By – This allows you to group and summarize your data similar to a Group By in SQL.
- Data Type – This allows you to change the data type of any column.
- Use First Row as Headers – This allows you to promote the first row of data to column headings or demote the column headings to a row of data.
- Replace Values – This allows you to find and replace any value from a column.

1 Combine – This sections contains all the commands for joining your query to with other queries. You can merge, append queries or combine files when working with a from folder query.

2 Parameters – Power Query allows you to create parameters for your queries. For example when setting up a from folder query, you may want the folder path to be a parameter as so you can easily change the location. You can create and manage existing parameters from this section.

3 Data Sources – This section contains the data source settings including permissions management for any data sources that require passwords to access.

4 New Query – You can create new queries from new data sources or previously used data sources from this section.

The Difference Between the Transform and Add Column Tabs

The bulk of all transformations available in power query can be accessed through either the Transform tab or the Add Column tab.

You might think there is a lot of duplication between these two tabs. For example, both tabs contain a From Text section with a lot of the same commands. It's not really the case, there is a subtle difference!

When you use a command from the Add Column tab that is found in both tabs, it will create a new column with the transformed data and the original column will stay intact. Whereas using the equivalent command from the Transform tab will change the original column and no new column is created.

The Transform Tab

The Transform tab sections.

1 Table – This section contains commands that will transform the entire table. You can group and aggregate your query, promote rows to headers, demote headers to rows transpose your data, reverse row order, and count rows.

2 Any Column – This section contains commands that will work on any column of data regardless of data type. You can change the data type, automatically detect and change the data type, rename the column heading, find and replace values, fill values down (or up) a column to replace any blanks or nulls with the value above it (or below it), pivot or unpivot columns, move columns to a new location or convert a column to a list.

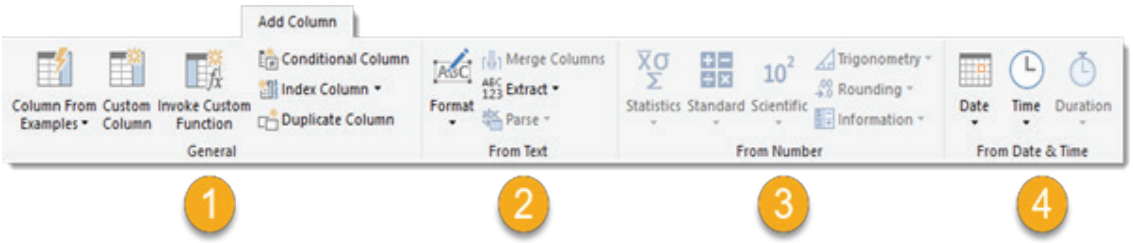
3 Text Column – This section contains commands for text data. You can split columns with a delimiter, format the case, trim and clean, merge two or more columns together, extract text, and parse XML or JSON objects.

4 Number Column – This section contains commands for numerical data. You can perform various aggregations like sums and averages, perform standard algebra operations or trigonometry and round numbers up or down.

5 Date & Time Column – This section contains commands for date and time data. You can extract information from your dates, times and duration data.

6 Structured Column – This section contains commands for working with nested data structures such as when your column contains tables.

The Add Column Tab



The Add Column tab contains a lot of commands similar to the Transform tab, but the key difference is they will create a new column with the transformation.

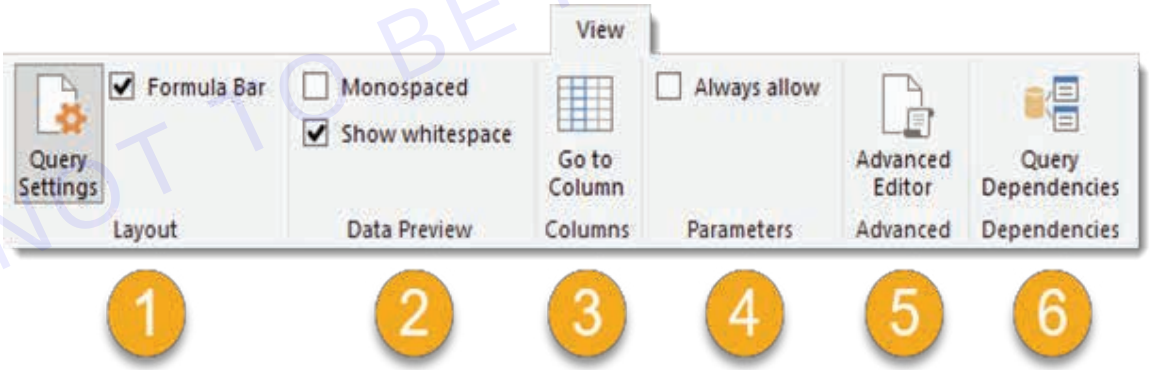
1 General – This section allows you to add new columns based on formulas or custom functions. You can also add index columns or duplicate a column from here.

2 From Text – Very similar to the From Text section in the Transform tab, but these commands will create a new column with the transformation.

3 From Number – Very similar to the From Number section in the Transform tab, but these commands will create a new column with the transformation.

4 From Date & Time – Very similar to the From Date & Time section in the Transform tab, but these commands will create a new column with the transformation.

The View Tab



The View tab is quite sparse in comparison to the other tabs. There are no transformation commands to be found in it. Most Power Query users will rarely need to use this area, but there are still a few things worth knowing about.

1. Layout – This section allows you to either show or hide the Query Setting pane (which contain the properties and applied steps) and the Formula Bar.
2. Data Preview – This section allows you to show or hide whitespace characters or turn the font into a monospace font in the data preview area. This is handy when dealing with data delimited by a certain number of characters.
3. Columns – This allows you to go to and select a certain column in the data preview. This command is also available in the Home tab.
4. Parameters – This allows you to enable parameterization in data sources and transformation steps.
5. Advanced – This will open the advanced query editor which shows the M code for the query. This is also available from the Home tab.
6. Dependencies – This will open a diagram view of the query dependencies in the workbook.

MODULE 6 : Object Oriented Programming and JAVA Language

LESSON 78 - 84 : Object oriented programming and JAVA language

Objectives

At the end of this lesson, you will be able to:

- implement various object - oriented concepts based program designs with JAVA
- identify JAVA language components and how they work together in applications
- design dynamic web pages using JAVA (AWT, Applet).

Object Oriented Programming With Core JAVA

Object-Oriented Programming is a paradigm that provides many concepts, such as inheritance, data binding, polymorphism, etc. The programming paradigm where everything is represented as an object is known as a truly object-oriented programming language.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Java code DRY “Don’t Repeat Yourself”, and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time.

What is OOPS in Java?

OOPs can be defined as:

A modular approach where data and functions can be combined into a single unit known as an object. It emphasises data and security and provides the reusability of code. Using OOP, we can resemble our code in the real world.

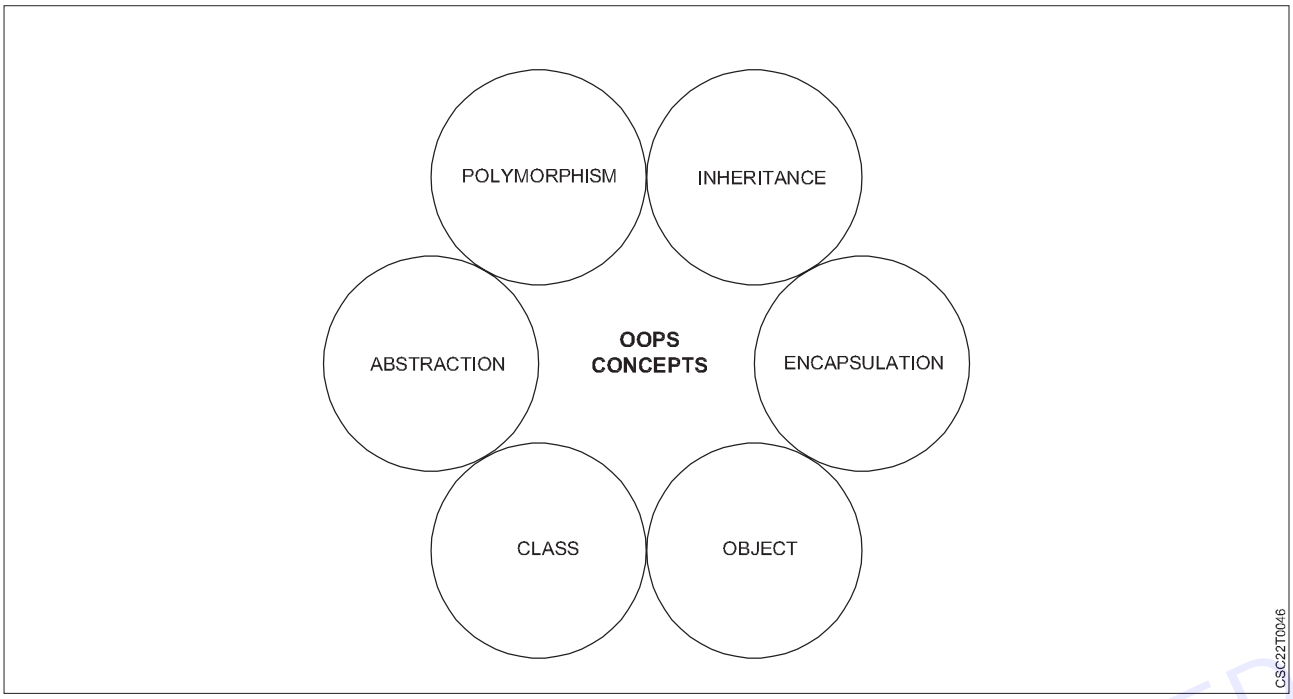
List of Java OOPs Concepts

The following are the concepts in OOPs :

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Apart from these concepts, there are some other terms which are used in Object-Oriented design:

- Coupling
- Cohesion
- Association
- Aggregation
- Composition



Object

Any entity that has state and behaviour is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Class

A class can be considered as a blueprint using which you can create as many objects as you like. The class is a group of similar entities. It is only a logical component and not a physical entity. **For example**, if you had a class called "Expensive Cars" it could have objects like Mercedes, BMW, Toyota, etc.

Inheritance

Inheritance is an OOPS concept in which one object acquires the properties and behaviours of the parent object. It's creating a parent-child relationship between two classes. It offers a robust and natural mechanism for organising and structuring any software. It provides code reusability. It is used to achieve **runtime polymorphism. For example,** let's say we have an Employee class. Employee class has all common attributes and methods which all employees must have within the organisation. There can be other specialised employees as well e.g. Manager. Managers are regular employees of the organisation but, additionally, they have few more attributes over other employees e.g. they have reports or subordinates.

Polymorphism

It refers to the ability of object-oriented programming languages to differentiate between entities with the same name efficiently. This is done by Java with the help of the signature and declaration of these entities. The ability to appear in many forms is called **polymorphism.**

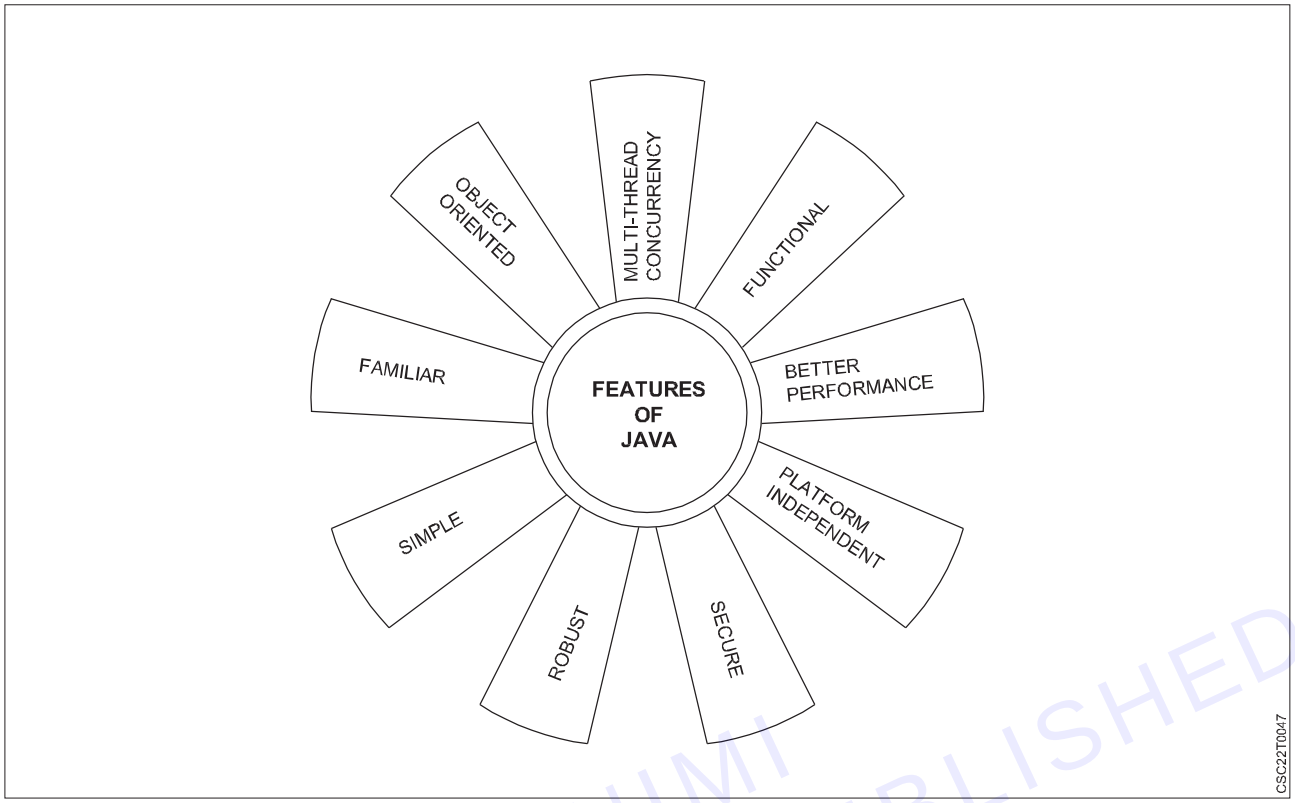
Abstraction

Abstraction means showing only the relevant details to the end-user and hiding the irrelevant features that serve as a distraction. For example, during an ATM operation, we only answer a series of questions to process the transaction without any knowledge about what happens in the background between the bank and the ATM.

Encapsulation

Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines.

Features of JAVA Programming



Java is simple

Java is Easy to write and more readable and eye

Java has a concise, cohesive set of features that makes it easy to learn and use.

Most of the concepts are drawn from C++ thus making Java learning simpler.

Java is secure

Java programs can not harm other systems thus secure. Java provides a secure means of creating Internet applications. Java provides a secure way to access web applications.

Java is portable

Java programs can execute in any environment which there is a Java run-time system.(JVM) Java programs can be run on any platform (Linux,Windows,Mac) for Java programs can be transferred over world wide web (e.g applets).

Java is object-oriented

Java programming is an object-oriented programming language.Like C++ java provides most of the object oriented features. Java is pure OOP. Language. (while C++ is semi object oriented)

Java is robust

Java encourages error-free programming by being strictly typed and performing run-time checks.

Java is multithreaded

Java provides integrated support for multithreaded programming.

Java is architecture-neutral

Java is not tied to a specific machine or operating system architecture. Machine Independent i.e Java is independent of hardware.

Java is interpreted

Java supports cross-platform code through the Java bytecode. use of Bytecode can be interpreted on any platform by JVM.

Java is distributed

Java was designed with the distributed environment. Java can be transmit, run over the internet.

JVM, Byte codes and Class path

JDK, JRE, and JVM are essential components in the world of Java programming, each serving a specific role in the development and execution of Java applications:

JDK (Java Development Kit)

The JDK is a software package provided by Oracle (and other vendors) that includes all the tools and libraries necessary for Java application development. It contains the following key components:

- Java Compiler (javac): This tool is used to compile Java source code (.java files) into bytecode (.class files), which can be executed by the Java Virtual Machine (JVM).
- Java Runtime Environment (JRE): The JRE is also included in the JDK. It is needed for running Java applications but not for developing them. The JRE consists of the JVM and essential libraries.
- Development Tools: The JDK includes various development tools like debugging tools, JavaDoc for generating documentation, and more.

  Developers use the JDK to write, compile, and package Java applications.

JRE (Java Runtime Environment)

The JRE is a subset of the JDK and is required to run Java applications. It includes the following components:

- Java Virtual Machine (JVM): The JVM is the runtime engine responsible for executing Java bytecode. It interprets and translates bytecode into machine-specific code that the underlying operating system can understand. Each platform (e.g., Windows, Linux, macOS) has its own JVM implementation,
- Java Standard Library: The JRE includes a set of core libraries and classes that provide essential functionality to Java applications. These libraries offer features like input/output operations, networking, and more.

End-users who want to run Java applications need the JRE installed on their system. It allows them to execute Java programs without the need for development tools.

JVM (Java Virtual Machine)

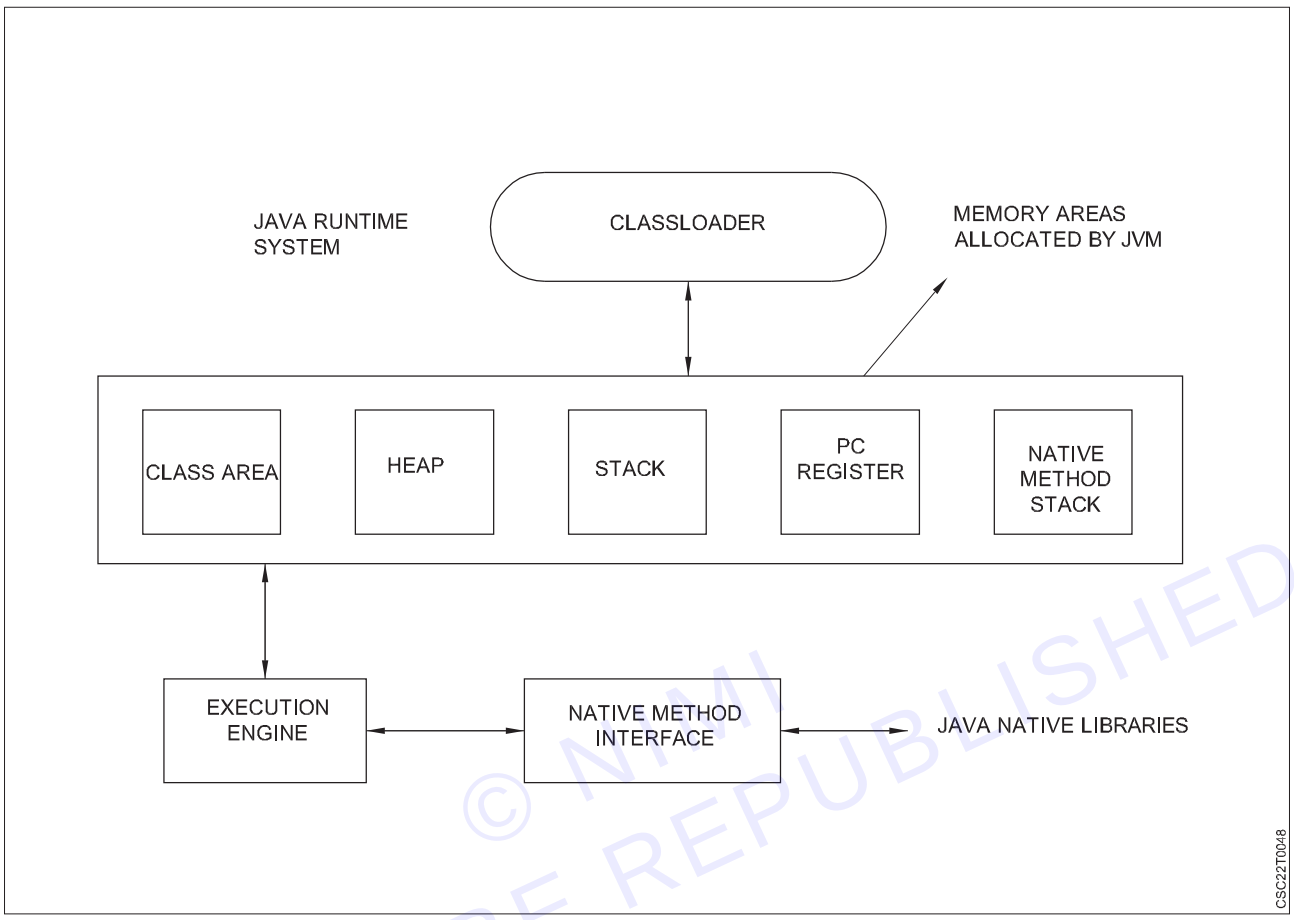
The JVM is the runtime environment in which Java bytecode is executed. It is an integral part of both the JDK and the JRE. The JVM performs several crucial tasks:

- **Loading:** It loads compiled Java bytecode (class files) into memory.
- **Verification:** The JVM checks the bytecode for security and integrity to prevent potential security vulnerabilities.
- **Execution:** The bytecode is executed by the JVM, which translates it into machine code specific to the underlying hardware and operating system.
- **Memory Management:** The JVM manages memory allocation and garbage collection to ensure efficient memory usage.
- **Security:** The JVM enforces Java's security model, including access control, class-loading restrictions, and more.

JVM Architecture

Java applications are often referred to as WORA, which stands for "Write Once Run Anywhere." This concept signifies that a programmer can write Java code on one system and anticipate it to run on any other Java-enabled system without the need for modifications. This seamless portability is made possible due to the presence of the Java Virtual Machine (JVM).

When a .java file is compiled, it produces **.class** files that contain bytecode and share the same class names as the corresponding **.java** files. When these **.class** files are executed, they undergo a series of steps that collectively define the functionality of the JVM (Java Virtual Machine).



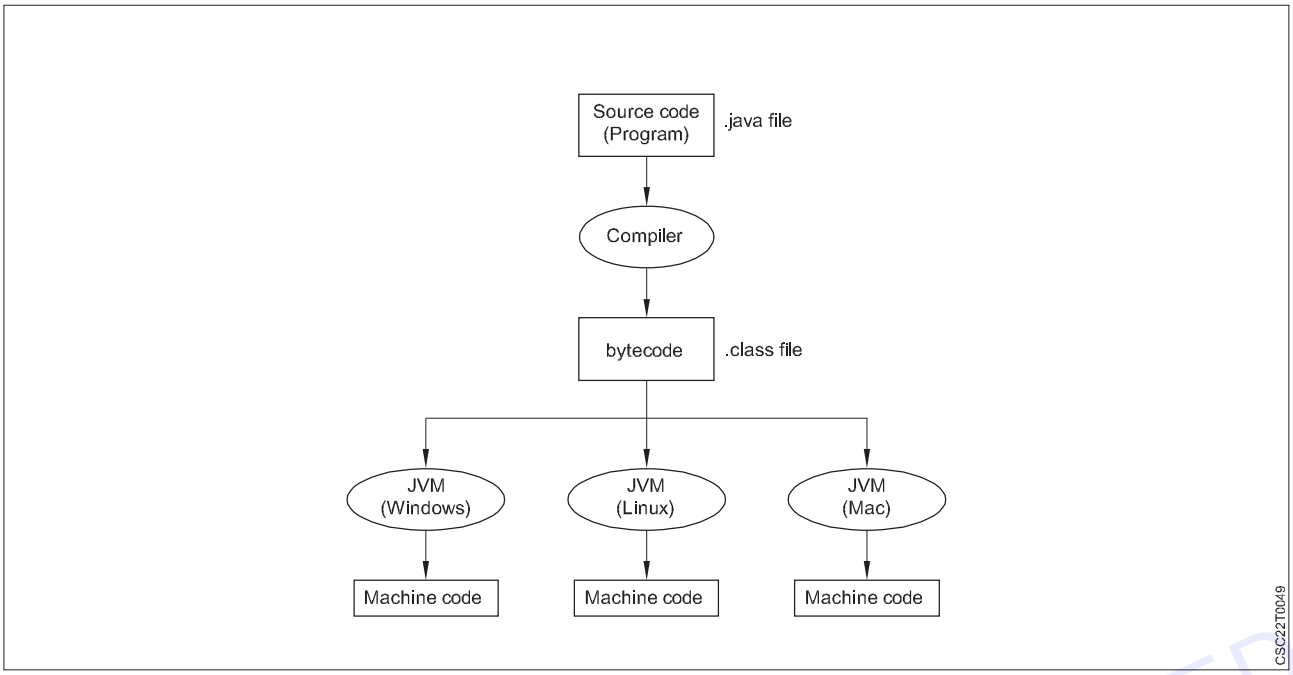
Java bytecode

Java bytecode serves as the instruction set for the Java Virtual Machine (JVM). It functions akin to an assembler, providing a symbolic representation of Java code, much like how C++ code can be represented in an alias form. When a Java program is compiled, it results in the generation of Java bytecode. To put it succinctly, Java bytecode can be thought of as machine code encapsulated in a .class file. The utilisation of Java bytecode is what enables Java to attain platform independence, ensuring that Java programs can run on different systems without modification.

How does it work?

When we develop a program in Java, the initial step involves compiling the code, resulting in the creation of bytecode. This bytecode serves as an intermediary representation of the code. Importantly, when we intend to execute this .class file on different platforms, we can readily do so. Following the initial compilation, it's the Java Virtual Machine (JVM), rather than the specific processor of the platform, that takes charge of running the bytecode.

In essence, this implies that we only need a basic Java installation on any platform where we wish to execute our code. The JVM plays a crucial role by managing the resources required for executing the bytecode. It communicates with the processor to allocate the necessary resources. It's worth noting that JVMs operate in a stack-based manner, using a stack implementation to interpret and execute the bytecode instructions.



What is the path ?

After installing Java on your computer, it's important to configure the PATH environment variable. This configuration allows you to conveniently execute Java-related executables (such as javac.exe, java.exe, javadoc.exe, etc.) from any directory without the need to specify the full path to the command.

```
C:\javac DemoClass.java
```

If you don't set the PATH environment variable, you would indeed have to specify the full path each time you run a Java command, like this:

```
C:\Java\jdk1.7.0\bin\javac DemoClass.java
```

Without the PATH variable configuration, this full path specification is necessary for running Java executables.

What is Classpath?

The classpath is a system environment variable utilised by both the Java compiler and the JVM (Java Virtual Machine).

It serves as a reference to help these components locate the necessary class files.

```
C:\Program Files\Java\jdk1.6.0\lib
```

Java Program Development

Developing, compiling, and executing Java programs involves several steps. Java is a popular programming language that follows a “write once, run anywhere” philosophy, meaning you can write code on one platform and run it on any platform that has a Java Virtual Machine (JVM) installed. Here’s a step-by-step guide:

1 **Install Java Development Kit (JDK):** To develop Java programs, you need to have the Java Development Kit (JDK) installed on your computer. You can download the JDK from the official Oracle website or choose an open-source alternative like OpenJDK. Make sure to set up the PATH environment variable to include the JDK’s bin directory.

2 **Choose a Text Editor or Integrated Development Environment (IDE):** You can write Java code using a simple text editor like Notepad or a more sophisticated Integrated Development Environment (IDE) like Eclipse, IntelliJ IDEA, or Visual Studio Code. IDEs often provide features like code completion, debugging, and project management.

3 **Write Your Java Code:** Create a new Java source code file with a .java extension. For example, you can create a file named MyProgram.java. Write your Java code in this file. Here’s a simple “Hello, World!” program in Java

```
public class MyProgram {
    public static void main(String[] args) {
      System.out.println("Hello, World!");
      }
}
```

4 Save Your Java File: Save your Java source code file in a directory of your choice.

5 Compile Your Java Code: Open your terminal or command prompt and navigate to the directory where you saved your Java source file. To compile the code, use the javac command followed by your Java source file’s name:

```
javac MyProgram.java
```

If there are no errors in your code, this will generate a bytecode file with a .class’ extension, such as ‘MyProgram.class’.

6 **Execute Your Java Program:** After successfully compiling your Java code, you can execute it using the ‘java’ command followed by the class name (excluding the ‘.class’ extension):

```
java MyProgram
```

If everything is correct, you should see the output of your program in the terminal:

```
Hello, World!
```

7 **Debugging and Testing:** Use debugging tools provided by your chosen IDE or standard tools like ‘System.out.println()’ to debug and test your Java code.

8 **Packaging and Distribution (if necessary):** If you want to distribute your Java program, you can package it into a JAR (Java Archive) file, which contains all the necessary class files and resources. This allows others to run your program easily.

Compilation and Execution of JAVA programs

Developing, compiling, and executing Java programs involves several steps. Java is a popular programming language that follows a “write once, run anywhere” philosophy, meaning you can write code on one platform and run it on any platform that has a Java Virtual Machine (JVM) installed. Here’s a step-by-step guide:

1 **Install Java Development Kit (JDK):** To develop Java programs, you need to have the Java Development Kit (JDK) installed on your computer. You can download the JDK from the official Oracle website or choose an open-source alternative like OpenJDK. Make sure to set up the PATH environment variable to include the JDK’s bin directory.

2 **Choose a Text Editor or Integrated Development Environment (IDE):** You can write Java code using a simple text editor like Notepad or a more sophisticated Integrated Development Environment (IDE) like Eclipse, IntelliJ IDEA, or Visual Studio Code. IDEs often provide features like code completion, debugging, and project management.

3 **Write Your Java Code:** Create a new Java source code file with a .java extension. For example, you can create a file named MyProgram.java. Write your Java code in this file. Here’s a simple “Hello, World!” program in Java

```
public class MyProgram {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

4 **Save Your Java File:** Save your Java source code file in a directory of your choice.

5 **Compile Your Java Code:** Open your terminal or command prompt and navigate to the directory where you saved your Java source file. To compile the code, use the javac command followed by your Java source file’s name

```
javac MyProgram.java
```

If there are no errors in your code, this will generate a bytecode file with’.class extension, such as **‘MyProgram.class’.**

6 **Execute Your Java Program:** After successfully compiling your Java code, you can execute it using the **‘java’** command followed by the class name (excluding the ‘.class’ extension):

```
java MyProgram
```

If everything is correct, you should see the output of your program in the terminal:

```
Hello, World!
```

7 **Debugging and Testing:** Use debugging tools provided by your chosen IDE or standard tools like ‘System.out.println()’ to debug and test your Java code.

8 **Packaging and Distribution (if necessary):** If you want to distribute your Java program, you can package it into a JAR (Java Archive) file, which contains all the necessary class files and resources. This allows others to run your program easily.

Basic JAVA language elements keywords, comments, data types and variables

Keywords

There are certain words with a specific meaning in java which tell (help) the compiler what the program is supposed to do. These Keywords cannot be used as variable names, class names, or method names. Keywords in java are case sensitive, all characters being lower case.

Keywords are reserved words that are predefined in the language; see the table below (Taken from Sun Java Site). All the keywords are in lowercase.

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	switch	
continue	goto	package	synchronized	

Comments

Comments are descriptions that are added to a program to make code easier to understand. The compiler ignores comments and hence its only for documentation of the program.

Java supports three comment styles.

Block style comments begin with /* and terminate with */ that spans multiple lines.

Line style comments begin with // and terminate at the end of the line. (Shown in the above program)

Documentation style comments begin with /** and terminate with */ that spans multiple lines. They are generally created using the automatic documentation generation tool, such as javadoc. (Shown in the above program)

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java

1 **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.

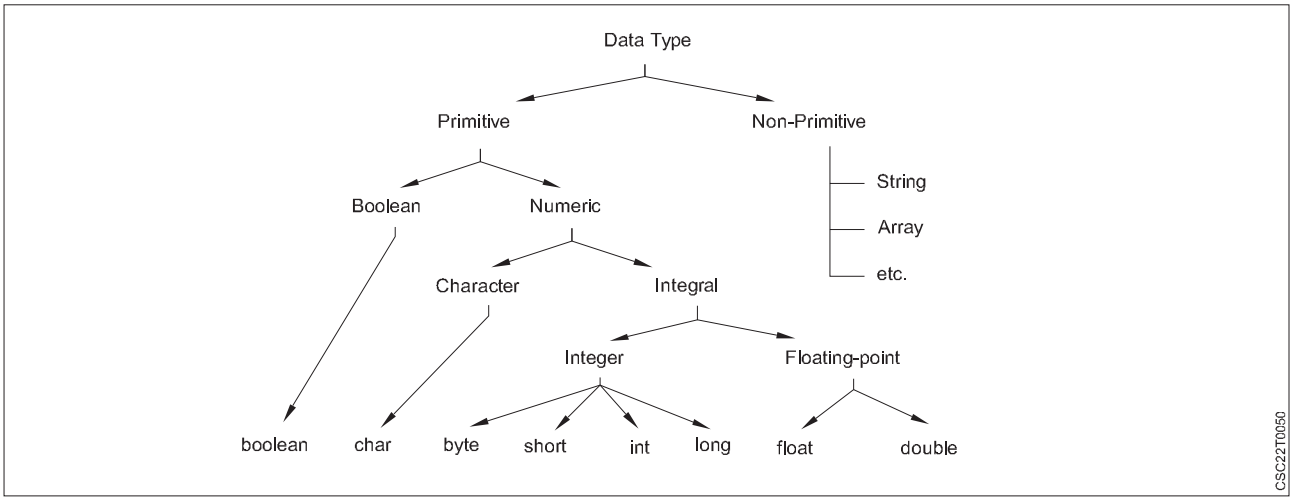
2 **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.

There are 8 types of primitive data types

- boolean data type
- byte data type
- char data type
- short data type
- int data type
- long data type
- float data type
- double data type



What is a Variable in Java?

Variable in Java is a data container that stores the data values during Java program execution. Every variable is assigned a data type which designates the type and quantity of value it can hold. Variable is a memory location name of the data. The Java variables have mainly three types : Local, Instance and Static.

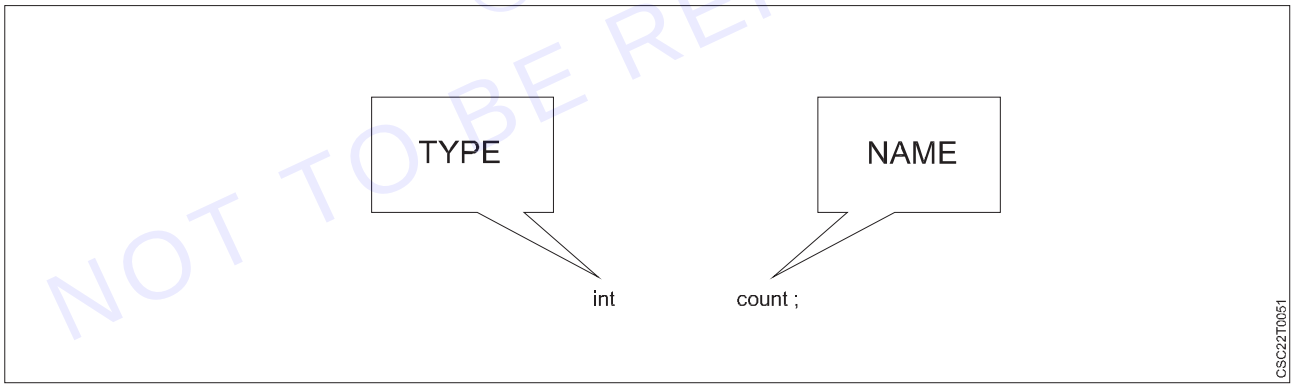
In order to use a variable in a program you to need to perform 2 steps

1 Variable Declaration

2 Variable Initialization

Variable Declaration

To declare a variable, you must specify the data type & give the variable a unique name.



Examples of other Valid Declarations are

```
int a,b,c;

float pi;

double d;

char a;
```

JAVA Arithmetic, Assignment, Relational, Logical, Increment / Decrement operators and expressions

Java Operators

Operators are used to perform operations on variables and values.

Java supports the following types of operators

- Arithmetic Operators
- Assignment Operators
- Logical Operators
- Relational Operators
- Unary Operators
- Bitwise Operators
- Ternary Operators
- Shift Operators

Arithmetic Operators in Java

Arithmetic Operators are used to performing mathematical operations like addition, subtraction, etc. Assume that A = 10 and B = 20 for the below table.

Operator	Description	Example
+ Addition	Adds values on either side of the operator	A+B=30
- Subtraction	Subtracts the right-hand operator with left-hand operator	A-B=-10
* Multiplication	Multiplies values on either side of the operator	A*B=200
/ Division	Divides left hand operand with right hand operator	A/B=0
% Modulus	Divides left hand operand by right hand operand and returns remainder	A%B=0

```
// Java code to illustrate Addition operator

import java.io.*;

class Addition {
    public static void main(String[] args)
    {
        // initializing variables
        int num1 = 10, num2 = 20, sum = 0;

        // Displaying num1 and num2
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2);

        // adding num1 and num2
        sum = num1 + num2;
        System.out.println("The sum = " + sum);
    }
}
```

Output : num1 = 10 num2 = 20

The sum = 30

Assignment Operators

- **= (Assignment):** Assigns the value of the right operand to the left operand.
- **+= (Addition Assignment):** Adds the right operand to the left operand and assigns the result to the left operand.
- **-= (Subtraction Assignment):** Subtracts the right operand from the left operand and assigns the result to the left operand.
- ***= (Multiplication Assignment):** Multiplies the left operand by the right operand and assigns the result to the left operand.
- **/= (Division Assignment):** Divides the left operand by the right operand and assigns the result to the left operand.
- **%= (Modulus Assignment):** Computes the modulus of the left operand and the right operand and assigns the result to the left operand.

Example: int x = 10;

x += 5; // x is now 15

x -= 3; // x is now 12

x *= 2; // x is now 24

x /= 4; // x is now 6

x %= 3; // x is now 0

Relational Operators

- == (Equal to): Checks if two operands are equal.
- != (Not equal to): Checks if two operands are not equal.
- < (Less than): Checks if the left operand is less than the right operand.
- > (Greater than): Checks if the left operand is greater than the right operand.
- <= (Less than or equal to): Checks if the left operand is less than or equal to the right operand.
- >= (Greater than or equal to): Checks if the left operand is greater than or equal to the right operand.

Example

int num1 = 10, num2 = 20;

boolean isEqual = (num1 == num2); // false

boolean isNotEqual = (num1 != num2); // true

boolean isLessThan = (num1 < num2); // true

boolean isGreaterThan = (num1 > num2); // false

Logical Operators

- && (Logical AND): Returns true if both operands are true.
- || (Logical OR): Returns true if at least one operand is true.
- ! (Logical NOT): Returns the opposite boolean value of the operand.

Example

boolean isTrue = true, isFalse = false;

boolean result1 = isTrue && isFalse; // false

boolean result2 = isTrue || isFalse; // true

boolean result3 = !isTrue; // false

Increment and Decrement Operators

- ++ (Increment): Increases the value of the operand by 1.
- -- (Decrement): Decreases the value of the operand by 1.

Example:1

int count = 5;

count++; // count is now 6

count--; // count is now 5

Example:2

int j = 0, k = 0; // Initially both j and k are 0

j = ++k; // Final values of both j and k are 1

Example:3

int i = 0, k = 0; // Initially both i and k are 0

i = k++; // Final value of i is 0 and k is

Operator	Description
()	Parentheses (grouping)
[]	Array subscript
.	Member access (dot operator)
++ and --	Post-increment and post-decrement
+ (unary) and - (unary)	Unary plus and unary minus
! and ~	Logical NOT and bitwise NOT
*, /, and %	Multiplication, division, and modulo
+ and -	Addition and subtraction
<< and >>	Bitwise left shift and bitwise right shift
<, <=, >, and >=	Relational operators (less than, less/equal, greater than, greater/equal)
instanceof	Type comparison (checks if an object is an instance of a class or interface)
== and !=	Equality and inequality (equals and not equals)
&	Bitwise AND
^	Bitwise XOR
`	`
&&	Conditional AND (short-circuiting)
`	
? :	Ternary conditional (conditional operator)
=	Assignment
+=, -=, *=, /=, %=	Assignment with addition, subtraction, multiplication, division, and modulo
&=, ^=, `	=`
<<=, >>=, >>>=	Assignment with bitwise left shift, right shift, and unsigned right shift

Examples of Operator

```
public class OperatorDemo {
public static void main(String[] args) {
// Arithmetic operators
int num1 = 10;
int num2 = 5;
int sum = num1 + num2;
int difference = num1 - num2;
int product = num1 * num2;
int quotient = num1 / num2;
int remainder = num1 % num2;
System.out.println("Arithmetic Operators:");
System.out.println("Sum: " + sum);
System.out.println("Difference: " + difference);
System.out.println("Product: " + product);
System.out.println("Quotient: " + quotient);
System.out.println("Remainder: " + remainder);

 // Assignment operators
int x = 5;
x += 3; // Equivalent to x = x + 3
System.out.println("\nAssignment Operators:");
System.out.println("x after assignment: " + x);

// Relational operators
int a = 10; int b = 20;
boolean isEqual = (a == b);
boolean isNotEqual = (a != b);
boolean isGreater = (a > b);
boolean isLess = (a < b);
System.out.println("\nRelational Operators:");
System.out.println("Is equal: " + isEqual);
System.out.println("Is not equal: " + isNotEqual);
System.out.println("Is greater: " + isGreater);
System.out.println("Is less: " + isLess);

// Logical operators
boolean p = true;
boolean q = false;
```

```
boolean logicalAnd = p && q;
boolean logicalOr = p || q;
boolean logicalNot = !p;
System.out.println("\nLogical Operators:");
System.out.println("Logical AND: " + logicalAnd);
System.out.println("Logical OR: " + logicalOr);
System.out.println("Logical NOT: " + logicalNot);

// Increment and Decrement operators
int count = 5; count++; // Increment by 1
int anotherCount = 10;
anotherCount--; // Decrement by 1
System.out.println("\nIncrement/Decrement Operators:");
System.out.println("count after increment: " + count);
System.out.println("anotherCount after decrement: " + anotherCount);
  }
}
```

Output

Arithmetic Operators

Sum: 15

Difference: 5

Product: 50

Quotient: 2

Remainder: 0

Assignment Operators

x after assignment: 8

Relational Operators

Is equal: false

Is not equal: true

Is greater: false

Is less: true

Logical Operators:

Logical AND: false

Logical OR: true

Logical NOT: false

Increment/Decrement Operators

count after increment: 6

anotherCount after decrement: 9

JAVA String Operators

In Java, there aren't specific "string operators" like you might find in languages like JavaScript or Python. Instead, Java primarily relies on methods and concatenation to perform operations on strings. Here are some common string operations and techniques in Java:

For Example:-

1 Concatenation Operator +:

- You can use the + operator to concatenate (join) strings together.

```
String firstName = "John";
String lastName = "Doe";
String fullName = firstName + " " + lastName;
```

Example

```
class Concat_Operator {
   public static void main( String args[] ) {
      String first = "Hello";
      String second = "World";

      String third = first + second;
      System.out.println(third);

      // yet another way to concatenate strings
      first += second;
      System.out.println(first);
   }
}
```

Output

```
HelloWorld
HelloWorld
```

2 Concatenation with Other Data Types

- You can also use the + operator to concatenate strings with other data types, which automatically converts the non-string operands to strings.

```
int age = 30;
String message = "My age is " + ag
```

Example

```
public class Demo {
  public static void main(String args[]){
    String st1 = "Hello";
    int age = 500;
```

```
    String res = st1+age;
    System.out.println(res);
  }
}
```

Output

Hello500

3 String Methods

- Java provides a wide range of methods for working with strings, such as **length(), charAt(), substring(), indexOf(), toUpperCase(), toLowerCase(),** and many others. These methods allow you to perform various operations on strings.

```
String text = "Hello, World!";
int length = text.length(); // 13
char firstChar = text.charAt(0); // 'H'
String substring = text.substring(7); // "World!"
```

4 String Comparison

- You can compare strings in Java using the equals() method for content-based comparison and == for reference-based comparison. It's important to use equals() when comparing the contents of two strings because == checks if two string references point to the same memory location.

```
String str1 = "Hello";
String str2 = "Hello";
boolean areEqual = str1.equals(str2); // true
```

Example

```
class Test {
 public static void main(String args[]){
   String s1="Rachin";
   String s2="Rachin";
   String s3=new String("Rachin");
//true (because both refer to same instance)
   System.out.println(s1==s2);
//false(because s3 refers to instance created in nonpool)
   System.out.println(s1==s3);
 }
}
```

Output

true

false

5 String Formatting

- Java provides the **String.format()** method and the **printf()** method for formatting strings using placeholders and format specifiers, similar to C's **printf().**

```
String formatted = String.format("Hello, %s!", "John");
System.out.println(formatted); // Hello, John!
```

Example

```
public class FormatExample{
public static void main(String args[]){
String name="Pradyumn";
String sf1=String.format("name is %s",name);
String sf2=String.format("value is %f",32.33434);

//returns 12 char fractional part filling with 0

System.out.println(sf1);
System.out.println(sf2);
}
}
```

Output

name is Pradyumn

value is 32.334340

6 StringBuilder and StringBuffer

- For efficient string manipulation, especially when you need to concatenate strings in a loop or modify them frequently, you can use the StringBuilder (not thread-safe) or StringBuffer (thread-safe) classes.

```
StringBuilder stringBuilder = new StringBuilder();
stringBuilder.append("Hello, ");
stringBuilder.append("World!");
String result = stringBuilder.toString();
```

Example

```
public class BufferTest{
   public static void main(String[] args){
      StringBuffer buffer=new StringBuffer("hello");
      buffer.append("java");
      System.out.println(buffer);
   }
}
```

Output

hellojava

Example

```
public class BuilderTest{
   public static void main(String[] args){
      StringBuilder builder=new StringBuilder("hello");
      builder.append("Pradyumn");
      System.out.println(builder);
   }
}
```

Output

Indeed, the Java String class offers a wide range of methods for performing various operations on strings. Some of the commonly used String methods include:

- **compare():** Compares two strings lexicographically.
- **concat():** Concatenates one string with another.
- **equals():** Compares two strings for equality.
- **split():** Splits a string into an array of substrings based on a delimiter.
- **length():** Returns the length (number of characters) of a string.
- **replace():** Replaces occurrences of a specified substring with another substring.
- **compareTo():** Compares two strings lexicographically and returns an integer.
- **intern():** Returns a canonical representation of the string.
- **substring():** Extracts a portion of the string based on given indices etc.

There are two ways to create String object

1 By string literal

2 By new keyword

<String_Type> <string_variable> = "<sequence_of_string>";

1 String literal

One of the advantages of using the `intern()` method in Java is to enhance memory efficiency. This is achieved by preventing the creation of new objects if an identical string already exists in the string constant pool.

Example:

String demoString = "ctiworld";

=

In the statement **String s = new String("Welcome");**, the JVM operates as follows:

It creates a new string object in the regular (non-pool) heap memory, and the literal "Welcome" is placed in the string constant pool. Consequently, the variable s will point to the object in the heap (non-pool) memory.

Example

String demoString = new String ("ctiworld");

Interfaces and Classes in Strings in Java

CharBuffer is a class that implements the CharSequence interface. Its primary purpose is to enable the substitution of character buffers for CharSequences. An illustrative application of this capability can be found in the java.util.regex package, where CharBuffer is used.

A String is essentially a sequence of characters. In Java, String objects are immutable, signifying that they are constant and cannot be altered after their creation

CharSequence Interface: The CharSequence interface in Java serves as a representation for sequences of characters. Below are some of the classes that implement the CharSequence interface

1 String

2 StringBuffer

3 StringBuilder

1 **StringBuffer:** StringBuffer is a companion class to String in Java, and it offers extensive functionality for manipulating character sequences. While strings in Java are immutable and represent fixed-length character sequences, StringBuffer represents character sequences that can dynamically grow and be modified.

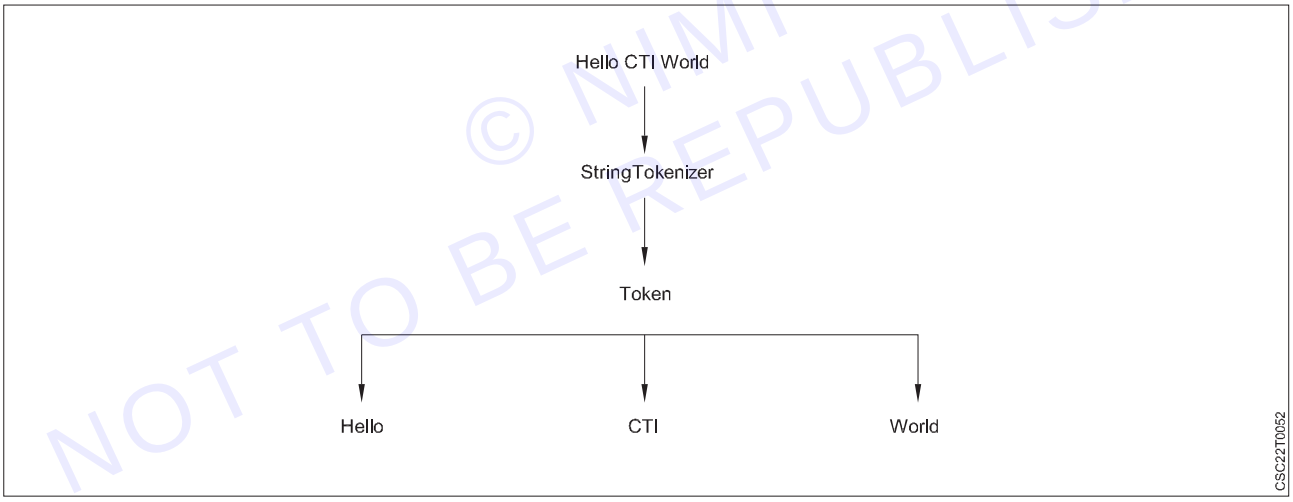
StringBuffer demoString = new StringBuffer("ctiworld");

2 **StringBuilder:** In Java, the StringBuilder class represents a mutable sequence of characters. Unlike the String class, which creates immutable character sequences, StringBuilder provides an alternative that allows you to create and modify mutable character sequences

StringBuilder demoString = new StringBuilder();
demoString.append("CTI");

3 **StringTokenizer:** The StringTokenizer class in Java is employed for splitting a string into tokens or smaller components.

Example



Indeed, a StringTokenizer object in Java keeps track of an internal current position within the string that it's tokenizing. Certain operations cause this position to move forward as characters are processed. To obtain a token, the StringTokenizer object extracts a substring from the original string used to create it.

StringJoiner is a class within the java.util package in Java, designed for constructing a sequence of characters (strings) separated by a specified delimiter. It also offers the option to begin with a provided prefix and end with a supplied suffix. While similar functionality can be achieved using the StringBuilder class by manually appending a delimiter after each string, StringJoiner simplifies this process, reducing the amount of code you need to write.

Syntax:

public StringJoiner(CharSequence delimiter)

As mentioned earlier, you can create a string in Java using a string literal. A string literal is a sequence of characters enclosed within double quotation marks

String myString = "Hello, World!";

You're absolutely correct! In Java, when you create a string using a string literal, the JVM checks the String Constant Pool. If an identical string exists in the pool, it returns a reference to that existing string instead of creating a new object. This behaviour is part of string interning, which helps conserve memory by reusing identical string instances.

Immutable String in Java

In Java, string objects are immutable, which means they cannot be modified or changed after creation. Instead, any operations that appear to modify a string actually create a new string object with the desired changes.

```
class Demo {
   public static void main(String[] args)
   {
      String name = "Sachin";
      name.concat(" Tendulkar");
      System.out.println(name);
   }
}
```

Output :-

Sachin

```
class Demo {
   public static void main(String[] args)
   {
      String name = "Sachin";
       name = name.concat(" Tendulkar");
      System.out.println(name);
   }
}
```

Output :-

Sachin Tendulkar

JAVA Input and Output streams(I/O)

Java I/O, which stands for Input and Output, is a fundamental part of Java programming used for handling data input and producing output in various applications.

Java leverages the concept of streams to optimise I/O operations, and the java.io package encompasses all the necessary classes for handling input and output tasks efficiently.

File handling in Java can be achieved through the Java I/O API.

Stream

In Java, a stream is a sequence of elements that you can process in a functional and declarative manner. Java Streams were introduced in Java 8 and provide a powerful way to work with collections, arrays, and other data sources. Streams allow you to perform operations on the elements of a sequence, such as filtering, mapping, and reducing, without the need for explicit loops.

In Java, three streams are automatically created for us, all of which are associated with the console:

1 **`System.in`:** This represents the standard input stream, which serves the purpose of reading characters from the keyboard or any other standard input source.

2 **`System.out`:** This refers to the standard output stream, utilised for displaying the program's results on an output device, such as the computer screen. Here is a list of various print functions used for outputting statements

- **'print()':** In Java, this method is employed to exhibit text on the console. The text is supplied as a parameter in the form of a string. When invoked, this method displays the text on the console while keeping the cursor at the end of the printed text. Subsequent printing operations will begin from this point.

Syntax:

```
System.out.print(parameter);
```

```
// Java code to illustrate print()
import java.io.*;
class Demo {
    public static void main(String[] args)
    {
        // using print()
        // all are printed in the
        // same line
        System.out.print("CTI WORLD ");
        System.out.print("CTI WORLD ");
        System.out.print("CTI WORLD ");
    }
}
```

Output:

CTI WORLD CTI WORLD CTI WORLD

- println(): In Java, this method serves the purpose of showcasing text on the console. It outputs the text to the console and positions the cursor at the beginning of the next line. Subsequent printing operations will begin from the next line

Syntex

```
System.out.println(parameter);
```

```
// Java code to illustrate println()

import java.io.*;
class Demo {
    public static void main(String[] args)
    {

        // using println()
        // all are printed in the
        // different line
        System.out.println("CSA WORLD ");
        System.out.println("CSA WORLD ");
        System.out.println("CSA WORLD ");
    }
}
```

Output:

CSA WORLD
CSA WORLD
CSA WORLD

- **printf():** This method in Java is reminiscent of the printf function in C. It is worth noting that while System.out. print() and System.out.println() accept a single argument, printf() can accept multiple arguments. Its primary purpose is to format the output in Java.

```
public class PrintfExample {
        public static void main(String[] args) {
   String name = "ANSHU";
   int age = 24;
   double salary = 50000.75;
   // Using printf to format output
     System.out.printf("Name: %s%n", name);
     System.out.printf("Age: %d%n", age);
     System.out.printf("Salary: $%.2f%n", salary);
      }
 }
```

Output

Name: ANSHU

Age: 34

Salary: 50000.75

In this example

- %s is a placeholder for a string (name).
- %d is a placeholder for an integer (age).
- %.2f is a placeholder for a floating-point number (salary) with 2 decimal places.

The %n is used to insert a platform-independent line separator.

Types of Streams

Streams can be categorised into two main classes based on the type of operations they are used for

1 **Input Stream:** These streams are employed to retrieve data as input from various sources such as arrays, files, or peripheral devices. Examples include FileInputStream, BufferedInputStream, and ByteArrayInputStream

2 **Output Stream:** These streams are used to write data as output to various destinations such as arrays, files, or output peripheral devices. Examples include FileOutputStream, BufferedOutputStream, and ByteArrayOutputStream

Input using Scanner class and Console class methods

In Java, you can use the Scanner class and the Console class to interact with the user via the command line for input and output. Here are examples of how to use both classes for input:

Scanner class

In Java, the Scanner class from the java.util package is employed for acquiring input of primitive types such as int, double, as well as strings.

While using the Scanner class is the simplest method for reading input in a Java program, it may not be the most efficient choice for situations where input processing time is critical, such as competitive programming.

Syntax

Scanner sc=new Scanner(System.in);

Methods of Java Scanner Class

- nextBoolean(): Used for reading a Boolean value.
- nextByte(): Used for reading a Byte value.
- nextDouble(): Used for reading a Double value.
- nextFloat(): Used for reading a Float value.
- nextInt(): Used for reading an Int value.
- nextLine(): Used for reading a Line value (usually a String).
- nextLong(): Used for reading a Long value.
- nextShort(): Used for reading a Short value.

“Let’s examine a code snippet that demonstrates how to read input of various data types.”

Example

```
// Java program to read data of various types using Scanner
// class.
import java.util.Scanner;
public class ScannerDemo1 {
    public static void main(String[] args)
    {
```

```
        // Declare the object and initialise with
        // predefined standard input object
        Scanner sc = new Scanner(System.in);
        // String input
        String name = sc.nextLine();
        // Character input
        char gender = sc.next().charAt(0);
        // Numerical data input
        // byte, short and float can be read
// using similar-named functions.
        int age = sc.nextInt();
        double cgpa = sc.nextDouble();


        // Print the values to check if the input was
        // correctly obtained.
        System.out.println("Name: " + name);
        System.out.println("Gender: " + gender);
        System.out.println("Age: " + age);
        System.out.println("CGPA: " + cgpa);
    }
}
```

1 It imports the Scanner class to facilitate user input.

2 In the main method:

- It creates a Scanner object sc to read input from the standard input stream (System.in).
- It uses sc.nextLine() to read a line of input as a String and stores it in the name variable.
- It uses sc.next().charAt(0) to read the next token (word) as a String and then extracts the first character of that string as a char, storing it in the gender variable.
- It uses sc.nextInt() to read the next token as an int and stores it in the age variable.
- It uses sc.nextDouble() to read the next token as a double and stores it in the cgpa variable.

3 After reading all the values, it prints them to the console to verify that the input was correctly obtained.

Input

AKASH MOHAN

Male

24

91

Output

Name: AKASH MOHAN

Gender: M

Age: 24

CGPA: 91.0

Sometimes, it's necessary to verify if the next value we're about to read is of a specific data type or if the input has reached its end (EOF marker encountered).

To accomplish this, we can utilise the hasNextXYZ() functions, where XYZ represents the type we are interested in. These functions return true if the Scanner has a token of the specified type, and false otherwise. For example, in the code below, we have employed hasNextInt() to check for an integer input. To check for a string, we use hasNextLine(), and for a single character, we use hasNext().charAt(0)

Let's review a code snippet for reading numbers from the console and calculating their mean

```
// Java program to read some values using Scanner
// class and print their mean.
import java.util.Scanner;
 public class ScannerDemo2 {
    public static void main(String[] args)
    {
        // Declare an object and initialise with
        // predefined standard input object
        Scanner sc = new Scanner(System.in);
         // Initialize sum and count of input elements
        int sum = 0, count = 0;
         // Check if an int value is available
        while (sc.hasNextInt()) {
            // Read an int value
            int num = sc.nextInt();
            sum += num;
            count++;
        }
        if (count > 0) {
            int mean = sum / count;
            System.out.println("Mean: " + mean);
        }
        else {
            System.out.println(
                "No integers were input. Mean cannot be calculated.");
        }
    }
}
```

This Java program reads integer values from the standard input using the Scanner class and calculates their mean (average). Here's a breakdown of how the program works:

1 It imports the Scanner class to facilitate user input.

2 In the main method

- It creates a Scanner object sc to read input from the standard input stream (System.in).
- Initialises two variables, sum and count, to keep track of the sum of input values and the count of input values, respectively.

- Uses a while loop to continuously check if the next input is an integer using sc.hasNextInt(). If an integer is available, it reads the integer value using sc.nextInt(), adds it to the sum, and increments the count.

3 After the loop, it checks whether any integers were input (i.e., count > 0). If count is greater than zero, it calculates the mean (average) by dividing sum by count and then prints the mean value. If no integers were input (i.e., count is still zero), it prints a message stating that the mean cannot be calculated.

Input

1

2

3

4

5

Output

Mean: 3

- Importing the Class: To use the Scanner class, you need to import it with import java.util.Scanner;.
- Reading from Standard Input: The Scanner class is commonly used to read input from the standard input stream (System.in) by creating a Scanner object with System.in as the argument.
- Reading from Files: You can also read input from files by passing a File object as an argument to the Scanner constructor.
- Reading Different Data Types: Scanner provides methods like nextByte(), nextInt(), nextLong(), nextFloat(), nextDouble(), etc., to read different data types.
- Reading Strings: To read strings, you can use next() or nextLine() methods. next() reads the next token (usually a word), while nextLine() reads the entire line.
- Reading Characters: You can read a single character by combining next() and charAt(0), as in next().charAt(0).
- Delimiter: By default, Scanner uses whitespace as the delimiter to separate tokens. You can change the delimiter using the useDelimiter() method.
- Checking for Input: You can check if there is more input to read with methods like hasNext(), hasNextInt(), hasNextLine(), etc.
- Closing the Scanner: It's important to close the Scanner object using close() when you're done with it to release resources.
- Handling Input Errors: Always validate input using methods like hasNextInt() or exception handling to prevent unexpected program behaviour due to invalid input.
- Tokenization: Scanner reads input and tokenizes it. Tokens are smaller units of input separated by the specified delimiter.
- Locale and Locale-Sensitive Parsing: You can set the locale to influence how Scanner parses numbers and strings. For example, comma (,) or period (.) as a decimal separator.
- Not Suitable for Competitive Programming: While Scanner is convenient, it may not be the most efficient choice for scenarios where input processing time is critical, such as competitive programming.
- Exception Handling: Be prepared to handle exceptions like InputMismatchException or NoSuchElementException that may occur if the input doesn't match the expected format.
- Thread Safety: Scanner is not thread-safe, so avoid sharing Scanner objects between multiple threads without proper synchronisation.
- Unicode Support: Scanner supports Unicode characters, allowing you to read and process text in various languages.

LESSON 85 - 93 : JAVA Program Flow Control

Java Control Statements / Control Flow in Java

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.

Java provides three types of control flow statements.

1 Decision Making statements

- if statements
- switch statement

2 Loop statements

- do while loop
- while loop
- for loop
- for-each loop

3 Jump statements

- break statement
- continue statement

Decision-Making statements:

As the name suggests, decision-making statements decide which statement to execute and when. Decision-making statements evaluate the Boolean expression and control the program flow depending upon the result of the condition provided. There are two types of decision-making statements in Java, i.e., If statement and switch statement.

1W If Statement:

In Java, the “if” statement is used to evaluate a condition. The control of the program is diverted depending upon the specific condition. The condition of the If statement gives a Boolean value, either true or false. In Java, there are four types of if-statements given below.

1 Simple if statement

2 if-else statement

3 if-else-if ladder

4 Nested if-statement

Let’s understand the if-statements one by one.

1 Simple if statement:

It is the most basic statement among all control flow statements in Java. It evaluates a Boolean expression and enables the program to enter a block of code if the expression evaluates to true.

Syntax of if statement is given below.

```
f(condition) {
statement 1; //executes when condition is true
}
```

Consider the following example in which we have used the if statement in the java code.

Student.java

Student.java

```
public class Student {
public static void main(String[] args) {
int x = 10;
int y = 12;
if(x+y > 20) {
System.out.println(“x + y is greater than 20”);
}
}
}
```

Output:

x + y is greater than 20

2 if-else statement

The if-else statement is an extension to the if-statement, which uses another block of code, i.e., else block. The else block is executed if the condition of the if-block is evaluated as false.

Syntax:

```
if(condition) {
statement 1; //executes when condition is true
}
else{
statement 2; //executes when condition is false
}
```

Consider the following example.

Student.java

```
public class Student {
public static void main(String[] args) {
int x = 10;
int y = 12;
if(x+y < 10) {
System.out.println(“x + y is less than 10”);
```

```
} else {
System.out.println("x + y is greater than 20");
}
}
}
```

Output:

x + y is greater than 20

3 if-else-if ladder:

The if-else-if statement contains the if-statement followed by multiple else-if statements. In other words, we can say that it is the chain of if-else statements that create a decision tree where the program may enter in the block of code where the condition is true. We can also define an else statement at the end of the chain.

Syntax of if-else-if statement is given below.

```
if(condition 1) {
statement 1; //executes when condition 1 is true
}
else if(condition 2) {
statement 2; //executes when condition 2 is true
}
else {
statement 2; //executes when all the conditions are false
}
```

Consider the following example.

Student.java

```
public class Student {
public static void main(String[] args) {
String city = "Delhi";
if(city == "Meerut") {
System.out.println("city is meerut");
}else if (city == "Noida") {
System.out.println("city is noida");
}else if(city == "Agra") {
System.out.println("city is agra");
}else {
System.out.println(city);
}
}
}
```

Output:

Delhi

Nested if-statement

In nested if-statements, the if statement can contain a if or if-else statement inside another if or else-if statement.

Syntax of Nested if-statement is given below.

```
if(condition 1) {
statement 1; //executes when condition 1 is true
if(condition 2) {
statement 2; //executes when condition 2 is true
}
else{
statement 2; //executes when condition 2 is false
}
}
```

Consider the following example.

Student.java

```
public class Student {
public static void main(String[] args) {
String address = “Delhi, India”;
if(address.endsWith(“India”)) {
if(address.contains(“Meerut”)) {
System.out.println(“Your city is Meerut”);
}else if(address.contains(“Noida”)) {
System.out.println(“Your city is Noida”);
}else {
System.out.println(address.split(“,”)[0]);
}
}else {
System.out.println(“You are not living in India”);
}
}
}
```

Output:

Delhi

Switch Statement:

In Java, Switch statements are similar to if-else-if statements. The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched. The switch statement is easier to use instead of if-else-if statements. It also enhances the readability of the program.

Points to be noted about switch statement:

- The case variables can be int, short, byte, char, or enumeration. String type is also supported since version 7 of Java
- Cases cannot be duplicate

- Default statement is executed when any of the case doesn't match the value of expression. It is optional.
- Break statement terminates the switch block when the condition is satisfied.

  It is optional, if not used, next case is executed.
- While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value.

The syntax to use the switch statement is given below.

```
switch (expression){
case value1:
statement1;
break;
.
.
.
case valueN:
statementN;
break;
default:
default statement;
}
```

Consider the following example to understand the flow of the switch statement.

Student.java

```
public class Student implements Cloneable {
public static void main(String[] args) {
int num = 2;
switch (num){
case 0:
System.out.println("number is 0");
break;
case 1:
System.out.println("number is 1");
break;
default:
System.out.println(num);
}
}
}
```

Output:

2

While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value. The switch permits only int, string, and Enum type variables to be used.

Loop Control Flow Using

Loop Statements

In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true. However, loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.

In Java, we have three types of loops that execute similarly. However, there are differences in their syntax and condition checking time.

1 for loop

2 while loop

3 do-while loop

Let's understand the loop statements one by one.

Java for loop

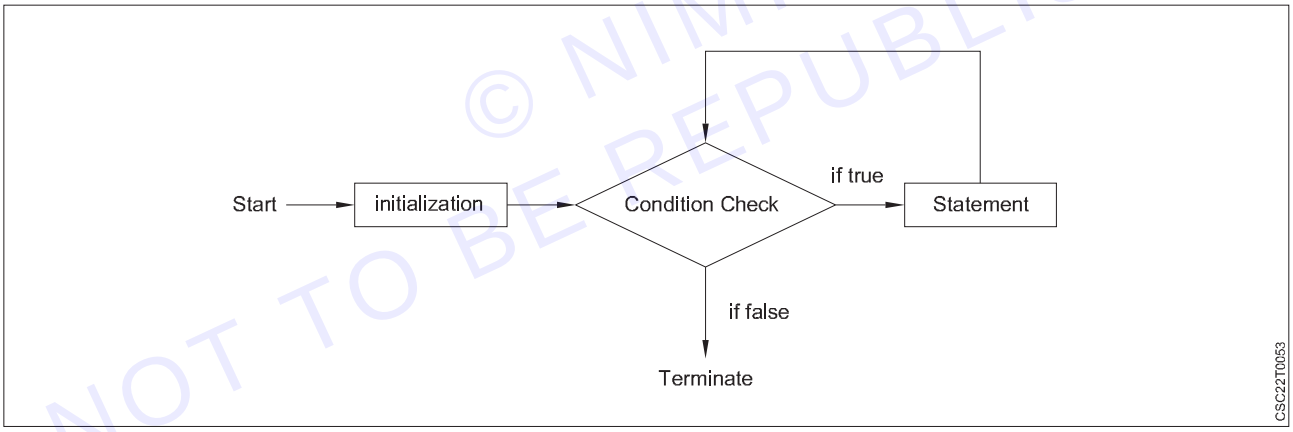
In Java, for loop is similar to C and C++. It enables us to initialize the loop variable, check the condition, and increment/decrement in a single line of code. We use the for loop only when we exactly know the number of times, we want to execute the block of code.

for(initialization, condition, increment/decrement) {

//block of statements

}

The flow chart for the for-loop is given below.



Consider the following example to understand the proper functioning of the for loop in java.

Calculation.java

```
public class Calculattion {
public static void main(String[] args) {
// TODO Auto-generated method stub
int sum = 0;
for(int j = 1; j<=10; j++) {
sum = sum + j;
}
System.out.println("The sum of first 10 natural numbers is " + sum);
```

}

}

Output:

The sum of first 10 natural numbers is 55

Java for-each loop

Java provides an enhanced for loop to traverse the data structures like array or collection. In the for-each loop, we don't need to update the loop variable. The syntax to use the for-each loop in java is given below.

for(data_type var : array_name/collection_name){

//statements

}

Consider the following example to understand the functioning of the for-each loop in Java.

Calculation.java

public class Calculation {

public static void main(String[] args) {

// TODO Auto-generated method stub

String[] names = {"Java","C","C++","Python","JavaScript"};

System.out.println("Printing the content of the array names:\n");

for(String name:names) {

System.out.println(name);

}

}

}

Output:

Printing the content of the array names:

Java

C

C++

Python

JavaScript

Java while loop

The while loop is also used to iterate over the number of statements multiple times. However, if we don't know the number of iterations in advance, it is recommended to use a while loop. Unlike for loop, the initialization and increment/decrement doesn't take place inside the loop statement in while loop.

It is also known as the entry-controlled loop since the condition is checked at the start of the loop. If the condition is true, then the loop body will be executed; otherwise, the statements after the loop will be executed.

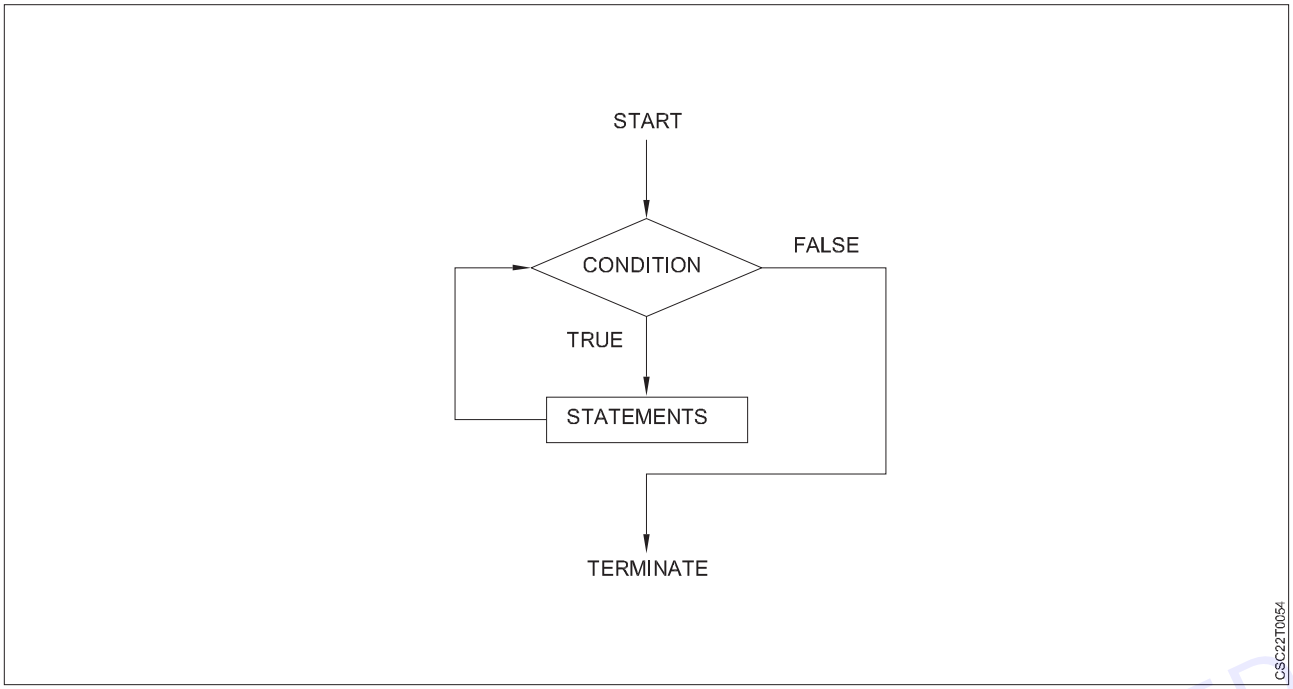
The syntax of the while loop is given below.

while(condition){

//looping statements

}

The flow chart for the while loop is given in the following image.



Consider the following example to understand the functioning of the do-while loop in Java.

Calculation .java

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbers \n");
while(i<=10) {
System.out.println(i);
i = i + 2;
}
}
}
```

Output:

Printing the list of first 10 even numbers

0

2

4

6

8

10

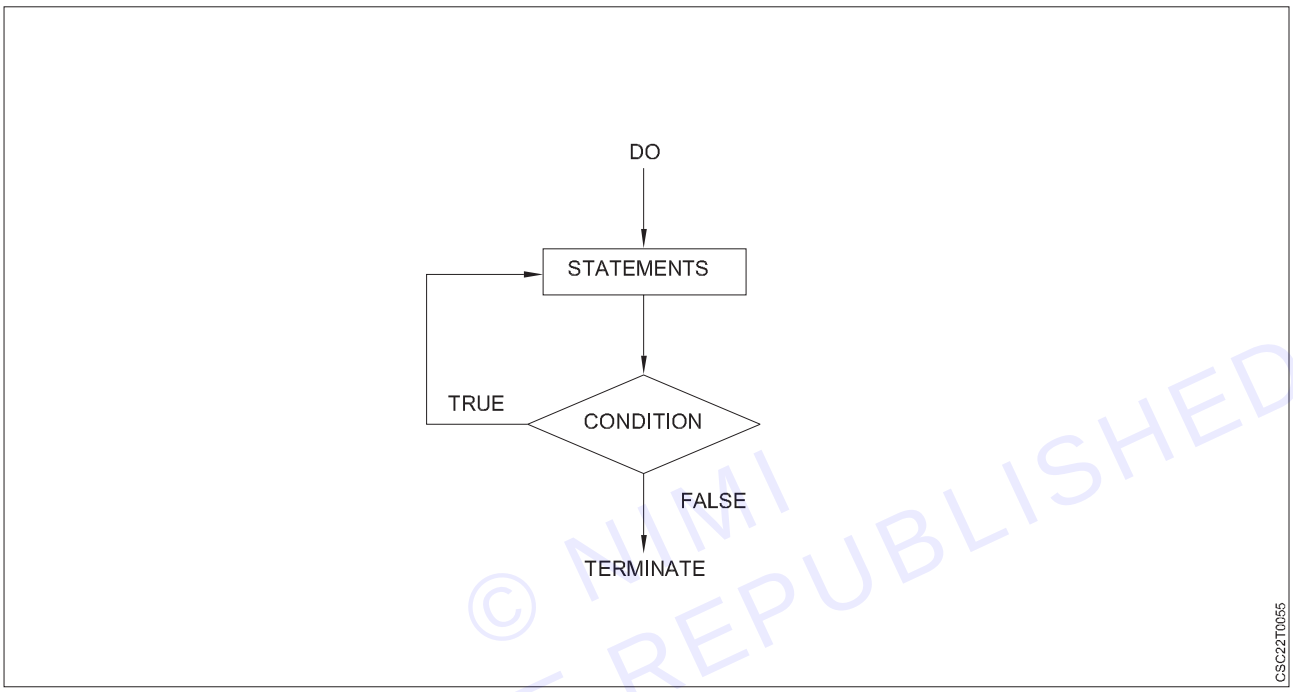
Java do-while loop

The do-while loop checks the condition at the end of the loop after executing the loop statements. When the number of iteration is not known and we have to execute the loop at least once, we can use do-while loop.

It is also known as the exit-controlled loop since the condition is not checked in advance. The syntax of the do-while loop is given below.

```
do
{
//statements
} while (condition);
```

The flow chart of the do-while loop is given in the following image.



Consider the following example to understand the functioning of the do-while loop in Java.

Calculation.java

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
int i = 0;
System.out.println("Printing the list of first 10 even numbers \n");
do {
System.out.println(i);
i = i + 2;
}while(i<=10);
}
}
```

Output:

```
Printing the list of first 10 even numbers
0
2
4
```

6

8

10

Jump Statements

Jump statements are used to transfer the control of the program to the specific statements. In other words, jump statements transfer the execution control to the other part of the program. There are two types of jump statements in Java, i.e., break and continue.

Java break statement

As the name suggests, the break statement is used to break the current flow of the program and transfer the control to the next statement outside a loop or switch statement. However, it breaks only the inner loop in the case of the nested loop.

The break statement cannot be used independently in the Java program, i.e., it can only be written inside the loop or switch statement.

The break statement example with for loop

Consider the following example in which we have used the break statement with the for loop.

BreakExample.java

```
public class BreakExample {
 public static void main(String[] args) {
// TODO Auto-generated method stub
for(int i = 0; i<= 10; i++) {
System.out.println(i);
if(i==6) {
break;
}
}
}
}
```

Output:

0

1

2

3

4

5

6

break statement example with labeled for loop

Calculation.java

```
public class Calculation {
public static void main(String[] args) {
// TODO Auto-generated method stub
a:
```

```
for(int i = 0; i<= 10; i++) {
b:
for(int j = 0; j<=15;j++) {
c:
for (int k = 0; k<=20; k++) {
System.out.println(k);
if(k==5) {
break a;
}
}
}
}
}
}
```

Output:

0
1
2
3
4
5

Java continue statement

Unlike break statement, the continue statement doesn't break the loop, whereas, it skips the specific part of the loop and jumps to the next iteration of the loop immediately.

Consider the following example to understand the functioning of the continue statement in Java.

```
public class ContinueExample {
public static void main(String[] args) {
// TODO Auto-generated method stub
for(int i = 0; i<= 2; i++) {
for (int j = i; j<=5; j++) {
if(j == 4) {
continue;
}
System.out.println(j);
}
}
}
}
```

Output:

0

```
1
2
3
5
1
2
3
5
2
3
5
```

Terminate a Java program

What is exit() Method in Java?

Java provides exit() method to exit the program at any point by terminating running JVM, based on some condition or programming logic. In Java exit() method is in java.lang.System class. This System.exit() method terminates the current JVM running on the system which results in termination of code being executed currently. This method takes status code as an argument.

Uses

- exit() method is required when there is an abnormal condition and we need to terminate the program immediately.
- exit() method can be used instead of throwing an exception and providing the user a script that mentions what caused the program to exit abruptly.
- Another use-case for exit() method in java, is if a programmer wants to terminate the execution of the program in case of wrong inputs.

Syntax of exit() in Java

public static void exit(int status)

Parameters of exit() in jJava

System.exit() method takes status as a parameter, these status codes tell about the status of termination. This status code is an integer value and its meanings are as follows:

- exit(0) - Indicates successful termination
- exit(1) - Indicates unsuccessful termination
- exit(-1) - Indicates unsuccessful termination with Exception

Note: Any non-zero value as status code indicates unsuccessful termination.

Return Value for exit() in Java

As the syntax suggests, it is a void method that does not return any value.

Exit a Java Method using System.exit(0)

As mentioned earlier, System.exit(0) method terminates JVM which results in termination of the currently running program too. Status is the single parameter that the method takes. If the status is 0, it indicates the termination is successful.

Let's see practical implementations of System.exit(0) method in Java.

Example 1

Here is a simple program that uses System.exit(0) method to terminate the program based on the condition.

```
public class SampleExitMethod {
  public static void exampleMethod(int[] array1) {
    for (int i = 0; i < array1.length; i++) {
      //Check condition for i
      if (i > 4) {
        System.out.println("Terminating JVM...");
     //Terminates JVM when if condition is satisfied
        System.exit(0);
      }
      System.out.println("Array Index: " + i + " Array Element: " + array1[i]);
    }
  }
  public static void main(String[] args) {
    int[] array1 = { 0, 2, 4, 6, 8, 10, 12, 14, 16 };
    //function call
    exampleMethod(array1);
  }
}
```

Output:

```
Array Index: 0 Array Element: 0
Array Index: 1 Array Element: 2
Array Index: 2 Array Element: 4
Array Index: 3 Array Element: 6
Array Index: 4 Array Element: 8
Terminating JVM...
```

Explanation:

In the above program, an array is being accessed till its index is 4. Until if condition is false i.e., array index <= 4, JVM will give Array index and element stored at that index as the output.

Once the if condition is true i.e., array index > 4, JVM will execute statements inside the if condition. Here it will first execute the print statement and when exit(0) method is called it will terminate JVM and program execution.

Example 2

Let's see one more example of exit() method in a try-catch-finally block.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class check {
public static void main(String[] args) {
try {
       //Reads "file.txt" file
      BufferedReader br = new BufferedReader(new FileReader("file.txt"));
// Closes bufferreader br
      System.out.println(br.readLine());
      br.close();
   } // Catches exception
    catch (IOException e) {
      System.out.println("Exception caught. Terminating JVM.");
      System.exit(0);
```

```
    } // Terminates JVM as file is not available
    finally {
      System.out.println("Exiting the program");
    }
  }
}
```

Output:

Exception caught. Terminating JVM.

Explanation:

The piece of code above is trying to read a file and print a line from it if the file exists. If a file doesn't exist, the exception is raised and execution will go in catch block and code exits with System.exit(0) method in the catch block.

- We know, finally block is executed every time irrespective of try-catch blocks are being executed or not. But, in this case, the file is not present so JVM will check the catch block where System.exit(0) method is used that terminates JVM and currently running the program.
- As the JVM is terminated in the catch block, JVM won't be able to read the final block hence, the final block will not be executed.

Exit a Java Method using Return

exit() method in java is the simplest way to terminate the program in abnormal conditions. There is one more way to exit the java program using return keyword. return keyword completes execution of the method when used and returns the value from the function. The return keyword can be used to exit any method when it doesn't return any value.

Let's see this using an example:

```
public class SampleExitMethod2 {
public static void SubMethod(int num1, int num2) {
    if (num2 > num1) return;

    int answer = num1 - num2;
    System.out.println(answer);
  }
public static void main(String[] args) {
    SubMethod(2, 5); // if condition is true
    SubMethod(3, 2);
    SubMethod(100, 20);
   SubMethod(102, 110); // if condition is true
  }
}
```

Output:

1

80

Explanation:

In the above program, Sub Method takes two arguments num1 and num2, subtracts num2 from num1 giving the result in the answer variable. But it has one condition that states num2 should not be greater than num1, this condition is implemented using if conditional block.

If this condition is true, it will execute if block which has a return statement. In this case, SubMethod is of void type hence, return doesn't return any value and in this way exits the function.

Does goto Exist in Java?

No, goto does not exist in Java. Because Java has reserved it for now and may use it in a later version.

In other languages, goto is a control statement used to transfer control to a certain point in the programming. After goto statement is executed program starts executing the lines where goto has transferred its control and then other lines are executed. goto statement works with labels that are identifiers, these labels are used to the state where goto has to transfer the control of execution.

Java does not support goto() method but it supports label. No support for goto method is due to the following reasons:

1 Goto-ridden code hard to understand and hard to maintain

2 These labels can be used with nested loops. A combination of break, continue and labels can be used to achieve the functionality of goto() method in Java.

3 Prohibits compiler optimization

Example:

```
// Java code
public class Main {
 public static void main(String[] args) {
    boolean t = true;
    first:{
      second:{
        third:{
          System.out.println(“Before the break statement”);
          if (
            t
          ) break second; // break out of second block
        }
        System.out.println(“No execution for this statement”);
      }
      System.out.println(“Part of first block, outside second block.”);
    }
  }
}
```

Output:

Before the break statement

Part of first block, outside second block.

In the above code, the outer label is created for the first for loop. When if the condition is true break outer; statement will go back to first for loop and break the execution. Use this Compiler to compile your Java code.

Conclusion

- exit() method is used to terminate JVM.
- Status code other than 0 in exit() method indicates abnormal termination of code.
- goto does not exist in Java, but it supports labels.
- It is better to use exception handling or plain return statements to exit a program while execution.
- System.exit() method suit better for script-based applications or wherever the status codes are interpreted.

Java Numbers

Numbers

Primitive number types are divided into two groups:

Integer types stores whole numbers, positive or negative (such as 123 or -456), without decimalsValid types are byte, short, int and long. Which type you should use, depends on the numeric value.

Floating point types represents numbers with a fractional part, containing one or more decimals. There are two types: float and double

Integer Types

Byte

The byte data type can store whole numbers from -128 to 127. This can be used instead of int or other integer types to save memory when you are certain that the value will be within -128 and 127:

Example

```
byte myNum = 100;
System.out.println(myNum);
```

Short

The short data type can store whole numbers from -32768 to 32767:

Example

```
short myNum = 5000;
System.out.println(myNum);
```

Int

The int data type can store whole numbers from -2147483648 to 2147483647. In general, and in our tutorial, the int data type is the preferred data type when we create variables with a numeric value.

Example

```
int myNum = 100000;
System.out.println(myNum);
```

Long

The long data type can store whole numbers from -9223372036854775808 to 9223372036854775807. This is used when int is not large enough to store the value. Note that you should end the value with an “L”:

Example

```
long myNum = 15000000000L;
System.out.println(myNum);
```

Floating Point Types

You should use a floating point type whenever you need a number with a decimal, such as 9.99 or 3.14515.

The float and double data types can store fractional numbers. Note that you should end the value with an “f” for floats and “d” for doubles:

Example

```
float myNum = 5.75f;
System.out.println(myNum);
```

Double Example

Example

```
double myNum = 19.99d;
System.out.println(myNum);
```

Java String

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string

For example:

```
1   char[] ch={'j','a','v','a','t','p','o','i','n','t'};
```

2 String s=new String(ch);

Java String cla s provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

The java.lang.String class implements Serializable, Comparable and CharSequence interfaces.

What is String in Java?

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The java.lang.String class is used to create a string object.

How to create a string object?

There are two ways to create String object:

1 By string literal

2 By new keyword

1 String Literal

Java String literal is created by using double quotes.

For example:

1 String s="welcome";

Why Java uses the concept of String literal?

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

2 By new keyword

1 String s=new String("Welcome");//creates two objects and one reference variable

In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

Java String Example

StringExample.java

```
public class StringExample{
public static void main(String args[]){
String s1="java";//creating string by Java string literal
char ch[]={'s','t','r','i','n','g','s'};
String s2=new String(ch);//converting char array to string
String s3=new String("example");//creating Java string by new keyword
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

Output:

```
java
```

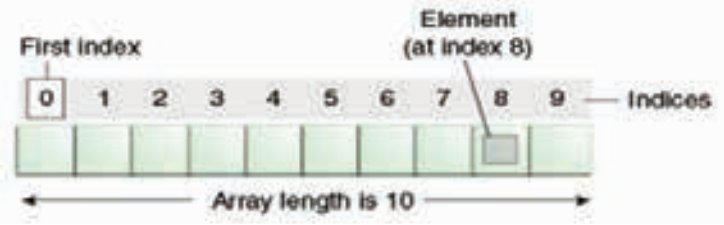
JAVA Arrays

Normally, an array is a collection of similar type of elements which has contiguous memory location

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

Unlike C/C++, we can get the length of the array using the length member. In C/C++, we need to use the sizeof operator.

In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java. Like C/C++, we can also create single dimensional or multidimensional arrays in Java. Moreover, Java provides the feature of anonymous arrays which is not available in C/C++.



Advantages

Code Optimization: It makes the code optimized, we can retrieve or sort the data efficiently.

Random access: We can get any data located at an index position

Disadvantages

Size Limit: We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

Types of array:

Single Dimensional Array

Multidimensional Array

Syntax to Declare an Array in Java

dataType[] arr; (or)

dataType []arr; (or)

dataType arr[];

Instantiation of an Array in Java

arrayRefVar=new datatype[size];

Example of Java Array

Let's see the simple example of java array, where we are going to declare, instantiate, initialize and traverse an array.

Java Program to illustrate how to declare, instantiate, initialize

and traverse the Java array.

```
class Testarray{
public static void main(String args[]){
int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
```

```
a[3]=40;
a[4]=50;
//traversing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

Output

```
10
20
70
40
50
```

We can declare, instantiate and initialize the java array together by:

```
int a[]={33,3,4,5};//declaration, instantiation and initialization
Java Program to illustrate the use of declaration, instantiation
and initialization of Java array in a single line
class Testarray1{
public static void main(String args[]){
int a[]={33,3,4,5};//declaration, instantiation and initialization
//printing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);
}}
```

Output

```
33
3
4
5
```

For-each Loop for Java Array

We can also print the Java array using for-each loop. The Java for-each loop prints the array elements one by one. It holds an array element in a variable, then executes the body of the loop.

The syntax of the for-each loop is given below:

```
for(data_type variable:array){
//body of the loop
}
Java Program to print the array elements using for-each loop
class Testarray1{
public static void main(String args[]){
int arr[]={33,3,4,5};
//printing array using for-each loop
for(int i:arr)
System.out.println(i);
}}
```

Passing Array to a Method in Java
We can pass the java array to method so that we can reuse the same logic on any array.

Let's see the simple example to get the minimum number of an array using a method.

```
//Java Program to demonstrate the way of passing an array
//to method.
class Testarray2{
//creating a method which receives an array as a parameter
static void min(int arr[]){
int min=arr[0];
for(int i=1;i<arr.length;i++)
 if(min>arr[i])
  min=arr[i];

System.out.println(min);
}

public static void main(String args[]){
int a[]={33,3,4,5};//declaring and initializing an array
min(a);//passing array to method
}}
```

Output
3

Multidimensional Array in Java
In such case, data is stored in row and column based index (also known as matrix form).

Syntax to Declare Multidimensional Array in Java

```
dataType[][] arrayRefVar; (or)
dataType [][]arrayRefVar; (or)
dataType arrayRefVar[][]; (or)
dataType []arrayRefVar[];
```

Example to instantiate Multidimensional Array in Java

```
int[][] arr=new int[3][3];//3 row and 3 column
```

Example to initialize Multidimensional Array in Java

```
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
arr[1][0]=4;
arr[1][1]=5;
arr[1][2]=6;
arr[2][0]=7;
```

```
arr[2][1]=8;
arr[2][2]=9;
```

Example of Multidimensional Java Array

Let's see the simple example to declare, instantiate, initialize and print the 2Dimensional array.

```
//Java Program to illustrate the use of multidimensional array
class Testarray3{
public static void main(String args[]){
//declaring and initializing 2D array
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
//printing 2D array
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
   System.out.print(arr[i][j]+" ");
 }
 System.out.println();
}
}}
```

Output

```
1 2 3
2 4 5
4 4 5
```

Addition of 2 Matrices in Java

Let's see a simple example that adds two matrices.

```
//Java Program to demonstrate the addition of two matrices in Java
class Testarray5{
public static void main(String args[]){
//creating two matrices
int a[][]={{1,3,4},{3,4,5}};
int b[][]={{1,3,4},{3,4,5}};

//creating another matrix to store the sum of two matrices
int c[][]=new int[2][3];

//adding and printing addition of 2 matrices
for(int i=0;i<2;i++){
for(int j=0;j<3;j++){
c[i][j]=a[i][j]+b[i][j];
System.out.print(c[i][j]+" ");
}
System.out.println();//new line
}
```

```
}}
```

Output

```
2 6 8
6 8 10
```

Multiplication of 2 Matrices in Java

In the case of matrix multiplication, a one-row element of the first matrix is multiplied by all the columns of the second matrix which can be understood by the image given below.

Matrix Multiplication in Java

Let's see a simple example to multiply two matrices of 3 rows and 3 columns.

```
//Java Program to multiply two matrices
public class MatrixMultiplicationExample{
public static void main(String args[]){
//creating two matrices
int a[][]={{1,1,1},{2,2,2},{3,3,3}};
int b[][]={{1,1,1},{2,2,2},{3,3,3}};

//creating another matrix to store the multiplication of two matrices
int c[][]=new int[3][3];  //3 rows and 3 columns

//multiplying and printing multiplication of 2 matrices
for(int i=0;i<3;i++){
for(int j=0;j<3;j++){
c[i][j]=0;
for(int k=0;k<3;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}//end of k loop
System.out.print(c[i][j]+" ");  //printing matrix element
}//end of j loop
System.out.println();//new line
}
}}
```

Output

```
6 6 6
12 12 12
18 18 18
```

LESSON 94 - 100: JAVA Classes, Overloading and Inheritance

Java Classes/Objects

Classes and objects are the two main aspects of object-oriented programming.

Java is an object-oriented programming language.

Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.

A Class is like an object constructor, or a “blueprint” for creating objects

Create a Class

To create a class, use the keyword class:

Main.java

Create a class named “Main” with a variable x:

```
public class Main {
  int x = 5;
}
```

Create an Object

In Java, an object is created from a class. We have already created the class named Main, so now we can use this to create objects.

To create an object of Main, specify the class name, followed by the object name, and use the keyword new:

Example

Create an object called “myObj” and print the value of x:

```
public class Main {
  int x = 5;
 public static void main(String[] args) {
   Main myObj = new Main();
   System.out.println(myObj.x);
  }
  }
```

Java Methods

A method is a block of code which only runs when it is called.

You can pass data, known as parameters, into a method.

Methods are used to perform certain actions, and they are also known as functions.

Why use methods? To reuse code: define the code once, and use it many times.

Create a Method

A method must be declared within a class. It is defined with the name of the method, followed by parentheses (). Java provides some pre-defined methods, such as System.out.println(), but you can also create your own methods to perform certain actions:

Example

Create a method inside Main:

```
public class Main {
  static void myMethod() {
    // code to be executed
  }
}
```

Example Explained

- My Method() is the name of the method
- Static means that the method belongs to the Main class and not an object of the Main class. You will learn more about objects and how to access methods through objects later in this tutorial.
- Void means that this method does not have a return value.

Call a Method

To call a method in Java, write the method's name followed by two parentheses () and a semicolon;

In the following example, myMethod() is used to print a text (the action), when it is called:

Example

```
public class Main {
  static void myMethod() {
    System.out.println("I just got executed!");
  }
  public static void main(String[] args) {
    myMethod();
  }
}
// Outputs "I just got executed!"
```


Java Object as Parameter

Objects, like primitive types, can be passed as parameters to methods in Java. When passing an object as a parameter to a method, a reference to the object is passed rather than a copy of the object itself. This means that any modifications made to the object within the method will have an impact on the original object.

```
public class MyClass {
    // Fields or attributes
    private int attribute1;
    private String attribute2;
    private double attribute3;
    // Constructor
    public MyClass(int attribute1, String attribute2, double attribute3) {
        this.attribute1 = attribute1;
        this.attribute2 = attribute2;
        this.attribute3 = attribute3;
```

```
    }
    // Method with object as parameter
    public void myMethod(MyClass obj) {
        // block of code to define this method
    }
    // More methods
}
```

Java Method Overloading

In Java, two or more methods may have the same name if they differ in parameters (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called method overloading.

Why method overloading?

Suppose, you have to perform the addition of given numbers but there can be any number of arguments (let's say either 2 or 3 arguments for simplicity).

In order to accomplish the task, you can create two methods sum2num(int, int) and sum3num(int, int, int) for two and three parameters respectively. However, other programmers, as well as you in the future may get confused as the behavior of both methods are the same but they differ by name.

The better way to accomplish this task is by overloading methods. And, depending upon the argument passed, one of the overloaded methods is called. This helps to increase the readability of the program

How to perform method overloading in Java?

Here are different ways to perform method overloading:

```
class MethodOverloading {
    private static void display(int a){
        System.out.println("Arguments: " + a);
    }
    private static void display(int a, int b){
        System.out.println("Arguments: " + a + " and " + b);
    }
    public static void main(String[] args) {
        display(1);
        display(1, 4);
    }
}
```

Output:

Arguments: 1

Arguments: 1 and 4

Important Points

- Two or more methods can have the same name inside the same class if they accept different arguments. This feature is known as method overloading.
- Method overloading is achieved by either:

- changing the number of arguments.
- or changing the data type of arguments.

- It is not method overloading if we only change the return type of methods. There must be differences in the number of parameters.

Constructors and Overloaded constructors

Java Constructors

A constructor in Java is similar to a method that is invoked when an object of the class is created.

Unlike Java methods, a constructor has the same name as that of the class and does not have any return type. For example,

```
class Test {
  Test() {
    // constructor body
  }
}
```

Example: Java Constructor

```
class Main {
  private String name;

  // constructor
  Main() {
    System.out.println("Constructor Called:");
    name = "Programiz";
  }
  public static void main(String[] args) {
    // constructor is invoked while
    // creating an object of the Main class
    Main obj = new Main();
    System.out.println("The name is " + obj.name);
  }
}
```

Output:

Constructor Called:

The name is Programiz

Types of Constructor

In Java, constructors can be divided into three types:

1 No-Arg Constructor

2 Parameterized Constructor

3 Default Constructor

1 Java No-Arg Constructors

Similar to methods, a Java constructor may or may not have any parameters (arguments).

If a constructor does not accept any parameters, it is known as a no-argument constructor.

2 Java Parameterized Constructor

A Java constructor can also accept one or more parameters. Such constructors are known as parameterized constructors (constructors with parameters).

3 Java Default Constructor

If we do not create any constructor, the Java compiler automatically creates a no-arg constructor during the execution of the program.

Important Notes on Java Constructors

- Constructors are invoked implicitly when you instantiate objects.
- The two rules for creating a constructor are:
  1 The name of the constructor should be the same as the class.
  2 A Java constructor must not have a return type.
- If a class doesn't have a constructor, the Java compiler automatically creates a default constructor during run-time. The default constructor initializes instance variables with default values. For example, the int variable will be initialized to 0
- **Constructor types:**

  No-Arg Constructor - a constructor that does not accept any arguments

  Parameterized constructor - a constructor that accepts arguments

  Default Constructor - a constructor that is automatically created by the Java compiler if it is not explicitly defined.
- A constructor cannot be abstract or static or final.
- A constructor can be overloaded but can not be overridden.

Constructors Overloading in Java

Similar to Java method overloading, we can also create two or more constructors with different parameters. This is called constructor overloading.

Example: Java Constructor Overloading

```
class Main {
 String language;
 // constructor with no parameter
 Main() {
  this.language = “Java”;
 }
 // constructor with a single parameter
 Main(String language) {
  this.language = language;
 }
 public void getName() {
  System.out.println(“Programming Language: “ + this.language);
 }
```

```
  public static void main(String[] args) {
    // call constructor with no parameter
    Main obj1 = new Main();
    // call constructor with a single parameter
    Main obj2 = new Main("Python");
    obj1.getName();
    obj2.getName();
  }
}
```

Output

Programming Language: Java

Programming Language: Python

Inheritance in JAVA

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

Why use inheritance in java

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

Terms used in Inheritance

- **Class**

  A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

- **Sub Class/Child Class**

  Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

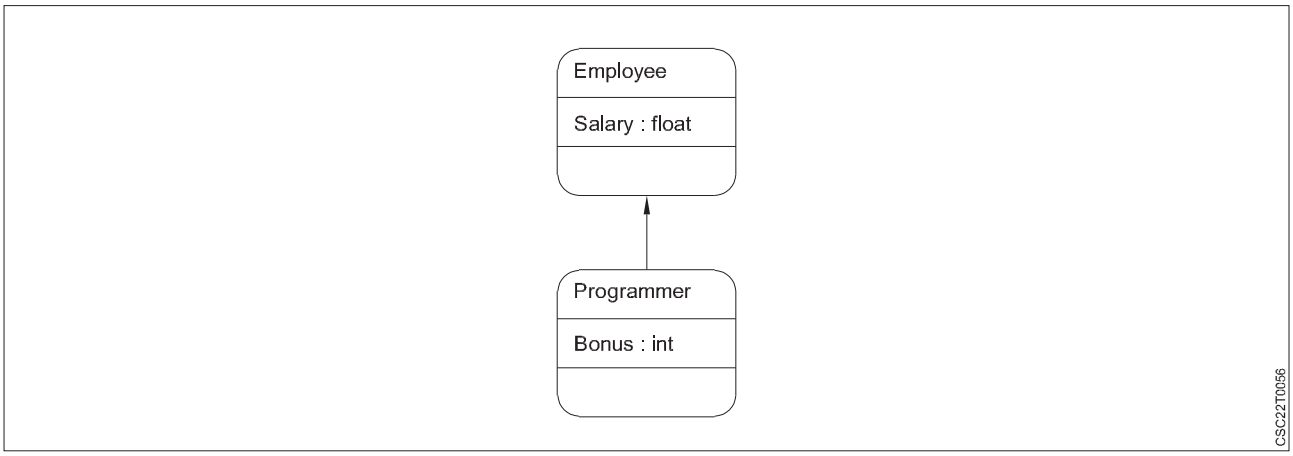
- **Super Class/Parent Class**

  Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

- **Reusability**

  As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

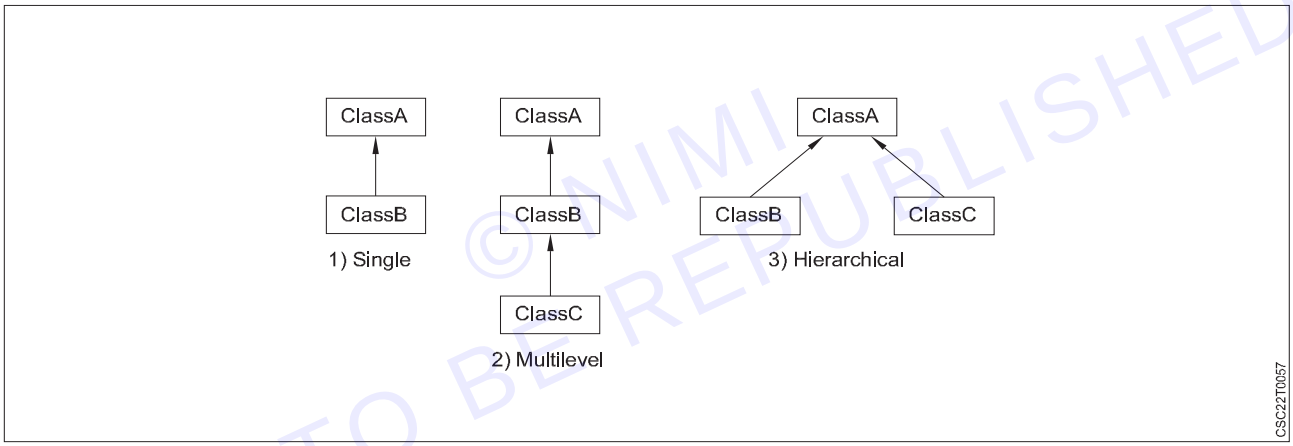
Java Inheritance Example



Types of inheritance in java

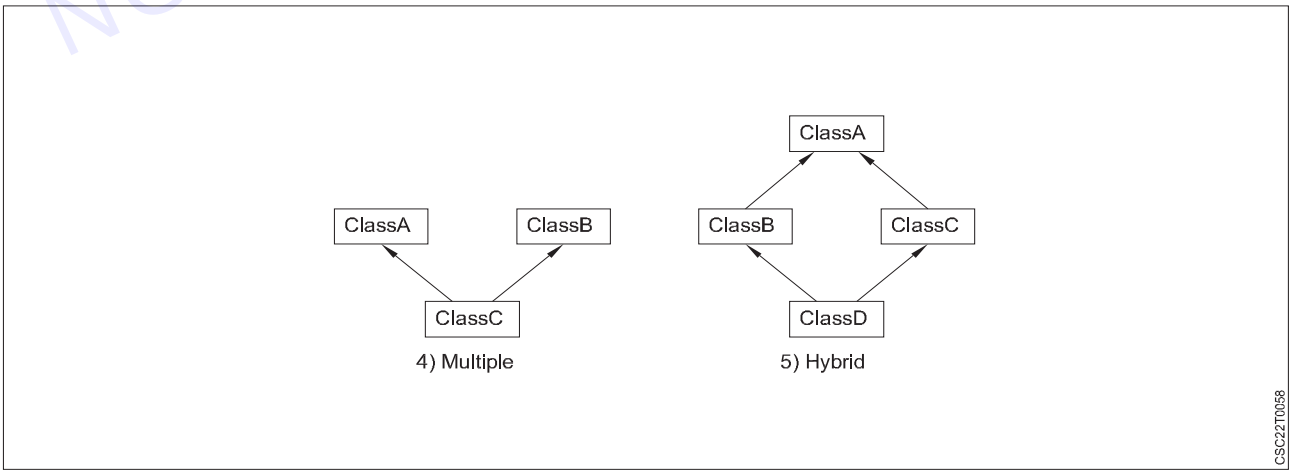
On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



When one class inherits multiple classes, it is known as multiple inheritance. For Example:

ADVERTISEMENT



Single Inheritance Example

When a class inherits another class, it is known as a single inheritance. In the example given below, Dog class inherits the Animal class, so there is the single inheritance.

File: TestInheritance.java

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Output:

barking...

eating...

Multilevel Inheritance Example

When there is a chain of inheritance, it is known as multilevel inheritance. As you can see in the example given below, BabyDog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.

File: TestInheritance2.java

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class BabyDog extends Dog{
void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

Output:

weeping...

barking...

eating...

Hierarchical Inheritance Example

When two or more classes inherits a single class, it is known as hierarchical inheritance. In the example given below, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.

File: TestInheritance3.java

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

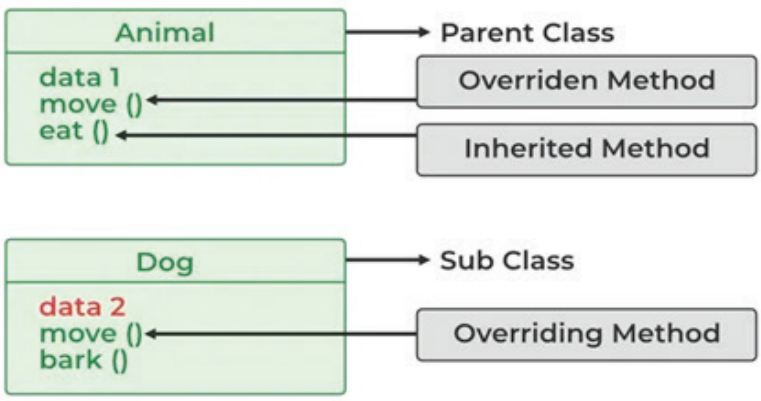
Output:

meowing...

eating...

Method Overriding in JAVA

In Java, Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes. When a method in a subclass has the same name, the same parameters or signature, and the same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.



Method overriding is one of the ways by which Java achieves Run Time Polymorphism. The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed. In other words, it is the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed.

Example of Method Overriding in Java

Java program to demonstrate method overriding in java

```
// Base Class
class Parent {
    void show() { System.out.println("Parent's show()"); }
}
// Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
    Override void show()
    {
        System.out.println("Child's show()");
    }
}
// Driver class
class Main {
    public static void main(String[] args)
    {
        // If a Parent type reference refers
        // to a Parent object, then Parent's
        // show is called
        Parent obj1 = new Parent();
        obj1.show();
        // If a Parent type reference refers  to a Child object Child's show() is called. This is called RUN TIME POLYMORPHISM.
        Parent obj2 = new Child();
        obj2.show();
    }
}
```

1 Rules for Java Method Overriding

The access modifier for an overriding method can allow more, but not less, access than the overridden method. For example, a protected instance method in the superclass can be made public, but not private, in the subclass. Doing so will generate a compile-time error.

A Simple Java program to demonstrate Overriding and Access-Modifiers

```
class Parent {
    // private methods are not overridden
```

```
        private void m1()
        {
                System.out.println("From parent m1()");
        }
        protected void m2()
        {
                System.out.println("From parent m2()");
        }
}
class Child extends Parent {
        // new m1() method
        // unique to Child class
        private void m1()
        {
                System.out.println("From child m1()");
        }
        // overriding method
        // with more accessibility
        @Override public void m2()
        {
                System.out.println("From child m2()");
        }
}
// Driver class
class Main {
        public static void main(String[] args)
        {
                Parent obj1 = new Parent();
                obj1.m2();
                Parent obj2 = new Child();
                obj2.m2();
        }
}
```

2 Final methods can not be overridden

If we don't want a method to be overridden, we declare it as final. Please see Using Final with Inheritance.
A Java program to demonstrate final methods which cannot be overridden

```
class Parent {
        // Can't be overridden
        final void show() {}
}
```

```
class Child extends Parent {
    // This would produce error
    void show() {}
}
```

3 Static methods can not be overridden (Method Overriding vs Method Hiding)

When you define a static method with the same signature as a static method in the base class, it is known as method hiding. The following table summarizes what happens when you define a method with the same signature as a method in a super-class.

	Superclass Instance Method	Superclass Static Method
Subclass Instance Method	Overrides	Generates a compile-time error
Subclass Static Method	Generates a compile-time error	Hides

Java program to show that if the static method is redefined by a derived class, then it is not overriding, it is hiding.

```
class Parent {
    // Static method in base class
    // which will be hidden in subclass
    static void m1()
    {
        System.out.println("From parent "
                                    + "static m1()");
    }
    // Non-static method which will
    // be overridden in derived class
    void m2()
    {
        System.out.println(
            "From parent "
            + "non - static(instance) m2() ");
    }
}
class Child extends Parent {
    // This method hides m1() in Parent
    static void m1()
    {
        System.out.println("From child static m1()");
    }
    // This method overrides m2() in Parent
    @Override public void m2()
```

```
    {
        System.out.println(
            "From child "
            + "non - static(instance) m2() ");
    }
}
// Driver class
class Main {
    public static void main(String[] args)
    {
        Parent obj1 = new Child();
        // As per overriding rules this
        // should call to class Child static
        // overridden method. Since static
        // method can not be overridden, it
        // calls Parent's m1()
        obj1.m1();
        // Here overriding works
        // and Child's m2() is called
        obj1.m2();
    }
}
```

4 Private methods can not be overridden

Private methods cannot be overridden as they are bonded during compile time. Therefore we can't even override private methods in a subclass.

```
class SuperClass {
    private void privateMethod()
    {
        System.out.println(
            "This is a private method in SuperClass");
    }
    public void publicMethod()
    {
        System.out.println(
            "This is a public method in SuperClass");
        privateMethod();
    }
}
class SubClass extends SuperClass {
    // This is a new method with the same name as the
```

```
        // private method in SuperClass
        private void privateMethod()
        {
                System.out.println(
                        "This is a private method in SubClass");
        }
        // This method overrides the public method in SuperClass
        public void publicMethod()
        {
                System.out.println(
                        "This is a public method in SubClass");
                privateMethod(); // calls the private method in
                                                // SubClass, not SuperClass
        }
}
public class Test {
        public static void main(String[] args)
        {
                SuperClass obj1 = new SuperClass();
                obj1.publicMethod(); // calls the public method in
                                                        // SuperClass
                SubClass obj2 = new SubClass();
                obj2.publicMethod(); // calls the overridden public
                                                        // method in SubClass
        }
}
```

5 The overriding method must have the same return type (or subtype)

From Java 5.0 onwards it is possible to have different return types for an overriding method in the child class, but the child's return type should be a sub-type of the parent's return type. This phenomenon is known as the covariant return type.

```
class SuperClass {
        public Object method()
        {
                System.out.println(
                        "This is the method in SuperClass");
                return new Object();
        }
}
class SubClass extends SuperClass {
        public String method()
```

```
    {
        System.out.println(
            "This is the method in SubClass");
        return "Hello, World!";
    }
}
public class Test {
    public static void main(String[] args)
    {
        SuperClass obj1 = new SuperClass();
        obj1.method();
        SubClass obj2 = new SubClass();
        obj2.method();
    }
}
```

6 Invoking overridden method from sub-class

We can call the parent class method in the overriding method using the super keyword.

A Java program to demonstrate that overridden method can be called from sub-class

```
class Parent {
    void show() { System.out.println("Parent's show()"); }
}
// Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
    @Override void show()
    {
        super.show();
        System.out.println("Child's show()");
    }
}
// Driver class
class Main {
    public static void main(String[] args)
    {
        Parent obj = new Child();
        obj.show();
    }
}
```

LESSON 101 - 108 : Multithreading and Exception Handling in Java

In Java programming, threads enable concurrent execution and multitasking within an application. Understanding the life cycle of threads in Java and the various states is essential for efficient and synchronized thread management.

Importance of understanding the life cycle of Thread in Java and its states

The thread life cycle in Java is an important concept in multithreaded applications. Let's see the importance of understanding life cycle of thread in java:

1 Understanding the life cycle of a thread in Java and the states of a thread is essential because it helps identify potential issues that can arise when creating or manipulating threads.

2 It allows developers to utilize resources more effectively and prevent errors related to multiple threads accessing shared data simultaneously.

3 Knowing the thread states in Java helps predict a program's behaviour and debug any issues that may arise.

4 It also guides the developer on properly suspending, resuming, and stopping a thread as required for a specific task.

The Life Cycle of Thread in Java - Threads State

In Java, the life cycle of Thread goes through various states. These states represent different stages of execution. Here are examples of each stage of the life cycle of Thread in Java with real-life use cases:

- **New (born) state**
  - **Example:** Creating a new thread using the Thread class constructor.
  - **Use case:** Creating a new thread to perform a background task while the main Thread continues with other operations
- **Runnable state**
  - **Example:** After calling the start() method on a thread, it enters the runnable state.
  - **Use case:** Multiple threads competing for CPU time to perform their tasks concurrently.
- **Running state**
  - **Example:** When a thread executes its code inside the run() method.
  - **Use case:** A thread executing a complex computation or performing a time-consuming task.
- **Blocked state**
  - **Example:** When a thread tries to access a synchronized block or method, but another thread already holds the lock.
  - **Use case:** Multiple threads accessing a shared resource can only be obtained by a single Thread, such as a database or a file.

- **Waiting state**
  - **Example:** Using the wait method inside a synchronized block, a thread can wait until another thread calls the notify() or notifyAll() methods to wake it up.
  - **Use case:** Implementing the producer-consumer pattern, where a thread waits for a specific condition to be met before continuing its execution
- **Timed waiting state**
  - **Example:** Using methods like sleep(milliseconds) or join(milliseconds) causes a thread to enter the timed waiting state for the specified duration.
  - **Use case:** Adding delays between consecutive actions or waiting for the completion of other threads before proceeding.
- **Terminated state**
  - **Example:** When the run() method finishes its execution or when the stop() method is called on the Thread.
  - **Use case:** Completing a task or explicitly stopping a thread's execution.

Example: Thread Life Cycle in Java:

Here's an example that demonstrates the life cycle of Thread in Java:

```
public class ThreadLifecycleExample {
    public static void main(String[] args) {
        Thread thread = new Thread(() -> {
            System.out.println("Thread is running.");
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("Thread is terminating.");
        });
        System.out.println("Thread is in the New state.");
        thread.start();
        System.out.println("Thread is in the Runnable state.");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Thread is in the Timed Waiting state.");
        try {
            thread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Thread is in the Terminated");
```

Handling Thread Exceptions and Terminating Threads

When dealing with exceptions in life cycle of a thread java, there are several ways to handle them.

- The most common way is wrapping the code in a try-catch block.
- Moreover, when a thread is no longer needed, it should be terminated for the application to avoid memory leaks and other issues. To do this, invoke the Thread's interrupt() or stop() methods. The former method sends an interruption signal to the Thread, while the latter stops it immediately and can cause instability in some cases.

Java Threads | How to create a thread in Java

There are two ways to create a thread:

1 By extending Thread class

2 By implementing Runnable interface.

Thread class

Thread class provide constructors and methods to create and perform operations on a thread.Thread class extends Object class and implements Runnable interface.

Commonly used Constructors of Thread class

- Thread()
- Thread(String name)
- Thread(Runnable r)
- Thread(Runnable r,String name)

Commonly used methods of Thread class

1 **public void run()**: is used to perform action for a thread.

2 **public void start():** starts the execution of the thread.JVM calls the run() method on the thread.

3 **public void sleep(long miliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.

4 **public void join():** waits for a thread to die.

5 **public void join(long miliseconds):** waits for a thread to die for the specified miliseconds.

6 **public int getPriority():** returns the priority of the thread.

7 **public int setPriority(int priority):** changes the priority of the thread.

8 **public String getName():** returns the name of the thread.

9 **public void setName(String name):** changes the name of the thread.

10 **public Thread currentThread():** returns the reference of currently executing thread.

11 **public int getId():** returns the id of the thread.

12 **public Thread.State getState():** returns the state of the thread.

13 **public boolean isAlive():** tests if the thread is alive.

14 **public void yield():** causes the currently executing thread object to temporarily pause and allow other threads to execute.

15 **public void suspend():** is used to suspend the thread(deprecated).

16 **public void resume():** is used to resume the suspended thread(deprecated).

17 **public void stop():** is used to stop the thread(deprecated).

18 **public boolean isDaemon()**: tests if the thread is a daemon thread.

19 **public void setDaemon(boolean b):** marks the thread as daemon or user thread.

20 public void interrupt(): interrupts the thread.

21 public boolean isInterrupted(): tests if the thread has been interrupted.

22 public static boolean interrupted(): tests if the current thread has been interrupted.

Runnable interface

The Runnable interface should be implemented by any class whose instances are intended to be executed by a thread. Runnable interface have only one method named run().

1 public void run(): is used to perform action for a thread.

Starting a thread

The **start() method** of Thread class is used to start a newly created thread. It performs the following tasks:

- A new thread starts(with new callstack).
- The thread moves from New state to the Runnable state.
- When the thread gets a chance to execute, its target run() method will run.

1 Java Thread Example by extending Thread class

FileName: Multi.java

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
}
}
```

Output:

```
thread is running...
```

2 Java Thread Example by implementing Runnable interface

FileName: Multi3.java

```
class Multi3 implements Runnable{
public void run(){
System.out.println("thread is running...");
}

public static void main(String args[]){
Multi3 m1=new Multi3();
Thread t1 =new Thread(m1);   // Using the constructor Thread(Runnable r)
t1.start();
}
    }
```

Output:

```
thread is running...
```

If you are not extending the Thread class, your class object would not be treated as a thread object. So you need to explicitly create the Thread class object. We are passing the object of your class that implements Runnable so that your class run() method may execute.

3 Using the Thread Class: Thread(String Name)

We can directly use the Thread class to spawn new threads using the constructors defined above.

FileName: MyThread1.java

```
public class MyThread1
{
// Main method
public static void main(String argvs[])
{
// creating an object of the Thread class using the constructor Thread(String name)
Thread t= new Thread("My first thread");

// the start() method moves the thread to the active state
t.start();
// getting the thread name by invoking the getName() method
String str = t.getName();
System.out.println(str);
    }
    }
```

Output

```
My first thread
```

4 Using the Thread Class: Thread(Runnable r, String name)

Observe the following program.

FileName: MyThread2.java

```
public class MyThread2 implements Runnable
{
public void run()
{
System.out.println("Now the thread is running ...");
}

// main method
public static void main(String argvs[])
{
// creating an object of the class MyThread2
Runnable r1 = new MyThread2();

// creating an object of the class Thread using Thread(Runnable r, String name)
```

```
Thread th1 = new Thread(r1, “My new thread”);


// the start() method moves the thread to the active state
th1.start();


// getting the thread name by invoking the getName() method
String str = th1.getName();
System.out.println(str);
}
}
```

Output

My new thread

Now the thread is running ...

Java Runnable Interface

Java runnable is an interface used to execute code on a concurrent thread. It is an interface which is implemented by any class if we want that the instances of that class should be executed by a thread.

The runnable interface has an undefined method run() with void as return type, and it takes in no arguments. The method summary of the run() method is given below-

Method	Description
public void run()	This method takes in no arguments. When the object of a class implementing Runnable class is used to create a thread, then the run method is invoked in the thread which executes separately.

The runnable interface provides a standard set of rules for the instances of classes which wish to execute code when they are active. The most common use case of the Runnable interface is when we want only to override the run method. When a thread is started by the object of any class which is implementing Runnable, then it invokes the run method in the separately executing thread.

A class that implements Runnable runs on a different thread without subclassing Thread as it instantiates a Thread instance and passes itself in as the target. This becomes important as classes should not be subclassed unless there is an intention of modifying or enhancing the fundamental behavior of the class.

Runnable class is extensively used in network programming as each thread represents a separate flow of control. Also in multi-threaded programming, Runnable class is used. This interface is present in **java.lang** package.

Implementing Runnable

It is the easiest way to create a thread by implementing Runnable. One can create a thread on any object by implementing Runnable. To implement a Runnable, one has only to implement the run method.

public void run()

In this method, we have the code which we want to execute on a concurrent thread. In this method, we can use variables, instantiate classes, and perform an action like the same way the main thread does. The thread remains until the return of this method. The run method establishes an entry point to a new thread.

How to create a thread using Runnable interface

To create a thread using runnable, use the following code-

```
1   Runnable runnable = new MyRunnable();
2
```

3 Thread thread = new Thread(runnable);

4 thread.start();

The thread will execute the code which is mentioned in the run() method of the Runnable object passed in its argument.

Multithreading in Java

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

Advantages of Java Multithreading

1 It doesn't block the user because threads are independent and you can perform multiple operations at the same time.

2 You can perform many operations together, so it saves time.

3 Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

Multitasking

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved in two ways:

1 Process-based Multitasking (Multiprocessing)

2 Thread-based Multitasking (Multithreading)

1 Process-based Multitasking (Multiprocessing)

- Each process has an address in memory. In other words, each process allocates a separate memory area.
- A process is heavyweight.
- Cost of communication between the process is high.
- Switching from one process to another requires some time for saving and loading registers, memory maps, updating lists, etc.

2 Thread-based Multitasking (Multithreading)

- Threads share the same address space.
- A thread is lightweight.
- Cost of communication between the thread is low.

Synchronization in Java

Synchronization in Java is the capability to control the access of multiple threads to any shared resource.

Java Synchronization is better option where we want to allow only one thread to access the shared resource.

Why use Synchronization?

The synchronization is mainly used to

1 To prevent thread interference.

2 To prevent consistency problem.

Types of Synchronization

There are two types of synchronization

1 Process Synchronization

2. Thread Synchronization

Here, we will discuss only thread synchronization.

Thread Synchronization

There are two types of thread synchronization mutual exclusive and inter-thread communication.

1 Mutual Exclusive

  1 Synchronized method.

  2 Synchronized block.

  3 Static synchronization.

2 Co-operation (Inter-thread communication in java)

Mutual Exclusive

Mutual Exclusive helps keep threads from interfering with one another while sharing data. It can be achieved by using the following three ways:

1 By Using Synchronized Method

2 By Using Synchronized Block

3 By Using Static Synchronization

Exception Handling in Java

- Exception Handling
- Advantage of Exception Handling
- Hierarchy of Exception classes
- Types of Exception
- Exception Example
- Scenarios where an exception may occur

The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

In this tutorial, we will learn about Java exceptions, it's types, and the difference between checked and unchecked exceptions.

What is Exception in Java?

Dictionary Meaning: Exception is an abnormal condition.

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

What is Exception Handling?

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Advantage of Exception Handling

The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions. Let's consider a scenario:

1 statement 1;

2 statement 2;

3 statement 3;

4 statement 4;

5 statement 5;//exception occurs

6 statement 6;

7 statement 7;

8 statement 8;

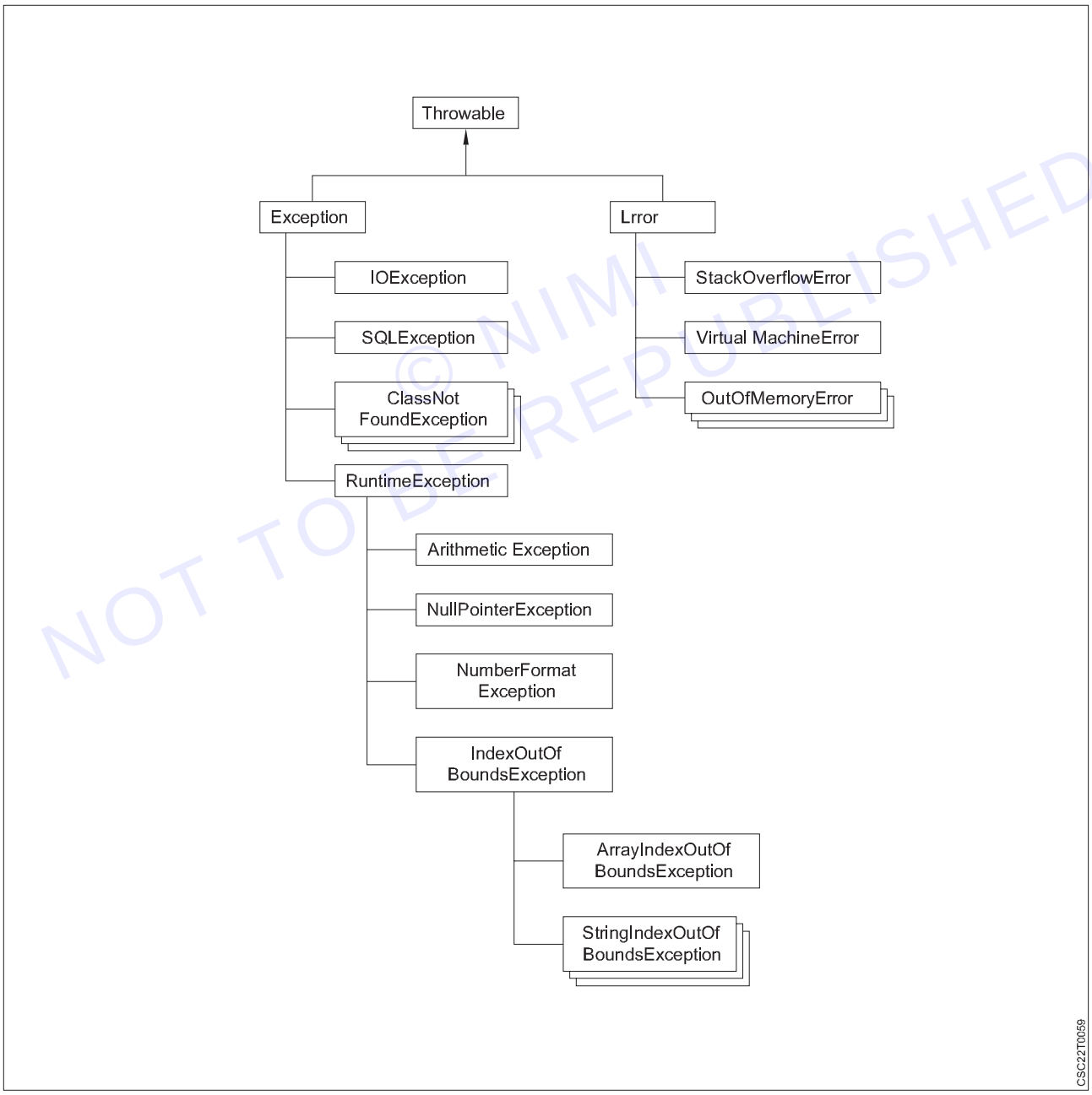
9 statement 9;

10 statement 10;

Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed. However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling in Java.

Hierarchy of Java Exception classes

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:



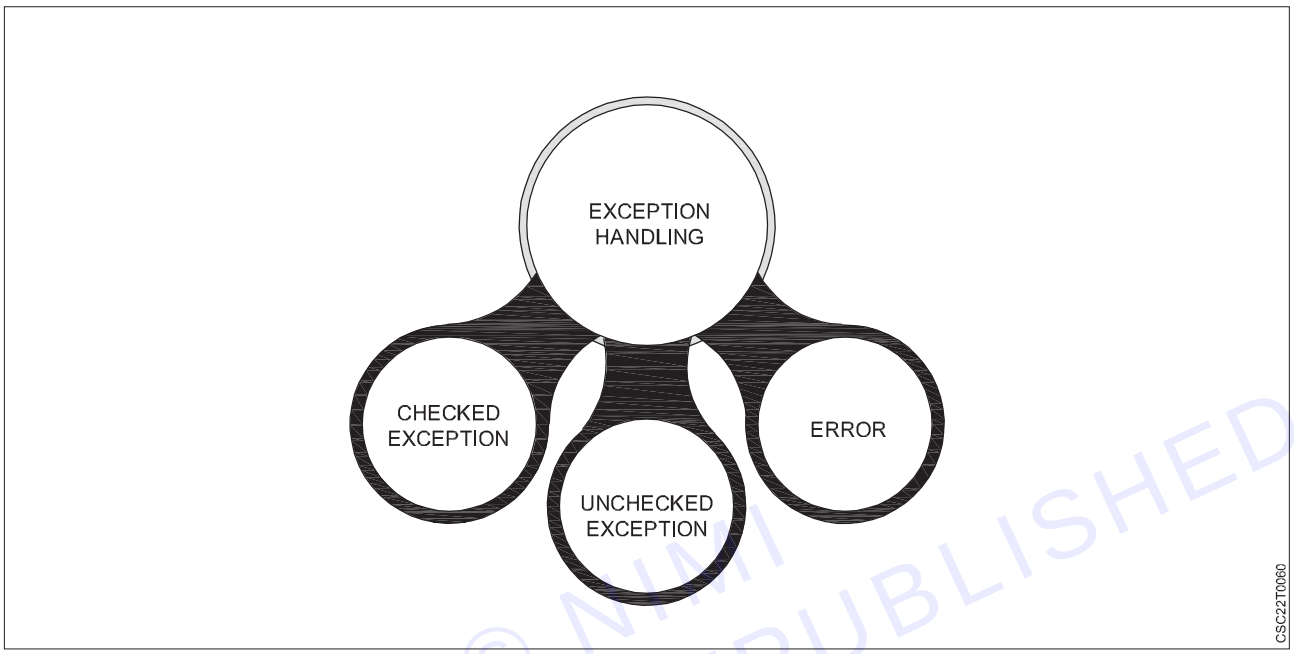
Types of Java Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

1 Checked Exception

2 Unchecked Exception

3 Error



Difference between Checked and Unchecked Exceptions

1 Checked Exception

The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

2 Unchecked Exception

The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

3 Error

Error is irrecoverable. Some example of errors are OutOfMemoryError, VirtualMachineError, AssertionError etc.

Java Exception Keywords

Java provides five keywords that are used to handle the exception. The following table describes each.

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The “catch” block is used to handle the exception. It must be preceded by try block which means we can’t use catch block alone. It can be followed by finally block later.
finally	The “finally” block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.

throw	The “throw” keyword is used to throw an exception.
throws	The “throws” keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn’t throw an exception. It is always used with method signature.

Java Exception Handling Example

Let’s see an example of Java Exception Handling in which we are using a try-catch statement to handle the exception.

JavaExceptionExample.java

```
public class JavaExceptionExample{
  public static void main(String args[]){
   try{
      //code that may raise exception
      int data=100/0;
   }catch(ArithmeticException e){System.out.println(e);}
   //rest code of the program
   System.out.println(“rest of the code...”);
  }
}
```

Test it Now

Output

Exception in thread main java.lang.ArithmeticException:/ by zero

rest of the code...

In the above example, 100/0 raises an ArithmeticException which is handled by a try-catch block.

Common Scenarios of Java Exceptions

There are given some scenarios where unchecked exceptions may occur. They are as follows:

1 A scenario where ArithmeticException occurs

If we divide any number by zero, there occurs an ArithmeticException.

```
1  int a=50/0;//ArithmeticException
```

2 A scenario where NullPointerException occurs

If we have a null value in any variable, performing any operation on the variable throws a NullPointerException.

```
1  String s=null;
2  System.out.println(s.length());//NullPointerException
```

3 A scenario where NumberFormatException occurs

If the formatting of any variable or number is mismatched, it may result into NumberFormatException. Suppose we have a string variable that has characters; converting this variable into digit will cause NumberFormatException.

```
1  String s=”abc”;
2  int i=Integer.parseInt(s);//NumberFormatException
```

4 A scenario where ArrayIndexOutOfBoundsException occurs

When an array exceeds to it’s size, the ArrayIndexOutOfBoundsException occurs. there may be other reasons to occur ArrayIndexOutOfBoundsException. Consider the following statements.

```
1  int a[]=new int[5];
2  a[10]=50; //ArrayIndexOutOfBoundsException
```

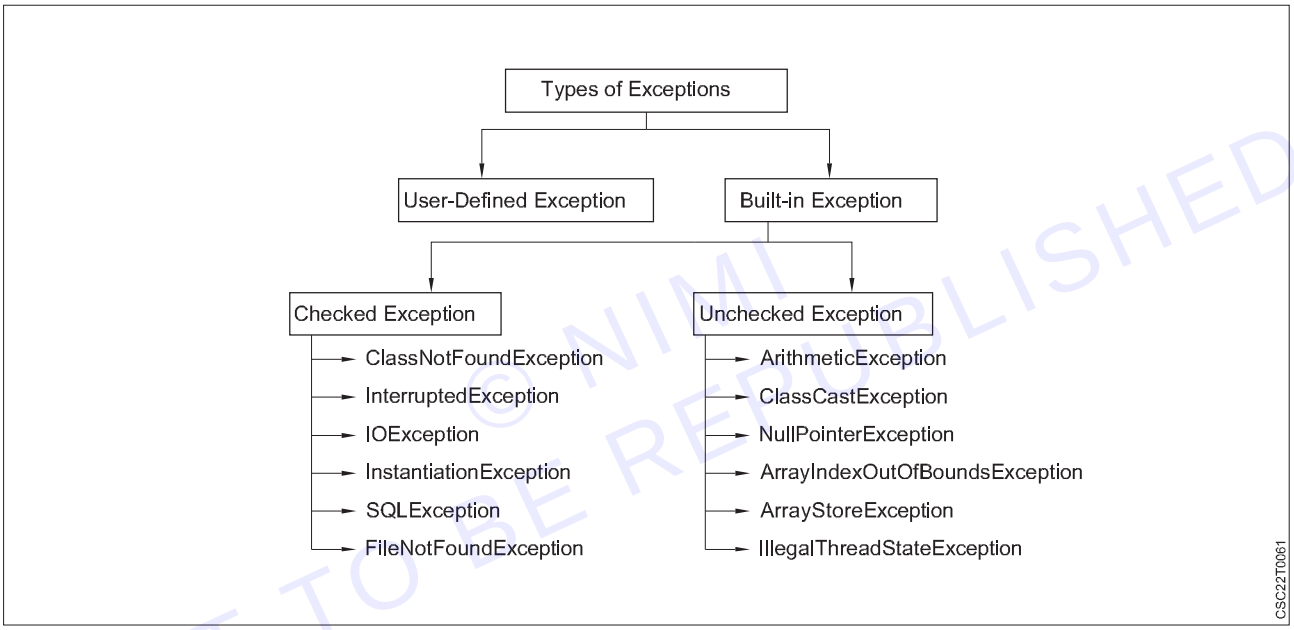
Types of Exception in Java

In Java, **exception** is an event that occurs during the execution of a program and disrupts the normal flow of the program's instructions. Bugs or errors that we don't want and restrict our program's normal execution of code are referred to as exceptions. In this section, we will focus on the **types of exceptions in Java** and the differences between the two.

Exceptions can be categorized into two ways:

1 Built-in Exceptions
   - Checked Exception
   - Unchecked Exception

2 User-Defined Exceptions



Built-in Exception

Exceptions that are already available in **Java libraries** are referred to as **built-in exception**. These exceptions are able to define the error situation so that we can understand the reason of getting this error. It can be categorized into two broad categories, i.e., **checked exceptions** and **unchecked exception**.

Checked Exception

Checked exceptions are called **compile-time** exceptions because these exceptions are checked at compile-time by the compiler. The compiler ensures whether the programmer handles the exception or not. The programmer should have to handle the exception; otherwise, the system has shown a compilation error.

CheckedExceptionExample.java

```
import java.io.*;
class CheckedExceptionExample {
   public static void main(String args[]) {
      FileInputStream file_data = null;
      file_data = new FileInputStream("C:/Users/ajeet/OneDrive/Desktop/Hello.txt");
      int m;
```

```
        while(( m = file_data.read() ) != -1) {
            System.out.print((char)m);
        }
        file_data.close();
    }
}
```

In the above code, we are trying to read the **Hello.txt** file and display its data or content on the screen. The program throws the following exceptions:

1 The **FileInputStream(File filename)** constructor throws the FileNotFoundException that is checked exception.

2 The **read()** method of the **FileInputStream** class throws the **IOException**.

3 The **close()** method also throws the IOException.

Output:

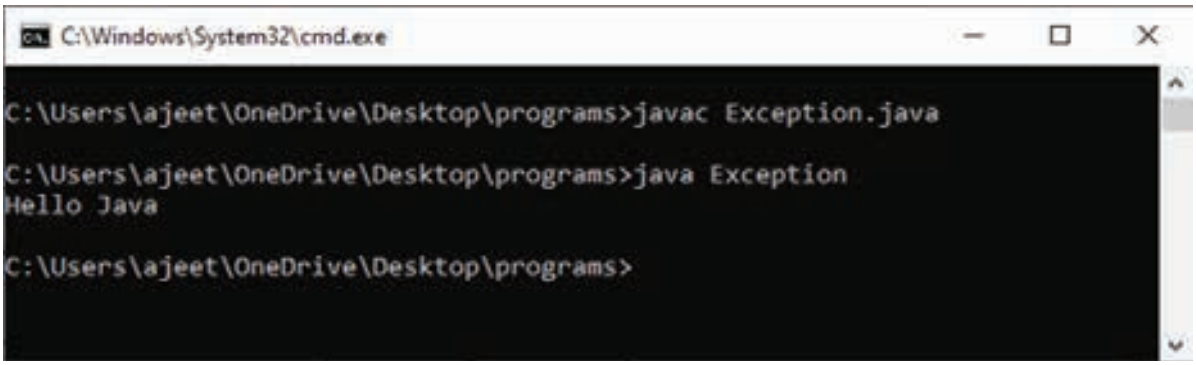
How to resolve the error?

There are basically two ways through which we can solve these errors.

1 The exceptions occur in the main method. We can get rid from these compilation errors by declaring the exception in the main method using the throws We only declare the IOException, not FileNotFoundException, because of the child-parent relationship. The IOException class is the parent class of FileNotFoundException, so this exception will automatically cover by IOException. We will declare the exception in the following way:

```
class Exception{
    public static void main(String args[])  throws IOException {
        ...
        ...
}
```

If we compile and run the code, the errors will disappear, and we will see the data of the file.

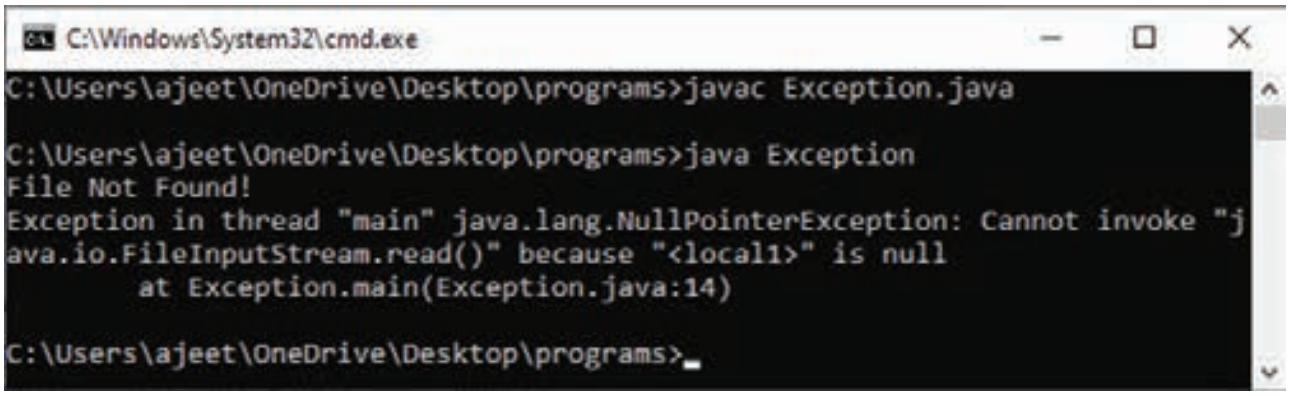


2 We can also handle these exception using try-catch However, the way which we have used above is not correct. We have to a give meaningful message for each exception type. By doing that it would be easy to understand the error. We will use the try-catch block in the following way:

Exception.java

```
import java.io.*;
class Exception{
   public static void main(String args[]) {
      FileInputStream file_data = null;
      try{
         file_data = new FileInputStream("C:/Users/ajeet/OneDrive/Desktop/programs/Hell.txt");
      }catch(FileNotFoundException fnfe){
         System.out.println("File Not Found!");
      }
      int m;
      try{
         while(( m = file_data.read() ) != -1) {
            System.out.print((char)m);
         }
         file_data.close();
      }catch(IOException ioe){
         System.out.println("I/O error occurred: "+ioe);
      }
   }
}
```

We will see a proper error message "File Not Found!" on the console because there is no such file in that location.



Unchecked Exceptions

The unchecked exceptions are just opposite to the checked exceptions. The compiler will not check these exceptions at compile time. In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error. Usually, it occurs when the user provides bad data during the interaction with the program.

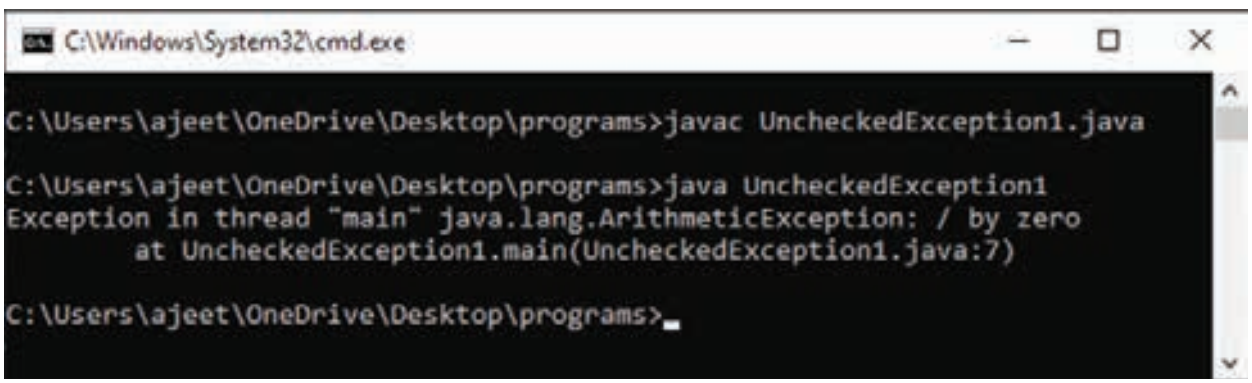
Note: The RuntimeException class is able to resolve all the unchecked exceptions because of the child-parent relationship.

UncheckedExceptionExample1.java

```
class UncheckedExceptionExample1 {
  public static void main(String args[])
  {
   int postive = 35;
   int zero = 0;
   int result = positive/zero;
   //Give Unchecked Exception here.
System.out.println(result);
  }
}
```

In the above program, we have divided 35 by 0. The code would be compiled successfully, but it will throw an ArithmeticException error at runtime. On dividing a number by 0 throws the divide by zero exception that is a uncheck exception.

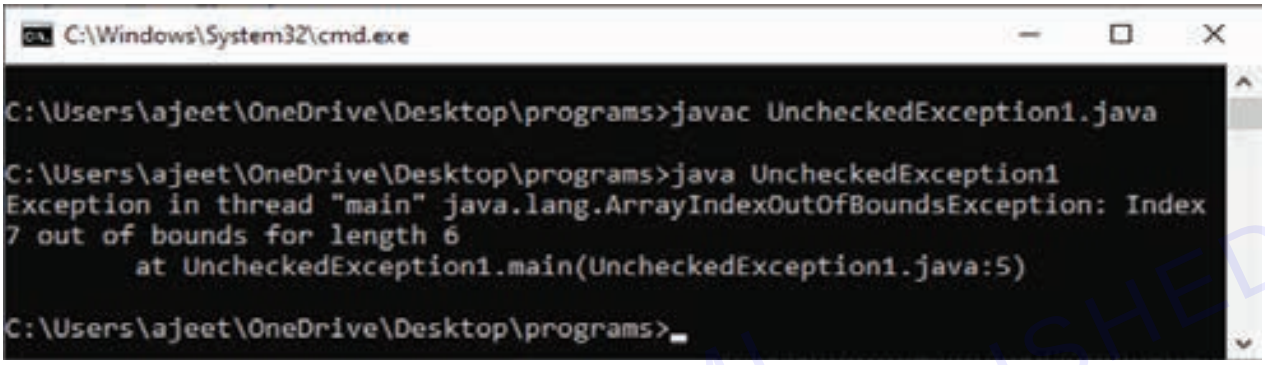
Output:



UncheckedException1.java

```
class UncheckedException1 {
  public static void main(String args[])
  {
   int num[] ={10,20,30,40,50,60};
   System.out.println(num[7]);
  }
}
```

Output



In the above code, we are trying to get the element located at position 7, but the length of the array is 6. The code compiles successfully, but throws the ArrayIndexOutOfBoundsException at runtime

User-defined Exception

In Java, we already have some built-in exception classes like ArrayIndexOutOfBoundsException, NullPointerException, and ArithmeticException. These exceptions are restricted to trigger on some predefined conditions. In Java, we can write our own exception class by extends the Exception class. We can throw our own exception on a particular condition using the throw keyword. For creating a user-defined exception, we should have basic knowledge of the try-catch block and throw keyword.

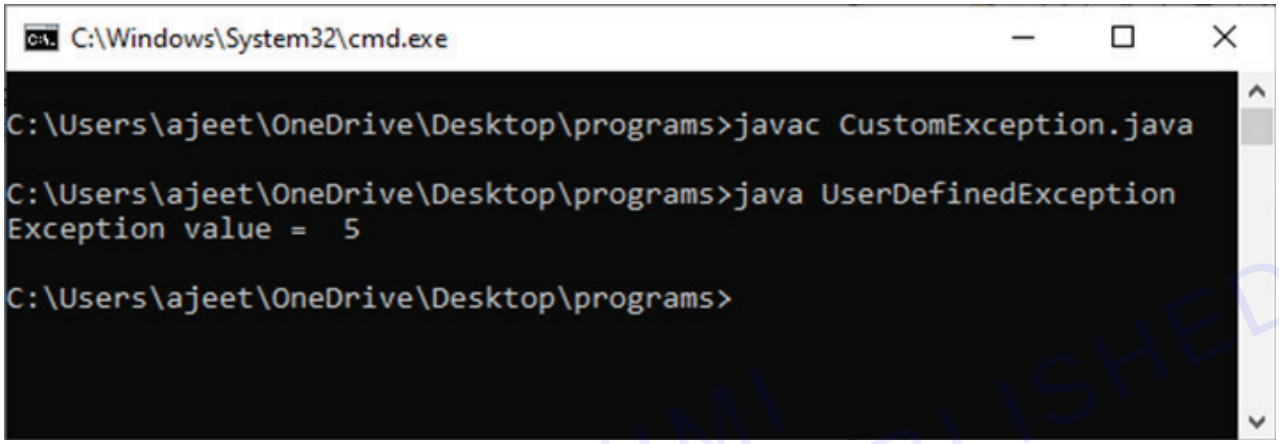
Let's write a Java program and create user-defined exception.

UserDefinedException.java

```
import java.util.*;
class UserDefinedException{
   public static void main(String args[]){
      try{
         throw new NewException(5);
      }
      catch(NewException ex){
         System.out.println(ex) ;
      }
   }
}
class NewException extends Exception{
   int x;
```

```
        NewException(int y) {
        x=y;
    }
    public String toString(){
        return ("Exception value =  "+x) ;
    }
}
```

Output:



Description

In the above code, we have created two classes, i.e., UserDefinedException and NewException. The User DefinedException has our main method, and the NewException class is our user-defined exception class, which extends exception. In the NewException class, we create a variable x of type integer and assign a value to it in the constructor. After assigning a value to that variable, we return the exception message.

In the UserDefinedException class, we have added a try-catch block. In the try section, we throw the exception, i.e., NewException and pass an integer to it. The value will be passed to the NewException class and return a message. We catch that message in the catch block and show it on the screen.

Difference Between Checked and Unchecked Exception

S.No.	Checked Exception	Unchecked Exception
1	These exceptions are checked at compile time. These exceptions are handled at compile time too.	These exceptions are just opposite to the checked exceptions. These exceptions are not checked and handled at compile time.
2	These exceptions are direct subclasses of exception but not extended from RuntimeException class.	They are the direct subclasses of the RuntimeException class.
3	The code gives a compilation error in the case when a method throws a checked exception. The compiler is not able to handle the exception on its own.	The code compiles without any error because the exceptions escape the notice of the compiler. These exceptions are the results of user-created errors in programming logic.
4	These exceptions mostly occur when the probability of failure is too high.	These exceptions occur mostly due to programming mistakes.
5	Common checked exceptions include IOException, DataAccessException, InterruptedException, etc.	Common unchecked exceptions include ArithmeticException,InvalidClassException, NullPointerException, etc.

6	These exceptions are propagated using the throws keyword.	These are automatically propagated.
7	It is required to provide the try-catch and try-finally block to handle the checked exception.	In the case of unchecked exception it is not mandatory.

Bugs or errors that we don't want and restrict the normal execution of the programs are referred to as exceptions.

ArithmeticException, ArrayIndexOutOfBoundExceptions, ClassNotFoundExceptions etc. are come in the category of Built-in Exception. Sometimes, the built-in exceptions are not sufficient to explain or describe certain situations. For describing these situations, we have to create our own exceptions by creating an exception class as a subclass of the Exception class. These types of exceptions come in the category of User-Defined Exception.

Java try-catch block

Java try block

Java try block is used to enclose the code that might throw an exception. It must be used within the method.

If an exception occurs at the particular statement in the try block, the rest of the block code will not execute. So, it is recommended not to keep the code in try block that will not throw an exception.

Java try block must be followed by either catch or finally block.

Syntax of Java try-catch

```
try{
//code that may throw an exception
}catch(Exception_class_Name ref){}
```

Syntax of try-finally block

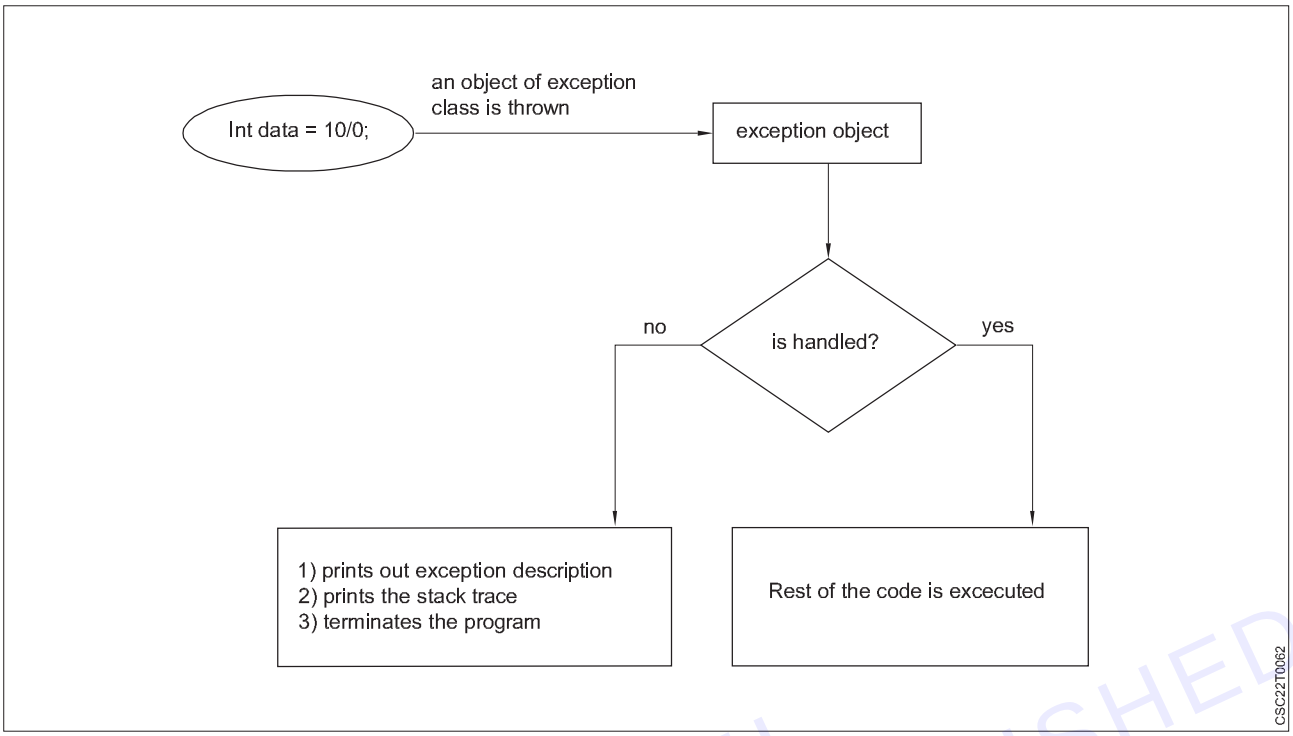
```
try{
//code that may throw an exception
}finally{}
```

Java catch block

Java catch block is used to handle the Exception by declaring the type of exception within the parameter. The declared exception must be the parent class exception (i.e., Exception) or the generated exception type. However, the good approach is to declare the generated type of exception.

The catch block must be used after the try block only. You can use multiple catch block with a single try block.

Internal Working of Java try-catch block



The JVM firstly checks whether the exception is handled or not. If exception is not handled, JVM provides a default exception handler that performs the following tasks:

- Prints out exception description.
- Prints the stack trace (Hierarchy of methods where the exception occurred).
- Causes the program to terminate.

But if the application programmer handles the exception, the normal flow of the application is maintained, i.e., rest of the code is executed.

Problem without exception handling

Let's try to understand the problem if we don't use a try-catch block.

Example 1

TryCatchExample1.java

```
public class TryCatchExample1 {

    public static void main(String[] args) {

        int data=50/0; //may throw exception

        System.out.println("rest of the code");

    }

}
```

Output

Exception in thread “main” java.lang.ArithmeticException: / by zero

As displayed in the above example, the rest of the code is not executed (in such case, the rest of the code statement is not printed).

There might be 100 lines of code after the exception. If the exception is not handled, all the code below the exception won’t be executed.

Solution by exception handling

Let’s see the solution of the above problem by a java try-catch block.

Example 2

TryCatchExample2.java

```
public class TryCatchExample2 {

    public static void main(String[] args) {
        try
        {
        int data=50/0; //may throw exception
        }
            //handling the exception
       catch(ArithmeticException e)
        {
            System.out.println(e);
        }
        System.out.println(“rest of the code”);
    }

}
```

Output

java.lang.ArithmeticException: / by zero

rest of the code

As displayed in the above example, the rest of the code is executed, i.e., the rest of the code statement is printed.

Example 3

In this example, we also kept the code in a try block that will not throw an exception.

TryCatchExample3.java

```
public class TryCatchExample3 {

    public static void main(String[] args) {
        try
        {
```

```
        int data=50/0; //may throw exception
                    // if exception occurs, the remaining statement will not exceute
        System.out.println("rest of the code");
        }
         // handling the exception
        catch(ArithmeticException e)
        {
           System.out.println(e);
        }

    }

}
```

Output

java.lang.ArithmeticException: / by zero

Here, we can see that if an exception occurs in the try block, the rest of the block code will not execute.

Example 4

Here, we handle the exception using the parent class exception.

TryCatchExample4.java

```
public class TryCatchExample4 {

    public static void main(String[] args) {
        try
        {
        int data=50/0; //may throw exception
        }
         // handling the exception by using Exception class
        catch(Exception e)
        {
           System.out.println(e);
        }
        System.out.println("rest of the code");
    }

}
```

Output

java.lang.ArithmeticException: / by zero

rest of the code

Java throw Exception

In Java, exceptions allow us to write good quality codes where the errors are checked at the compile time instead of runtime and we can create custom exceptions making the code recovery and debugging easier.

Java throw keyword

The Java throw keyword is used to throw an exception explicitly.

We specify the exception object which is to be thrown. The Exception has some message with it that provides the error description. These exceptions may be related to user inputs, server, etc.

We can throw either checked or unchecked exceptions in Java by throw keyword. It is mainly used to throw a custom exception. We will discuss custom exceptions later in this section.

We can also define our own set of conditions and throw an exception explicitly using throw keyword. For example, we can throw ArithmeticException if we divide a number by another number. Here, we just need to set the condition and throw exception using throw keyword.

The syntax of the Java throw keyword is given below.

throw Instance i.e.,

```
1   throw new exception_class("error message");
```

Let's see the example of throw IOException.

```
1   throw new IOException("sorry device error");
```

Where the Instance must be of type Throwable or subclass of Throwable. For example, Exception is the sub class of Throwable and the user-defined exceptions usually extend the Exception class.

Java throw keyword Example

Example 1: Throwing Unchecked Exception

In this example, we have created a method named validate() that accepts an integer as a parameter. If the age is less than 18, we are throwing the ArithmeticException otherwise print a message welcome to vote.

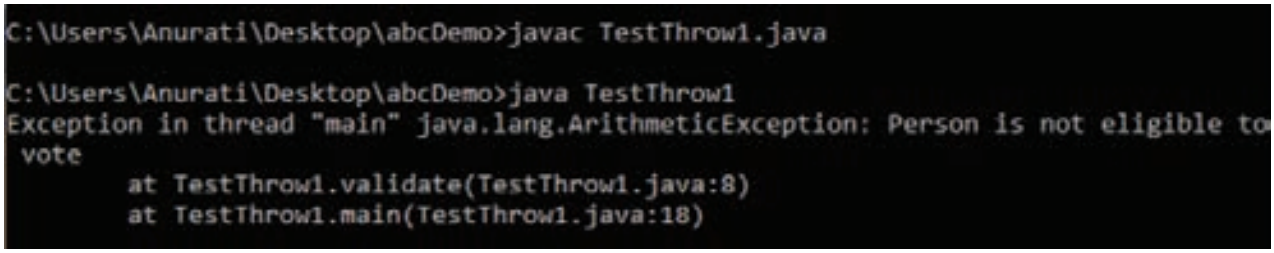
TestThrow1.java

In this example, we have created the validate method that takes integer value as a parameter. If the age is less than 18, we are throwing the ArithmeticException otherwise print a message welcome to vote.

```
public class TestThrow1 {
  //function to check if person is eligible to vote or not
  public static void validate(int age) {
    if(age<18) {
      //throw Arithmetic exception if not eligible to vote
      throw new ArithmeticException("Person is not eligible to vote");
    }
    else {
      System.out.println("Person is eligible to vote!!");
    }
  }
  //main method
  public static void main(String args[]){
```

```
        //calling the function
        validate(13);
        System.out.println("rest of the code...");
    }
}
```

Output



The above code throw an unchecked exception. Similarly, we can also throw unchecked and user defined exceptions.

Note: If we throw unchecked exception from a method, it is must to handle the exception or declare in throws clause.

If we throw a checked exception using throw keyword, it is must to handle the exception using catch block or the method must declare it using throws declaration.

Example 2: Throwing Checked Exception

Note: Every subclass of Error and RuntimeException is an unchecked exception in Java. A checked exception is everything else under the Throwable class.

TestThrow2.java

```
import java.io.*;

public class TestThrow2 {

    //function to check if person is eligible to vote or not
    public static void method() throws FileNotFoundException {

        FileReader file = new FileReader("C:\\Users\\Anurati\\Desktop\\abc.txt");
        BufferedReader fileInput = new BufferedReader(file);



        throw new FileNotFoundException();

    }
    //main method
    public static void main(String args[]){
        try
        {
```

```
            method();
        }
        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        System.out.println("rest of the code...");
    }
}
```

Output

```
C:\Users\Anurati\Desktop\abcDemo>javac TestThrow2.java

C:\Users\Anurati\Desktop\abcDemo>java TestThrow2
java.io.FileNotFoundException
        at TestThrow2.method(TestThrow2.java:12)
        at TestThrow2.main(TestThrow2.java:22)
rest of the code...
```

Example 3: Throwing User-defined Exception

exception is everything else under the Throwable class.

TestThrow3.java

```
// class represents user-defined exception
class UserDefinedException extends Exception
{
    public UserDefinedException(String str)
    {
        // Calling constructor of parent Exception
        super(str);
    }
}
// Class that uses above MyException
public class TestThrow3
{
    public static void main(String args[])
    {
        try
        {
            // throw an object of user defined exception
            throw new UserDefinedException("This is user-defined exception");
        }
```

```
            catch (UserDefinedException ude)
            {
                System.out.println("Caught the exception");
                // Print the message from MyException object
                System.out.println(ude.getMessage());
            }
        }
    }
```

Output

```
C:\Users\Anurati\Desktop\abcDemo>javac TestThrow3.java

C:\Users\Anurati\Desktop\abcDemo>java TestThrow3
Caught the exception
This is user-defined exception
```

Difference between final, finally and finalize

The final, finally, and finalize are keywords in Java that are used in exception handling. Each of these keywords has a different functionality. The basic difference between final, finally and finalize is that the final is an access modifier, finally is the block in Exception Handling and finalize is the method of object class.

Along with this, there are many differences between final, finally and finalize. A list of differences between final, finally and finalize are given below:

S.No.	Key	Final	Finally	Finalize
1	Definition	final is the keyword and access modifier which is used to apply restrictions on a class, method or variable.	finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not.	finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.
2	Applicable to	Final keyword is used with the classes, methods and variables.	Finally block is always related to the try and catch block in exception handling.	finalize() method is used with the objects.
3	Functionality	(1) Once declared, final variable becomes constant and cannot be modified. (2) final method cannot be overridden by sub class. (3) final class cannot be inherited.	(1) finally block runs the important code even if exception occurs or not. (2) finally block cleans up all the resources used in try block	cla
4	Execution	Final method is executed only when we call it.	Finally block is executed as soon as the try-catch block is executed. It's execution is not dependent on the exception.	finalize method is executed just before the object is destroyed.

Java final Example

Let's consider the following example where we declare final variable age. Once declared it cannot be modified.

FinalExampleTest.java

```
public class FinalExampleTest {
   //declaring final variable
   final int age = 18;
   void display() {

   // reassigning value to age variable
   // gives compile time error
   age = 55;
   }

   public static void main(String[] args) {

   FinalExampleTest obj = new FinalExampleTest();
   // gives compile time error
   obj.display();
   }
}
```

Output

```
C:\Users\Anurati\Desktop\abcDemo>javac FinalExampleTest.java
FinalExampleTest.java:10: error: cannot assign a value to final variable age
        age = 55;
        ^
1 error
```

In the above example, we have declared a variable final. Similarly, we can declare the methods and classes final using the final keyword.

Java finally Example

Let's see the below example where the Java code throws an exception and the catch block handles that exception. Later the finally block is executed after the try-catch block. Further, the rest of the code is also executed normally.

FinallyExample.java

```
public class FinallyExample {
    public static void main(String args[]){
```

```
        try {
          System.out.println("Inside try block");
        // below code throws divide by zero exception
         int data=25/0;
         System.out.println(data);
        }
        // handles the Arithmetic Exception / Divide by zero exception
        catch (ArithmeticException e){
          System.out.println("Exception handled");
          System.out.println(e);
        }
        // executes regardless of exception occurred or not
        finally {
          System.out.println("finally block is always executed");
        }
        System.out.println("rest of the code...");
        }
      }
```

Output:

```
C:\Users\Anurati\Desktop\abcDemo>java FinallyExample.java
Inside try block
Exception handled
java.lang.ArithmeticException: / by zero
finally block is always executed
rest of the code...
```

Java finalize Example

FinalizeExample.java

```
public class FinalizeExample {
    public static void main(String[] args)
    {
        FinalizeExample obj = new FinalizeExample();
        // printing the hashcode
        System.out.println("Hashcode is: " + obj.hashCode());
        obj = null;
        // calling the garbage collector using gc()
```

```
        System.gc();
        System.out.println("End of the garbage collection");
    }
    // defining the finalize method
    protected void finalize()
    {
        System.out.println("Called the finalize() method");
    }
}
```

Output:

```
C:\Users\Anurati\Desktop\abcDemo>javac FinalizeExample.java
Note: FinalizeExample.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\Anurati\Desktop\abcDemo>java FinalizeExample
Hashcode is: 746292446
End of the garbage collection
Called the finalize() method
```

LESSON 109 - 115 : Abstract Classes and Interfaces in JAVA

Concept of Virtual methods

Virtual Function in Java

A virtual function or virtual method in an OOP language is a function or method used to override the behavior of the function in an inherited class with the same signature to achieve the polymorphism.

When the programmers switch the technology from C++ to Java, they think about where is the virtual function in Java. In C++, the virtual function is defined using the virtual keyword, but in Java, it is achieved using different techniques. See Virtual function in C++.

Java is an object-oriented programming language; it supports OOPs features such as polymorphism, abstraction, inheritance, etc. These concepts are based on objects, classes, and member functions.

How to use Virtual function in Java

The virtual keyword is not used in Java to define the virtual function; instead, the virtual functions and methods are achieved using the following techniques:

- We can override the virtual function with the inheriting class function using the same function name. Generally, the virtual function is defined in the parent class and override it in the inherited class.
- The virtual function is supposed to be defined in the derived class. We can call it by referring to the derived class's object using the reference or pointer of the base class.
- A virtual function should have the same name and parameters in the base and derived class.
- For the virtual function, an IS-A relationship is necessary, which is used to define the class hierarchy in inheritance.
- The Virtual function cannot be private, as the private functions cannot be overridden.
- A virtual function or method also cannot be final, as the final methods also cannot be overridden.
- Static functions are also cannot be overridden; so, a virtual function should not be static.

Examples:-

Parent.Java:

```
1  class Parent {
2  void v1() //Declaring function
3  {
4  System.out.println("Inside the Parent Class");
5  }
6  }
```

Child.java:

```
1  public class Child extends Parent{
2  void v1() // Overriding function from the Parent class
3  {
4  System.out.println("Inside the Child Class");
5   }
6  public static void main(String args[]){
7  Parent ob1 = new Child(); //Refering the child class object using the parent class
8  ob1.v1();
9   }
```

```
10  }
```

Output:

Inside the Child Class

Concept of Abstract classes and methods

Data abstraction is the process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces (which you will learn more about in the next chapter).

The abstract keyword is a non-access modifier, used for classes and methods:

- **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

An abstract class can have both abstract and regular methods:

```
abstract class Animal {
public abstract void animalSound();
public void sleep() {
System.out.println("Zzz");
}
}
```

From the example above, it is not possible to create an object of the Animal class:

```
Animal myObj = new Animal(); // will generate an error
```

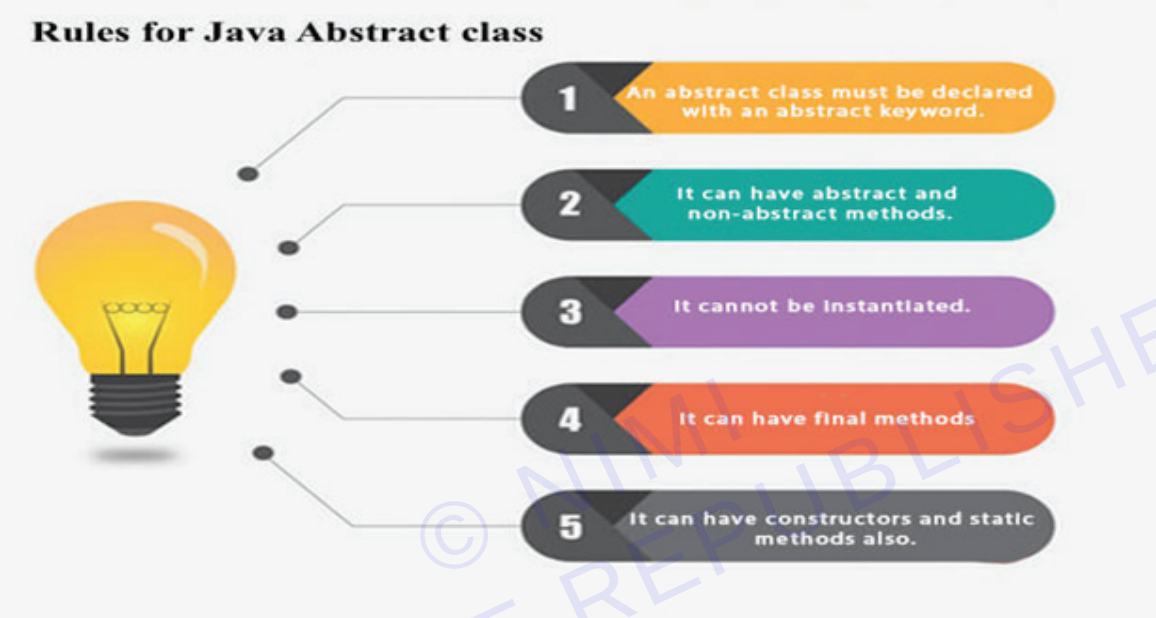
to access the abstract class, it must be inherited from another class. Let's convert the Animal class we used in the Polymorphism chapter to an abstract class:

Example

Get your own Java Server

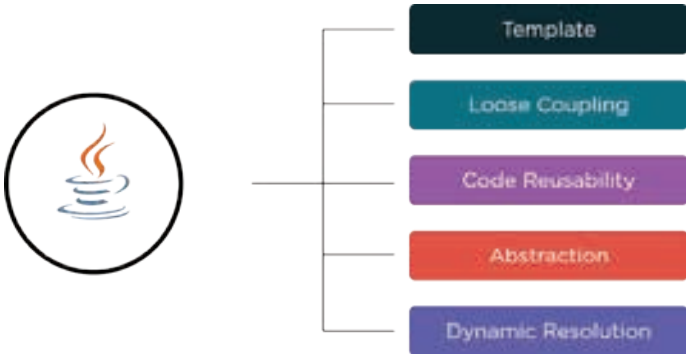
```
// Abstract class
abstract class Animal {
// Abstract method (does not have a body)
public abstract void animalSound();
// Regular method
public void sleep() {
System.out.println("Zzz");
}
}
// Subclass (inherit from Animal)
class Pig extends Animal {
public void animalSound() {
// The body of animalSound() is provided here
System.out.println("The pig says: wee wee");
```

```
}
}
class Main {
public static void main(String[] args) {
Pig myPig = new Pig(); // Create a Pig object
myPig.animalSound();
myPig.sleep();
}
```



Features of Abstract Class

Java-Abstract_Class_in_Java



The abstract class in Java enables the best way to execute the process of data abstraction by providing the developers with the option of hiding the code implementation. It also presents the end-user with a template that explains the methods involved.

Loose Coupling

Data abstraction in Java enables loose coupling, by reducing the dependencies at an exponential level.

Code Reusability

Using an abstract class in the code saves time. We can call the abstract method wherever the method is necessary. Abstract class avoids the process of writing the same code again.

Abstraction

Data abstraction in Java helps the developers hide the code complications from the end-user by reducing the project's complete characteristics to only the necessary components.

Dynamic Resolution

Using the support of dynamic method resolution, developers can solve multiple problems with the help of one abstract method.

Before moving forward, let's first understand how to declare an abstract class.

Java Interfaces and their advantages

An interface in Java is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Java Interface also represents the IS-A relationship

It cannot be instantiated just like the abstract class.

Since Java 8, we can have default and static methods in an interface.

Since Java 9, we can have private methods in an interface.

Method Overriding in Java

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Usage of Java Method Overriding

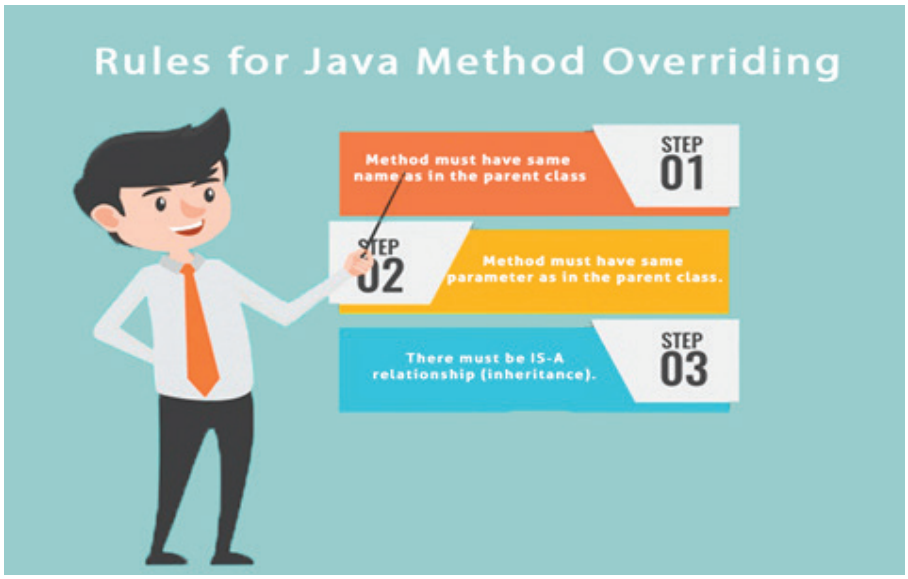
- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

1 The method must have the same name as in the parent class

2 The method must have the same parameter as in the parent class.

3 There must be an IS-A relationship (inheritance).



Example of method overriding

In this example, we have defined the run method in the subclass as defined in the parent class but it has some specific implementation. The name and parameter of the method are the same, and there is IS-A relationship between the classes, so there is method overriding.

```
1  //Java Program to illustrate the use of Java Method Overriding
2  //Creating a parent class.
3  class Vehicle{
4  //defining a method
5  void run(){System.out.println("Vehicle is running");}
6  }
7  //Creating a child class
8  class Bike2 extends Vehicle{
9  //defining the same method as in the parent class
10 void run(){System.out.println("Bike is running safely");}
11
12 public static void main(String args[]){
13 Bike2 obj = new Bike2();//creating object
14 obj.run();//calling method
15 }
16 }
```

Output: Bike is running safely

Polymorphism in Java

Polymorphism means “many forms”, and it occurs when we have many classes that are related to each other by inheritance.

Like we specified in the previous chapter; Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

For example, think of a superclass called Animal that has a method called animalSound(). Subclasses of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):

Example

Get your own Java Server

```
class Animal {
public void animalSound() {
System.out.println(“The animal makes a sound”);
}
}
class Pig extends Animal {
public void animalSound() {
System.out.println(“The pig says: wee wee”);
}
}
class Dog extends Animal {
public void animalSound() {
System.out.println(“The dog says: bow wow”);
}
}
```

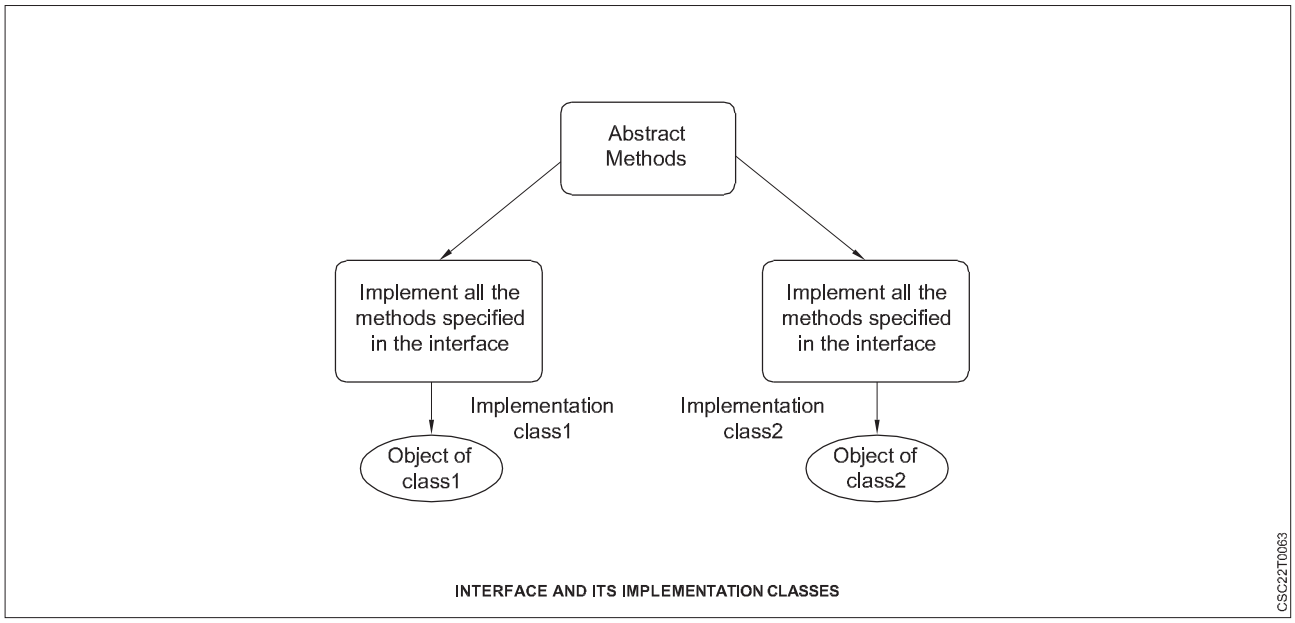

Creating, Implementing And Extending Interfaces

An interface in Java is syntactically similar to a class but can have only abstract methods declaration and constants as members.

In other words, an interface is a collection of abstract methods and constants (i.e. static and final fields). It is used to achieve complete abstraction.

Every interface in Java is abstract by default. So, it is not compulsory to write abstract keyword with an interface. Once an interface is defined, we can create any number of separate classes and can provide their own implementation for all the abstract methods defined by an interface.

A class that implements an interface is called implementation class. A class can implement any number of interfaces in Java. Every implementation class can have its own implementation for abstract methods specified in the interface as shown in the below figure



INTERFACE AND ITS IMPLEMENTATION CLASSES

Why do We Use Interface?

There are mainly five reasons or purposes of using an interface in Java. They are as follows:

1 In industry, architect-level people create interfaces, and then they are given to developers for writing classes by implementing interfaces provided.

2 Using interfaces is the best way to expose our project's API to some other projects. In other words, we can provide interface methods to the third-party vendors for their implementation.

For example, HDFC bank can expose methods or interfaces to various shopping carts.

3 Programmers use interface to customize features of software differently for different objects.

4 It is used to achieve full abstraction in java.

5 By using interfaces, we can achieve the functionality of multiple inheritance.

How to Declare Interface in Java?

In Java, an interface is declared syntactically much like a class. It is declared by using the keyword interface followed by interface name. It has the following general form:

Syntax:

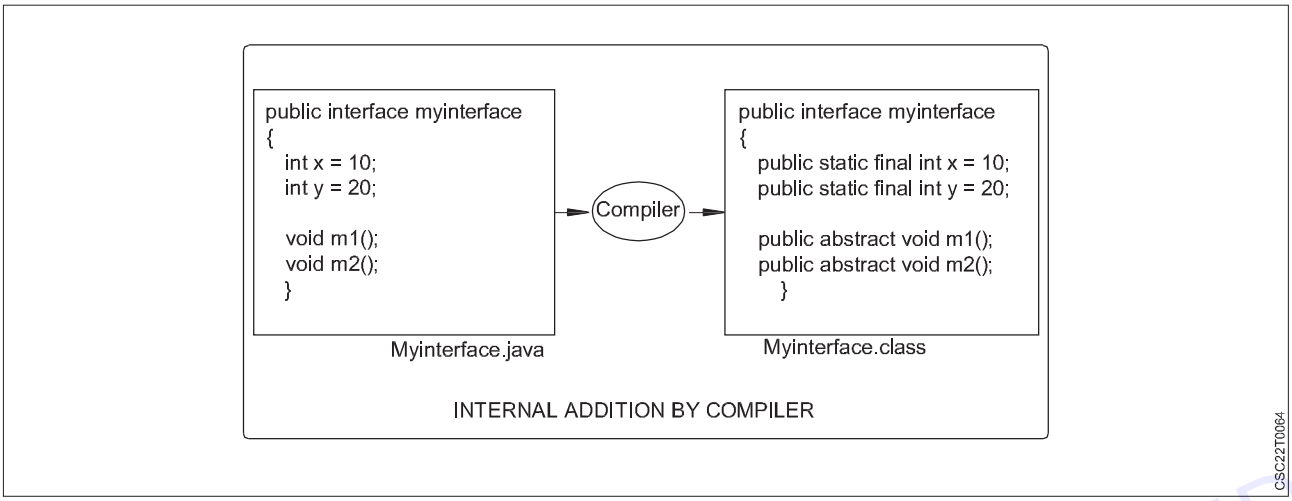
```
accessModifier interface interfaceName
{
// declare constant fields.
// declare methods that abstract by default.
}
```

Before interface keyword, we can specify access modifiers such as public, or default with abstract keyword. Let's understand the declaration of an interface with the help of an example.

```
public abstract interface MyInterfac
{
int x = 10; // public static final keyword invisibly present.
void m1(); // public and abstract keywords invisibly present.
void m2();
}
```

As you can see in the above example, both methods m1() and m2() defined in interface are declared with no body and do not have public or abstract modifiers present. The variable x declared in MyInterface is like a simple variable.

Java compiler automatically adds public and abstract keywords before to all interface methods. Moreover, it also adds public, static, and final keywords before interface variables. Look at the below figure to understand better.



Note:

a Earlier to Java 8, an interface could not define any implementation whatsoever. An interface can only declare abstract methods.

b Java 8 changed this rule. From Java 8 onwards, it is also possible to add a default implementation to an interface method.

c To support lambda functions, Java 8 has added a new feature to interface. We can also declare default methods and static methods inside interfaces.

d From Java 9 onwards, an interface can also declare private methods.

Features of Interface

There are the following features of an interface in Java. They are as follows:

1 Interface provides pure abstraction in Java. It also represents the Is-A relationship.

2 It can contain three types of methods: abstract, default, and static methods.

3 All the (non-default) methods declared in the interface are by default abstract and public. So, there is no need to write abstract or public modifiers before them.

4 The fields (data members) declared in an interface are by default public, static, and final. Therefore, they are just public constants. So, we cannot change their value by implementing class once they are initialized.

5 Interface cannot have constructors.

Extending Interface in Java with Example

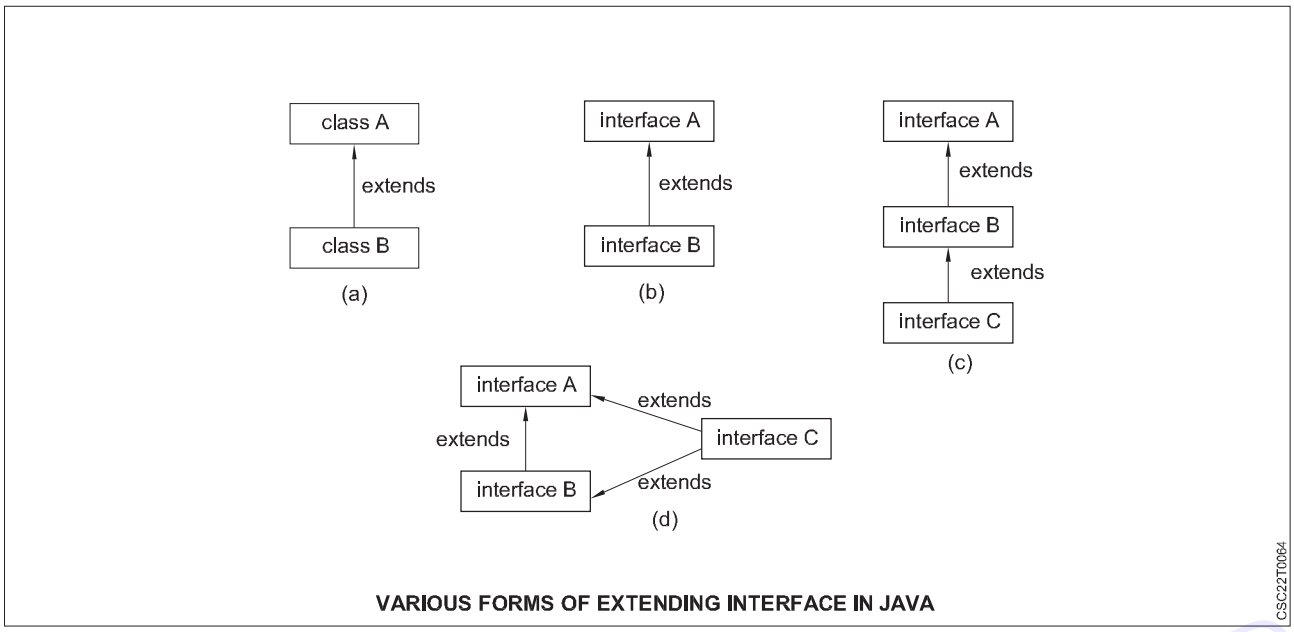
Like classes, an interface can also extend another interface. This means that an interface can be sub interfaces from other interfaces.

The new sub-interface will inherit all members of the super interface similar to subclasses. It can be done by using the keyword “extends”. It has the following general form:

Syntax:

```
interface interfaceName2 extends interfaceName1
{
// body of interfaceName2.
 }
```

Look at the below figure a, b, c, and d to understand better.



VARIOUS FORMS OF EXTENDING INTERFACE IN JAVA

1 We can define all the constants in one interface and methods in another interface. We can use constants in classes where methods are not required. Look at the example below.

```
interface A
{
int x = 10;
int y = 20;
 }
interface B extends A
{
void show();
}
```

The interface B would inherit both constants x and y into it.

2 We can also extend various interfaces together by a single interface. The general declaration is given below:

```
interface A
{
int x = 20;
int y = 30;
}
interface B extends A
{
void show();
}
interface C extends A, B
{
. . . . . .
 }
```

Key points:

1 An interface cannot extend classes because it would violate rules that an interface can have only abstract methods and constants.

2 An interface can extend Interface1, Interface2.

Implementing Interface in Java with Example

An interface is used as "superclass" whose properties are inherited by a class. A class can implement one or more than one interface by using a keyword implements followed by a list of interfaces separated by commas.

When a class implements an interface, it must provide an implementation of all methods declared in the interface and all its super interfaces.

Otherwise, the class must be declared abstract. The general syntax of a class that implements an interface is as follows:

Syntax:

1 accessModifier class className implements interfaceName

{

// method implementations;

// member declaration of class;

}

2 A more general form of interface implementation is given below:

accessModifier class className extends superClass implements interface1, interface2,.. .

{

// body of className.

}

This general form shows that a class can extend another class while implementing interfaces.

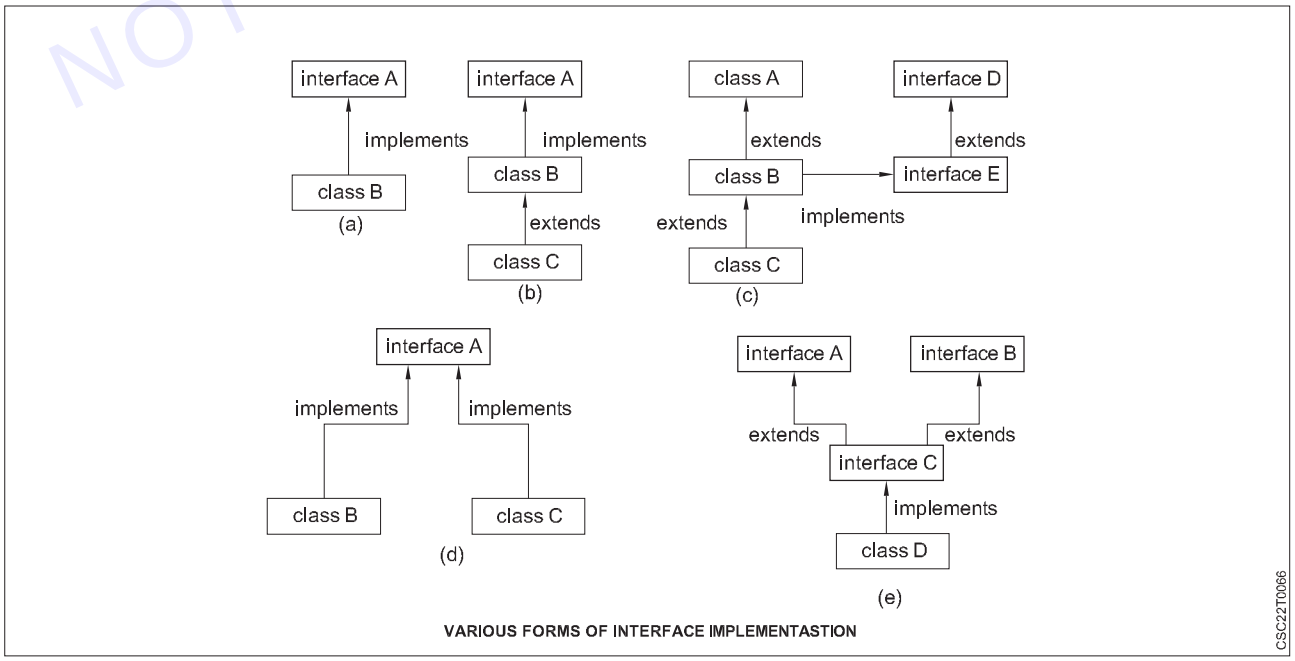
Key points:

1 All methods of interfaces when implementing in a class must be declared as public otherwise, you will get a compile-time error if any other modifier is specified.

2 Class extends class implements interface.

3 Class extends class implements Interface1, Interface2…

The implementation of interfaces can have the following general forms as shown in the below figure.



VARIOUS FORMS OF INTERFACE IMPLEMENTASTION

Packages in JAVA Creating and Using

In this tutorial, we are going to discuss packages in Java with the help of example programs. In small projects, all the Java files have unique names. So, it is not difficult to put them in a single folder.

But, in the case of huge projects where the number of Java files is large, it is very difficult to put files in a single folder because the manner of storing files would be disorganized.

Moreover, if different Java files in various modules of the project have the same name, it is not possible to store two Java files with the same name in the same folder because it may occur naming conflict.

This problem of naming conflict can be overcome by using the concept of packages. In Java, APIs consist of one or more packages where packages consist of many classes, classes contain several methods and fields.

When you create an application in Java, you should create a proper folder structure for better reusability, maintenance, and avoiding naming conflict but How?

What is Package in Java

A package is nothing but a physical folder structure (directory) that contains a group of related classes, interfaces, and sub-packages according to their functionality. It provides a convenient way to organize your work. The Java language has various in-built packages.

For example, java.lang, java.util, java.io, and java.net. All these packages are defined as a very clear and systematic packaging mechanism for categorizing and managing.

Let's understand it with the help of real-time examples.

Realtime Example of Packages in Java

A real-life example is when you download a movie, song, or game, you make a different folder for each category like movie, song, etc. In the same way, a group of packages in java is just like a library.

The classes and interfaces of a package are like books in the library that can reuse several times when we need them. This reusability nature of packages makes programming easy.

Therefore, when you create any software or application in Java programming language, they contain hundreds or thousands of individual classes and interfaces.

Advantage of using Packages in Java

1 **Maintenance:** Java packages are used for proper maintenance. If any developer newly joined a company, he can easily reach to files needed.

2 **Reusability:** We can place the common code in a common folder so that everybody can check that folder and use it whenever needed.

3 **Name conflict:** Packages help to resolve the naming conflict between the two classes with the same name. Assume that there are two classes with the same name Student.java. Each class will be stored in its own packages such as stdPack1 and stdPack2 without having any conflict of names.

4 **Organized:** It also helps in organizing the files within our project.

5 **Access Protection:** A package provides access protection. It can be used to provide visibility control. The members of the class can be defined in such a manner that they will be visible only to elements of that package.

Types of Packages in Java

There are mainly two types of packages available in Java. They are:

- User-defined package
- built-in package (also called predefined package)

Let's understand first user-defined package and how to create it easily in a Java program.

User-defined Package in Java

The package which is defined by the user is called user-defined or custom package in Java. It contains user-defined classes and interfaces. Let's understand how to create a user-defined package.

Creating User-defined Package

Java supports a keyword called “package” which is used to create user-defined packages in Java programming. The general syntax to create a package is as:

package packageName;

Here, packageName is the name of package. The package statement must be the first line in a Java source code file followed by one or more classes. For example:

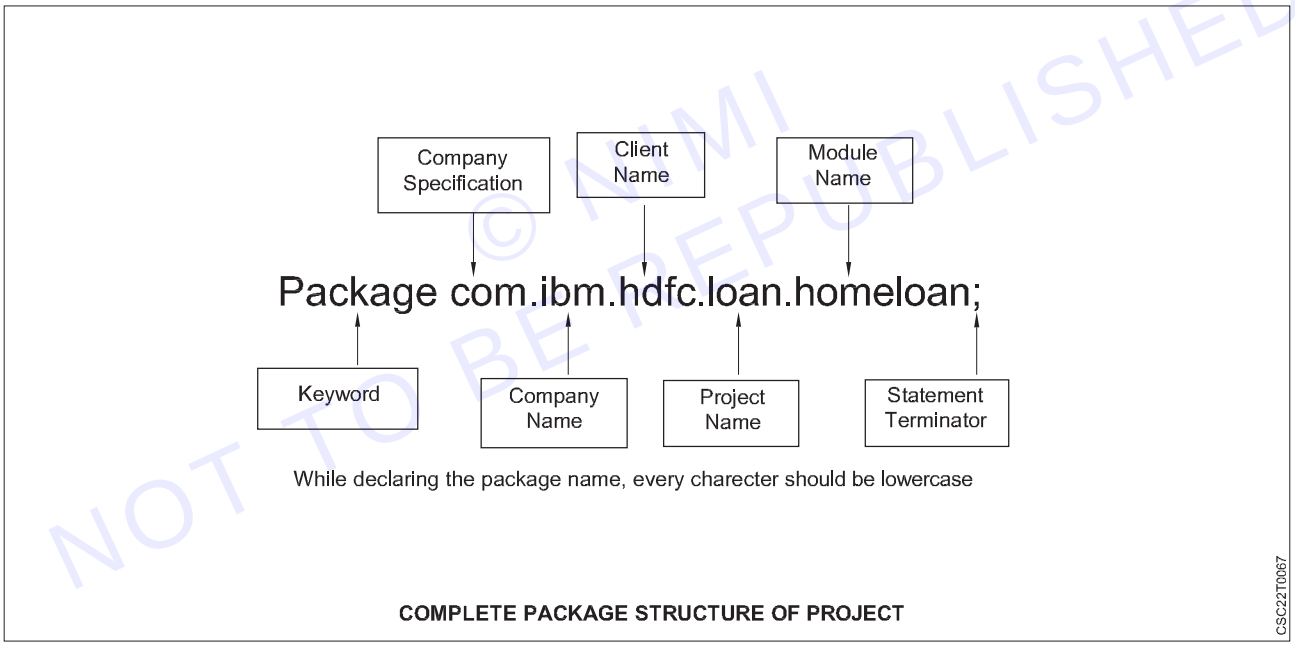
```
package myPackage;
public class A {
// class body
}
```

In this example, myPackage is the name of package. The statement “package myPackage;” signifies that the class “A” belongs to the “myPackage” package.

Naming Convention for User-defined Package in Realtime Project

While developing your project, you must follow some naming conventions regarding packages declaration. Let’s take an example to understand the convention.

Look at the below a complete package structure of the project.



COMPLETE PACKAGE STRUCTURE OF PROJECT

LESSON 116 - 119 : Abstract Windowing Tool Kit

Abstract Windowing Toolkit

The Abstract Windowing Toolkit (AWT) is a library of Java GUI object classes that is included with the Java Development Kit from Sun Microsystems. The AWT handles common interface elements for windowing environments including Windows.

The AWT contains the following set of GUI components:

Button

CheckBox

CheckBox Group (RadioList)

Choice (PopupList)

Label (StaticText)

List (ListBox)

Scroll Bar

Text Component (TextField)

Menu

Introduction to user interface and AWT components and containers

Introduction to the AWT:-

A description of Java's user interface toolkit

The Java programming language class library provides a user interface toolkit called the Abstract Windowing Toolkit, or the AWT. The AWT is both powerful and flexible. Newcomers, however, often find that its power is veiled. The class and method descriptions found in the distributed documentation provide little guidance for the new programmer. Furthermore, the available examples often leave many important questions unanswered. Of course, newcomers should expect some difficulty. Effective graphical user interfaces are inherently challenging to design and implement, and the sometimes complicated interactions between classes in the AWT only make this task more complex. However, with proper guidance, the creation of a graphical user interface using the AWT is not only possible, but relatively straightforward.

This article covers some of the philosophy behind the AWT and addresses the practical concern of how to create a simple user interface for an applet or application.

What is a user interface:-

The user interface is that part of a program that interacts with the user of the program. User interfaces take many forms. These forms range in complexity from simple command-line interfaces to the point-and-click graphical user interfaces provided by many modern applications.

[Also on InfoWorld: Why software engineering estimates are garbage]

At the lowest level, the operating system transmits information from the mouse and keyboard to the program as input, and provides pixels for program output. The AWT was designed so that programmers don't have worry about the details of tracking the mouse or reading the keyboard, nor attend to the details of writing to the screen. The AWT provides a well-designed object-oriented interface to these low-level services and resources.

Because the Java programming language is platform-independent, the AWT must also be platform-independent. The AWT was designed to provide a common set of tools for graphical user interface design that work on a variety of platforms. The user interface elements provided by the AWT are implemented using each platform's native GUI toolkit, thereby preserving the look and feel of each platform. This is one of the AWT's strongest points. The disadvantage of such an approach is the fact that a graphical user interface designed on one platform may look different when displayed on another platform.

Components and containers:-

A graphical user interface is built of graphical elements called components. Typical components include such items as buttons, scrollbars, and text fields. Components allow the user to interact with the program and provide the user with visual feedback about the state of the program. In the AWT, all user interface components are instances of class Component or one of its subtypes.

Components do not stand alone, but rather are found within containers. Containers contain and control the layout of components. Containers are themselves components, and can thus be placed inside other containers. In the AWT, all containers are instances of class Container or one of its subtypes.

Spatially, components must fit completely within the container that contains them. This nesting of components (including containers) into containers creates a tree of elements, starting with the container at the root of the tree and expanding out to the leaves, which are components such as buttons.

Types of containers:-

The AWT provides four container classes. They are class Window and its two subtypes -- class Frame and class Dialog -- as well as the Panel class. In addition to the containers provided by the AWT, the Applet class is a container -- it is a subtype of the Panel class and can therefore hold components. Brief descriptions of each container class provided by the AWT are provided below.

Window A top-level display surface (a window). An instance of the Window class is not attached to nor embedded within another container. An instance of the Window class has no border and no title.

Frame A top-level display surface (a window) with a border and title. An instance of the Frame class may have a menu bar. It is otherwise very much like an instance of the Window class.

Dialog A top-level display surface (a window) with a border and title. An instance of the Dialog class cannot exist without an associated instance of the Frame class.

Panel A generic container for holding components. An instance of the Panel class provides a container to which to add components.

Creating a container

Before adding the components that make up a user interface, the programmer must create a container. When building an application, the programmer must first create an instance of class Window or class Frame. When building an applet, a frame (the browser window) already exists. Since the Applet class is a subtype of the Panel class, the programmer can add the components to the instance of the Applet class itself.

The code in Listing 1 creates an empty frame. The title of the frame (“Example 1”) is set in the call to the constructor. A frame is initially invisible and must be made visible by invoking its show() method.

```
import java.awt.*;
public class Example1
{
  public static void main(String [] args)
  {
     Frame f = new Frame(“Example 1”);
     f.show();
  }
}
```

Adding components to a container

To be useful, a user interface must consist of more than just a container -- it must contain components. Components are added to containers via a container’s add() method. There are three basic forms of the add() method. The method to use depends on the container’s layout manager (see the section titled Component layout).

The code in Listing 4 adds the creation of two buttons to the code presented in Listing 3. The creation is performed in the init() method because it is automatically called during applet initialization. Therefore, no matter how the program is started, the buttons are created, because init() is called by either the browser or by the main() method. Figure 6 contains the resulting applet.

```
import java.awt.*;
public class Example3 extends java.applet.Applet
{
  public void init()
  {
      add(new Button("One"));
      add(new Button("Two"));
  }
  public Dimension preferredSize()
  {
      return new Dimension(200, 100);
  }
  public static void main(String [] args)
  {
      Frame f = new Frame("Example 3");
      Example3 ex = new Example3();
      ex.init();
      f.add("Center", ex);
      f.pack();
      f.show();
  }
}
```


Introduction to AWT UI controls, hierarchy and their features

Introduction to AWT UI controls

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc

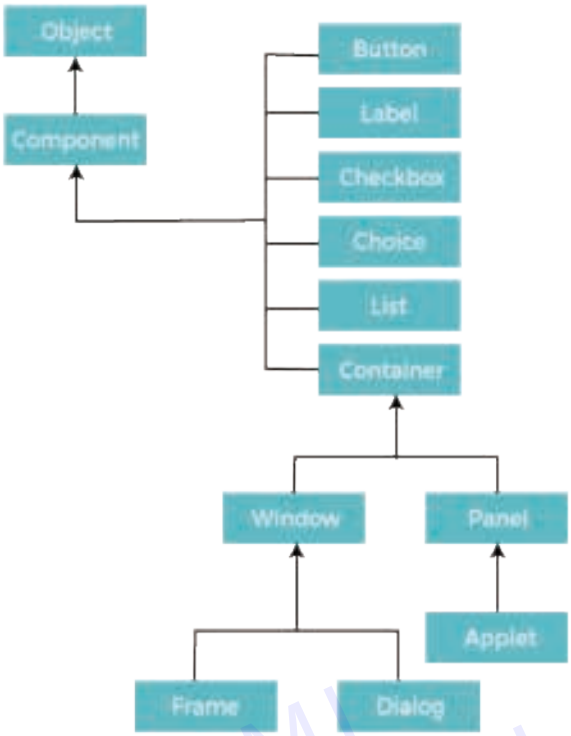
Java AWT calls the native platform calls the native platform (operating systems) subroutine for creating API components like TextField, ChechBox, button, etc.

For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

Java AWT Hierarchy:-

The hierarchy of Java AWT classes are given below.



Introduction to event Handling

What is an Event?

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event

The events can be broadly classified into two categories:

Foreground Events - Those events which require the direct interaction of user. They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.

Background Events - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model.

The Delegation Event Model has the following key participants namely:

Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for source object.

Listener - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns.

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model, Listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

Steps involved in event handling

The User clicks the button and the event is generated.

Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.

Event object is forwarded to the method of registered listener class.

the method is now get executed and returns.

Points to remember about listener:-

In order to design a listener class we have to develop some listener interfaces. These Listener interfaces forecast some public abstract callback methods which must be implemented by the listener class.

If you do not implement the any if the predefined interfaces then your class can not act as a listener class for a source object.

Callback Methods:-

These are the methods that are provided by API provider and are defined by the application programmer and invoked by the application developer. Here the callback methods represents an event method. In response to an event java are will fire callback method. All such callback methods are provided in listener interfaces.

If a component wants some listener will listen to it's events the source must register itself to the listener.

Event Handling Example:-

Create the following java program using any editor of your choice in say D:/ > AWT > com > tutorialspoint > gui >

AwtControlDemo.java

```
package com.tutorialspoint.gui;
import java.awt.*;
import java.awt.event.*;
public class AwtControlDemo {
   private Frame mainFrame;
   private Label headerLabel;
   private Label statusLabel;
   private Panel controlPanel;
   public AwtControlDemo(){
      prepareGUI();
   }
   public static void main(String[] args){
      AwtControlDemo  awtControlDemo = new AwtControlDemo();
      awtControlDemo.showEventDemo();
   }
   private void prepareGUI(){
```

```
    mainFrame = new Frame("Java AWT Examples");
    mainFrame.setSize(400,400);
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
    }
    });
    headerLabel = new Label();
    headerLabel.setAlignment(Label.CENTER);
    statusLabel = new Label();
    statusLabel.setAlignment(Label.CENTER);
    statusLabel.setSize(350,100);
    controlPanel = new Panel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}
private void showEventDemo(){
    headerLabel.setText("Control in action: Button");
    Button okButton = new Button("OK");
    Button submitButton = new Button("Submit");
    Button cancelButton = new Button("Cancel");
  okButton.setActionCommand("OK");
  submitButton.setActionCommand("Submit");
  cancelButton.setActionCommand("Cancel");
    okButton.addActionListener(new ButtonClickListener());
    submitButton.addActionListener(new ButtonClickListener());
    cancelButton.addActionListener(new ButtonClickListener());

    controlPanel.add(okButton);
    controlPanel.add(submitButton);
    controlPanel.add(cancelButton);

    mainFrame.setVisible(true);
}
private class ButtonClickListener implements ActionListener{
```

```
        public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        if( command.equals( “OK” ))  {
                statusLabel.setText(“Ok Button clicked.”);
        }
        else if( command.equals( “Submit” ) )  {
                statusLabel.setText(“Submit Button clicked.”);
        }
        else  {
              statusLabel.setText(“Cancel Button clicked.”);
        }
        }
    }
}
```

Introduction to Event Handling Classes

EventObject class:-

It is the root class from which all event state objects shall be derived. All Events are constructed with a reference to the object, the source, that is logically deemed to be the object upon which the Event in question initially occurred upon. This class is defined in java.util package.

Class declaration

Following is the declaration for java.util.EventObject class:

public class EventObject extends Object implements Serializable Field Following are the fields for java.util. EventObject class:

protected Object source -- The object on which the Event initially occurred.

Class constructors

Constructor & Description

EventObject(Object source)

Constructs a prototypical Event.

Class methods

Method & Description

Object getSource()

The object on which the Event initially occurred.

String toString()

Returns a String representation of this EventObject.

Methods inherited

This class inherits methods from the following classes:

java.lang.Object

AWT Event Classes:

Following is the list of commonly used event classes.

Control & Description

AWTEvent

It is the root event class for all AWT events. This class and its subclasses supercede the original java.awt. Event class.

ActionEvent

The ActionEvent is generated when button is clicked or the item of a list is double clicked.

InputEvent

The InputEvent class is root event class for all component-level input events.

KeyEvent

On entering the character the Key event is generated.

MouseEvent

This event indicates a mouse action occurred in a component.

Introduction to event listener interfaces

The Event listener represent the interfaces responsible to handle events. Java provides us various Event listener classes but we will discuss those which are more frequently used. Every method of an event listener method has a single argument as an object which is subclass of EventObject class. For example, mouse event listener methods will accept instance of MouseEvent, where MouseEvent derives from EventObject.

Event Listener interface:-

It is a marker interface which every listener interface has to extend.This class is defined in java.util package.

Class declaration:-

Following is the declaration for java.util.EventListener interface:

public interface EventListener

AWT Event Listener Interfaces:

Following is the list of commonly used event listeners.

Control & Description

1 ActionListener

This interface is used for receiving the action events.

2 ComponentListener

This interface is used for receiving the component events.

3 ItemListener

This interface is used for receiving the item events.

4 KeyListener

This interface is used for receiving the key events.

5 MouseListener

This interface is used for receiving the mouse events.

6 TextListener

This interface is used for receiving the text events.

7 WindowListener

This interface is used for receiving the window events.

Introduction to AWT Layouts

Introduction

Layout means the arrangement of components within the container. In other way we can say that placing the components at a particular position within the container. The task of layouting the controls is done automatically by the Layout Manager.

Layout Manager:-

The layout manager automatically positions all the components within the container. If we do not use layout manager then also the components are positioned by the default layout manager. It is possible to layout the controls by hand but it becomes very difficult because of the following two reasons.

1 It is very tedious to handle a large number of controls within the container.

2 Oftenly the width and height information of a component is not given when we need to arrange them

Java provide us with various layout manager to position the controls. The properties like size,shape and arrangement varies from one layout manager to other layout manager. When the size of the applet or the application window changes the size, shape and arrangement of the components also changes in response i.e. the layout managers adapt to the dimensions of appletviewer or the application window.

The layout manager is associated with every Container object. Each layout manager is an object of the class that implements the LayoutManager interface.

Following are the interfaces defining functionalities of Layout Managers.

Interface & Description

1 LayoutManager

The LayoutManager interface declares those methods which need to be implemented by the class whose object will act as a layout manager.

2 LayoutManager2

The LayoutManager2 is the sub-interface of the LayoutManager.This interface is for those classes that know how to layout containers based on layout constraint object.

AWT Layout Manager Classes:

Following is the list of commonly used controls while designed GUI using AWT.

LayoutManager & Description

1 BorderLayout

The borderlayout arranges the components to fit in the five regions: east, west, north, south and center.

2 CardLayout

The CardLayout object treats each component in the container as a card. Only one card is visible at a time.

3 FlowLayout

The FlowLayout is the default layout.It layouts the components in a directional flow.

4 GridLayout

The GridLayout manages the components in form of a rectangular grid.

5 GridBagLayout

This is the most flexible layout manager class.The object of GridBagLayout aligns the component vertically,horizontally or along their baseline without requiring the components of same size.

MODULE 7 : Programming language (Python)

LESSON 120 - 137 : Programming language (Python)

Objectives

At the end of this lesson, you will be able to:

- build basic programming using python
- understand basic concepts of python programming & to develop the basic applications
- solve complex computing problems by using built in modules.

Introduction To Python History

Python is a popular programming language. It is designed by **Guido van Rossum in 1991**. Nowadays, Python is one of the most popular and widely used programming language all over the world. It is a high-level programming language. It's syntax allows programmers to express concepts in fewer lines of code. It is used for set of tasks including console based, GUI based, web programming and data analysis. Python is a easy to learn and simple programming language so even if you are new to programming, you can learn python without facing any problems.

Story behind Python's invention

In the late 1980s, working on Python started by Guido Van Rossum. He began doing its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) which is situated in the Netherlands. It was started as a hobby project because he was looking for an interesting project to keep him occupied during Christmas.

From ABC language to Python

The success of Python programming language came from **ABC Programming Language**, which was operating on Amoeba Operating System and had the feature of exception handling. Guido Van Rossum had already helped create ABC language earlier in his career and had seen some issues with ABC language but liked most of the features. While creating Python, he had taken the syntax of ABC language, and some of its good features also. Initially it came with a lot of complaints too, so he fixed those issues completely and created a good scripting language that had removed all the flaws.

Named Python after a TV Show

The inspiration for the name came from the **BBC's TV Show – 'Monty Python's Flying Circus'**, as Guido Van Rossum was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! For quite some time he worked for Google, but currently, he is working at Dropbox.

Evolution of Python

The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released, it had more than enough capability to provide classes with inheritance, several core data types of exception handling and functions.

Features, Setting Up Path Basic Syntax, Comments, Variable

Python is a feature rich high-level, interpreted, interactive and object-oriented scripting language.

Features of Python

There are some important features of Python programming language...

1 Easy to Learn

This is one of the most important reasons for the popularity of Python. Python has a limited set of keywords. Its features such as simple syntax, usage of indentation to avoid clutter of curly brackets and dynamic typing that doesn't necessitate prior declaration of variable help a beginner to learn Python quickly and easily.

2 Interpreter Based

Any Programming languages is either compiler based or interpreter based. Python is an interpreter based language. The interpreter takes one instruction from the source code at a time, translates it into machine code and executes it. Instructions before the first occurrence of error are executed. With this feature, it is easier to debug the program and thus proves useful for the beginner level programmer to gain confidence gradually. Python therefore is a beginner-friendly language.

3 Interactive

Python prompt >>> works on the principle of **REPL** (Read – Evaluate – Print – Loop). You can type any valid Python expression here and press Enter. Python interpreter immediately returns the response and the prompt comes back to read the next expression like this >>>.

Example 1:

```
>>> 2*3+1
7
```

Example 2:

```
>>> print ("Hello World")
Hello World
```

The interactive mode is especially useful to get familiar with a library and test out its functionality. You can try out small code snippets in interactive mode before writing a program.

4 MultiParadigm

Python is a completely object-oriented language. Everything in a Python program is an object. However, Python conveniently encapsulates its object orientation to be used as an imperative or procedural language-such as C. Python also provides certain functionality that resembles functional programming. Moreover, certain third-party tools have been developed to support other programming paradigms such as aspect-oriented and logic programming.

5 Standard Library

Even though it has a very few keywords (only Thirty Five), Python software is distributed with a standard library made of large number of modules and packages. Thus Python has out of box support for programming needs such as serialization, data compression, internet data handling, and many more. Python is known for its batteries included approach.

6 Open Source and Cross Platform

Python's standard distribution can be downloaded from https://www.python.org/downloads/ without any restrictions. In addition, the source code is also freely available, which is why it comes under open source category.

Python is a cross-platform language. Pre-compiled binaries are available for use on various operating system platforms such as Windows, Linux, Mac OS, Android OS. A Python program can be easily ported from one OS platform to other.

7 GUI Applications

Python's standard distribution has an excellent graphics library called TKinter. It is a Python port for the vastly popular GUI toolkit called TCL/Tk. You can build attractive user-friendly GUI applications in Python. Examples are PyQt, WxWidgets, PySimpleGUI etc.

8 Database Connectivity

Almost any type of database can be used as a backend with the Python application. DB-API is a set of specifications for database driver software to let Python communicate with a relational database. With many third party libraries, Python can also work with NoSQL databases such as MongoDB.

9 Extensible

The term extensibility implies the ability to add new features or modify existing features. As stated earlier, CPython (which is Python's reference implementation) is written in C. Hence one can easily write modules/ libraries in C and incorporate them in the standard library. There are other implementations of Python such as Jython (written in Java) and IPython (written in C#). Hence, it is possible to write and merge new functionality in these implementations with Java and C# respectively.

10 Active Developer Community

As a result of Python's popularity and open-source nature, a large number of Python developers often interact with online forums and conferences. Python Software Foundation also has a significant member base, involved in the organization's mission to "Promote, Protect, and Advance the Python Programming Language". Major IT companies Google, Microsoft, and Meta contribute immensely by preparing documentation and other resources.

More Features of Python

Apart from the above-mentioned features, Python has another big list of good features, few are listed below...

1 It supports functional and structured programming methods as well as OOP.

2 It can be used as a scripting language or can be compiled to byte-code for building large applications.

3 It provides very high-level dynamic data types and supports dynamic type checking.

4 It supports automatic garbage collection.

5 It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Setting Up Path For Python

1 Get Python Installer from python.org.

2 Get the installer and an installation window will appear.

3 Press the "Add Python X.X to your PATH" option and install the python.

This way you can set up a default path without any headache.

BASIC SYNTAX

The Python syntax defines a set of rules that are used to create a Python Program. The Python Programming Language Syntax has many similarities to Perl, C, and Java Programming Languages. However, there are some definite differences between the languages.

Modes of Python Programming

There are two different **modes of Python Programming...**

1 Interactive Mode Programming

2 Script Mode Programming

Let's execute a Python program to print "Hello, World!" in both modes...

1 Interactive Mode Programming

>>>

Here >>> denotes a Python Command Prompt where you can type your commands. Let's type the following text at the Python prompt and press the Enter...

>>> print ("Hello, World!")

Result-

Hello, World!

2 Script Mode Programming

We can write a simple Python program in a script which is simple text file. Python files have extension .py. Type the following source code in a test.py file...

print (“Hello, World!”)

We assume that you have Python interpreter path set in PATH variable. Now, let’s try to run this program as follows...

test.py

Result-

Hello, World!

Python Comments

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

Creating a Comment

Comments starts with a #, and Python will ignore them.

Example:

#This is a comment

print(“Hello, World!”)

Single line Comment

Comments can be placed at the end of a line, and Python will ignore the rest of the line.

Example:

print(“Hello, World!”) #This is a comment

A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code.

Example:

#print(“Hello, World!”)

print(“Cheers, Mate!”)

Multiline Comments

Python does not really have a syntax for multiline comments.

To add a multiline comment you could insert a # for each line.

Example:

#This is a comment

#written in

#more than just one line

print(“Hello, World!”)

Or, not quite as intended, you can use a multiline string.

Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it.

Example:

“””

This is a comment

written in

more than just one line

“””

print(“Hello, World!”)

As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multi-line comment.

Variable

The word variable suggests that it is something that varies or changes.

A variable in Python is a container or storage box of which its contents can change depending on what we put into the container or storage box.

It is a storage holder that has a name or label pasted outside it and it holds something of value inside it.

We can imagine a storage box with the label course_name and inside this storage box it contains the “Python Course”.

The container’s label should match whatever is contained inside the container.

The container’s label should clearly and appropriately describe what will be stored inside the container.

80% of programming codes consist of containers, storage boxes, or variables because computers are so good at storing things that change.

Creating Variables

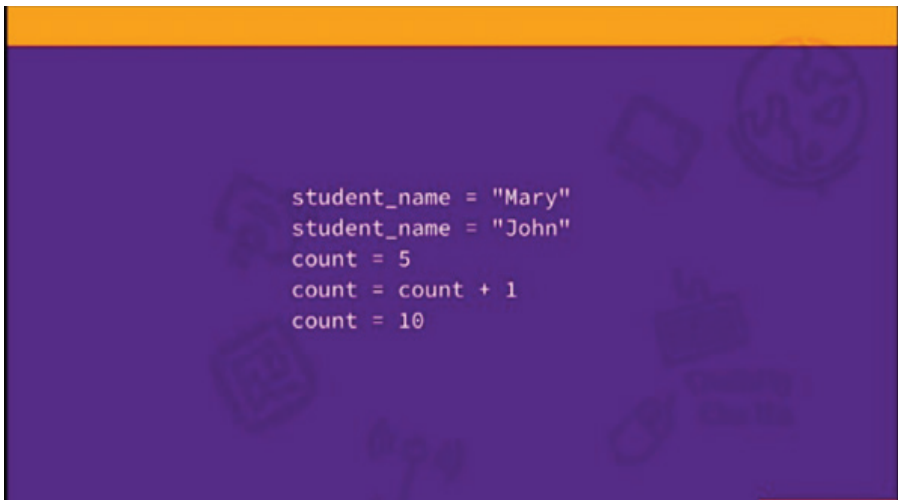
When we create a variable, we give it a name it is like when we use a storage box, the first thing we do is to label it with a name.

We use a variable name to reference data such that we can access the data inside the variable by referencing the variable’s name.

The variable’s name is fixed but its content or value can change it is like now we put something into the storage box and then further on we can take it out and put something else in.

Assigning Variables

To assign a value to a variable name, we use the assignment operator.



The = sign is called the assignment operator.

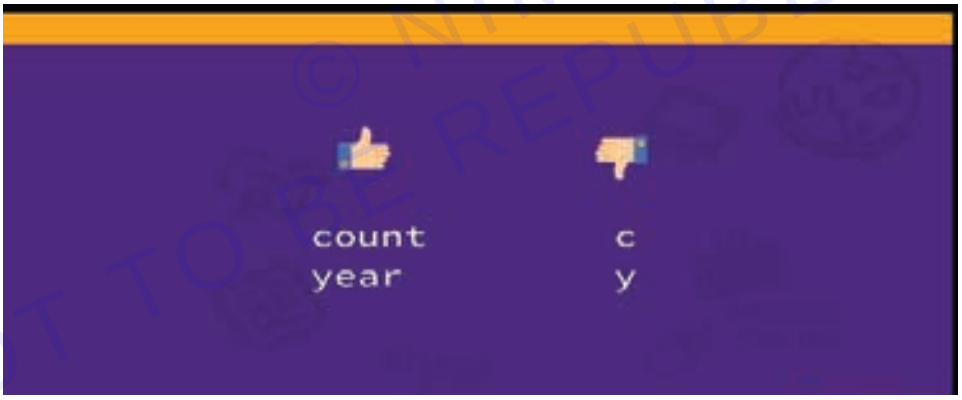
The variable name student_name has its value changed from “Mary” to “John”.

The variable name count has its value changed from 5 to 6 by adding 1 to it and then changed again from 6 to 10 by taking out the 6 and putting in the 10.

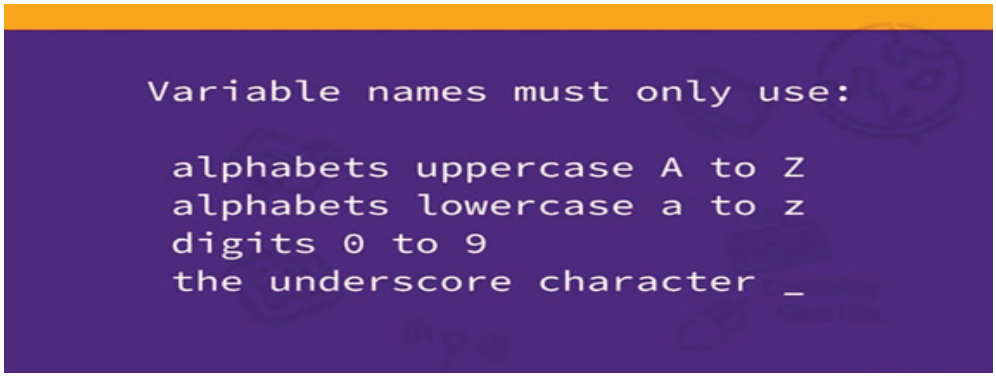
Naming Variables

Select meaningful variable names that represent what they are for.

Try to use the variable name count instead of just c, and year instead of just y.



Variable name must only use:



Examples of valid variable names are:

student_name

_student_name

Student_name

studentname

studentName

studentName1

STUDENTNAME_

Examples of invalid variable names are:

1studentName

student@name

student name

student-name

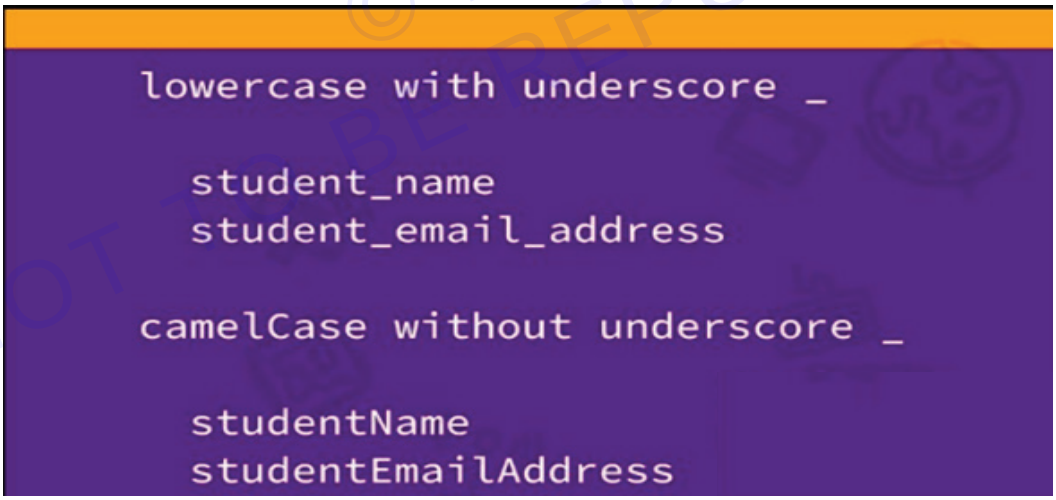
1studentName is invalid because variable names cannot start with a digit.

student@name is invalid because variable names cannot have any special characters except the underscore character.

student name is invalid because variable names cannot have any space characters.

student-name is invalid because hyphens are not allowed.

The general naming convention in Python when it comes to variable names, is to use lowercase alphabets and separate each word using the underscore character_.



The camelCase

However, many Python software developers use the camelCase instead of the lowercase separated by the underscore character.

camelCase refers to variable names starting with a lowercase character, and instead of using the underscore character _ to separate 2 words, it starts the next word with an uppercase character.

It is important to note that regardless of whether we use lowercase with underscore character _ or camelCase, we should be consistent and adopt the same naming convention throughout our program.

And variable names are case- sensitive:

student_name is not the same as STUDENT_NAME

Python Keywords

All Python keywords cannot be used for variable names:

Python is a case-sensitive language and Python keywords are case- sensitive.

Please take note that these 3 keywords - False, None, True all of them start with an uppercase character and not a lowercase character:

False

None

True

Variable names are also case-sensitive:

False is not the same as false

False cannot be used as a variable name

false can be used as a variable name

The Python keyword False, which starts with an uppercase F, is not the same as lowercase false.

Python keyword False cannot be used as a variable name, but lowercase false can be used as a variable name.

Tracing Variables

As the value of a variable changes throughout our program because new values are constantly assigned to it, we can use the print() function to trace the value of our variables.

The print() function can be used for debugging purposes whereby we need to know what went wrong in our program - and one way to find out is to trace how the value of our variables changes in the course of our program.

Data Types of Python

Data types are the classification or categorization of data items. Python supports the following built-in data types.

Scalar Types

1 **int:** Positive or negative whole numbers (without a fractional part) e.g. -10, 10, 456, 4654654.

2 **float:** Any real number with a floating-point representation in which a fractional component is denoted by a decimal symbol or scientific notation e.g. 1.23, 3.4556789e2.

3 **complex:** A number with a real and imaginary component represented as x + 2y.

4 **bool:** Data with one of two built-in values True or False. Notice that 'T' and 'F' are capital. true and false are not valid booleans and Python will throw an error for them.

None: The None represents the null object in Python. A None is returned by functions that don't explicitly return a value.

Sequence Type

A sequence is an ordered collection of similar or different data types. Python has the following built-in sequence data types...

1 **String:** A string value is a collection of one or more characters put in single, double or triple quotes.

2 **List:** A list object is an ordered collection of one or more data items, not necessarily of the same type, put in square brackets.

3 **Tuple:** A Tuple object is an ordered collection of one or more data items, not necessarily of the same type, put in parentheses.

Mapping Type

1 **Dictionary:** A dictionary Dict() object is an unordered collection of data in a key:value pair form. A collection of such pairs is enclosed in curly brackets. For example: {1:"Steve", 2:"Bill", 3:"Ram", 4: "Farha"}

Set Types

1 **set:** Set is mutable, unordered collection of distinct hashable objects. The set is a Python implementation of the set in Mathematics. A set object has suitable methods to perform mathematical set operations like union, intersection, difference, etc.

2 **frozenset:** Frozenset is immutable version of set whose elements are added from other iterables.

Mutable and Immutable Types

Data objects of the above types are stored in a computer's memory for processing. Some of these values can be modified during processing, but contents of others can't be altered once they are created in the memory.

Numbers, strings, and Tuples are immutable, which means their contents can't be altered after creation.

On the other hand, items in a List or Dictionary object can be modified. It is possible to add, delete, insert, and rearrange items in a list or dictionary. Hence, they are mutable objects.

Casting, string, Boolean

Casting

Casting is a process of converting a variable's data type into another data type. If you want to specify the data type of a variable, this can be done with casting.

Example:

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Python String

Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes.

In Python, strings can be created by enclosing the character or the sequence of characters in the quotes. Python allows us to use single quotes, double quotes, or triple quotes to create the string.

Creating a string in Python

Syntax:

str = “Hi Python !”

Here, if we check the type of the variable str using a Python script

print(type(str)), then it will print a string (str).

In Python, strings are treated as the sequence of characters, which means that Python doesn’t support the character data-type; instead, a single character written as ‘p’ is treated as the string of length 1.

#Using single quotes

str1 = ‘Hello Python’

print(str1)

#Using double quotes

str2 = “Hello Python”

print(str2)

#Using triple quotes

str3 = ‘’’Triple quotes are generally used for represent the multiline or docstring’’’

print(str3)

Output:

Hello Python

Hello Python

Triple quotes are generally used for represent the multiline or docstring.

Boolean

Booleans represent one of two values: True or False.

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

print(10 > 9)

print(10 == 9)

print(10 < 9)

When you run a condition in an if statement, Python returns True or False:

Example:

Print a message based on whether the condition is True or False:

```
a = 200
b = 33
if b > a:
   print(“b is greater than a”)
```

else:

print(“b is not greater than a”)

Evaluate Values and Variables

The bool() function allows you to evaluate any value, and give you True or False in return,

Example:

Evaluate a string and a number:

print(bool(“Hello”))

print(bool(15))

Example:

Evaluate two variables:

x = “Hello”

y = 15

print(bool(x))

print(bool(y))

ALSO READ:

Python Operators

Operators are special symbols that perform operations on variables and values.

Example:

print(5 + 6) # 11

Here, + is an operator that adds two numbers: 5 and 6.

Types of Python Operators

There are 7 types of Python operators...

1 Arithmetic Operators

2 Assignment Operators

3 Comparison Operators

4 Logical Operators

5 Bitwise Operators

6 Identity Operators

7 Membership Operators

1 Python Arithmetic Operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

Example:

sub = 10 - 5 # 5

Here, - is an arithmetic operator that subtracts two values or variables.

2 Python Assignment Operators

Assignment operators are used to assign values to variables. For example,

assign 5 to x

var x = 5

Here, = is an assignment operator that assigns 5 to x.

3 Python Comparison Operators

Comparison operators compare two values/variables and return a boolean result: True or False.

Example:

```
a = 5
b =2
print (a > b)    # True
```

4 Python Logical Operators

Logical operators are used to check whether an expression is True or False. They are used in decision-making. For example,

```
a = 5
b = 6
print((a > 2) and (b >= 6))    # True
```

5 Python Bitwise Operators

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name.

Example:

2 is 10 in binary, and 7 is 111.

6 Identity Operators

Identity operators are used for locating the memory unit of the objects, especially when both objects have the same name and can be differentiated only using their memory location.

Types of Identity Operators

There are two types of identity operators...

1 Is Operator

2 Is Not Operator

Is Operator

Is operator is used for comparing if the objects are in the same location while returning 'true' or 'false' values as a result.

Example:

```
m = 70
n = 70
if ( m is n ):
   print("Result: m and n have same identity")
else:
   print("Result: m and n do not have same identity")
```

Output: m and n have same identity

Is Not Operator

Is Not operator works in the opposite way of is operator? This returns true if the memory location of two objects is not the same. This returns False if the memory location of two objects is the same.

Example:

```
m = 70
n = 70
if ( m is not n ):
   print("Result: m and n have same identity")
else:
   print("Result: m and n do not have same identity")
```

Output: m and n do not have same identity

7 Membership Operators

We use membership operators to check whether a value or variable exists in a sequence (string, list, tuples, sets, dictionary) or not. Python has two membership operators: in and not in. Both return a Boolean result. The result of in operator is opposite to that of not in operator.

Example:

```
var = "TotalSach"
a = "s"
b = "tal"
c = "ac"
d = "Al"
print (a, "in", var, ":", a in var)
print (b, "not in", var, ":", b not in var)
print (c, "in", var, ":", c in var)
print (d, "not in", var, ":", d not in var)
```

Output:

```
s in TotalSach : True
tal not in TotalSach : False
ac in TotalSach : True
Al not in TotalSach : True
```


Conditional Statements

There are situations in real life when we need to make some decisions and based on these decisions, we decide what we should do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code.

Conditional statements in Python languages decide the direction(Control Flow) of the flow of program execution.

Types of Statements in Python

There are 4 types of Statements in Python...

1 The if statement

2 The if-else statement

3 The nested-if statement

4 The if-elif-else ladder

if statement

The if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not.

Syntax:

```
if condition:
  # Statements to execute if
  # condition is true
```

Here, the condition after evaluation will be either true or false. if the statement accepts boolean values-if the value is true then it will execute the block of statements below it otherwise not.

As we know, python uses indentation to identify a block. So the block under an if statement will be identified as shown in the below example:

```
if condition:
  statement1
statement2
# Here if the condition is true, if block
# will consider only statement1 to be inside
# its block.
```

Example:

As the condition present in the if statement is false. So, the block below the if statement is executed.

```
# python program to illustrate If else statement
#!/usr/bin/python
i = 20
if (i < 15):
    print("i is smaller than 15")
    print("i'm in if Block")
else:
    print("i is greater than 15")
    print("i'm in else Block")
print("i'm not in if and not in else Block")
```

Output:

```
i is greater than 15
i'm in else Block
i'm not in if and not in else Block
```

if-else Statement

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But if we want to do something else if the condition is false, we can use the else statement with the if statement to execute a block of code when the if condition is false.

Syntax:

```
if (condition):
    # Executes this block if
```

```
    # condition is true
else:
    # Executes this block if
    # condition is false
```

Example:

The block of code following the else statement is executed as the condition present in the if statement is false after calling the statement which is not in the block(without spaces).

```
# Explicit function
def digitSum(n):
    dsum = 0
    for ele in str(n):
        dsum += int(ele)
    return dsum

# Initializing list
List = [367, 111, 562, 945, 6726, 873]

# Using the function on odd elements of the list
newList = [digitSum(i) for i in List if i & 1]

# Displaying new list
print(newList)
```

Output :

[16, 3, 18, 18]

Nested-if Statement

A nested if is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement.

Yes, Python allows us to nest if statements within if statements. i.e., we can place an if statement inside another if statement.

Syntax:

```
if (condition1):
    # Executes when condition1 is true
    if (condition2):
        # Executes when condition2 is true
    # if Block is end here
# if Block is end here
```

Example:

In this example, we are showing nested if conditions in the code, All the If conditions will be executed one by one.

```
# python program to illustrate nested If statement
i = 10
if (i == 10):
```

```
#  First if statement
if (i < 15):
    print("i is smaller than 15")

# Nested - if statement
# Will only be executed if statement above
# it is true
if (i < 12):
    print("i is smaller than 12 too")
else:
    print("i is greater than 15")
```

Output:

i is smaller than 15

i is smaller than 12 too

if-elif-else Ladder

Here, a user can decide among multiple options. The if statements are executed from the top down.

As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final "else" statement will be executed.

Syntax:

```
if (condition):
    statement
elif (condition):
    statement
else:
    statement
```

Example:

In the example, we are showing single if condition, multiple elif conditions, and single else condition.

```
# Python program to illustrate if-elif-else ladder
#!/usr/bin/python

i = 20
if (i == 10):
    print("i is 10")
elif (i == 15):
    print("i is 15")
elif (i == 20):
    print("i is 20")
else:
    print("i is not present")
```

Output:

i is 20

Looping

The following loops are available in Python to fulfil the looping needs. Python offers 3 choices for running the loops. The basic functionality of all the techniques is the same, although the syntax and the amount of time required for checking the condition differ.

We can run a single statement or set of statements repeatedly using a loop command.

The following sorts of loops are available in the Python programming language.

S. No.	Name of the loop	Loop Type & Description
1	While loop	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2	For loop	This type of loop executes a code block multiple times and abbreviates the code that manages the loop variable.
3	Nested loops	We can iterate a loop inside another loop.

Loop Control Statements

Statements used to control loops and change the course of iteration are called control statements. All the objects produced within the local scope of the loop are deleted when execution is completed.

Python provides the following control statements. We will discuss them later in detail.

Let us quickly go over the definitions of these loop control statements.

S. No.	Name of the control statement	Loop Type & Description
1	Break statement	This command terminates the loop's execution and transfers the program's control to the statement next to the loop.
2	Continue statement	This command skips the current iteration of the loop. The statements following the continue statement are not executed once the Python interpreter reaches the continue statement.
3	Pass statement	The pass statement is used when a statement is syntactically necessary, but no code is to be executed.

The for Loop

Python's for loop is designed to repeatedly execute a code block while iterating through a list, tuple, dictionary, or other iterable objects of Python. The process of traversing a sequence is known as iteration.

Syntax of the for Loop

```
1  For value in sequence:
2  { code block }
```

In this case, the variable value is used to hold the value of every item present in the sequence before the iteration begins until this particular iteration is completed.

Loop iterates until the final item of the sequence are reached.

Code

```
1  # Python program to show how the for loop works
2
3  # Creating a sequence which is a tuple of numbers
4  numbers = [4, 2, 6, 7, 3, 5, 8, 10, 6, 1, 9, 2]
5
6  # variable to store the square of the number
```

```
7   square = 0
8
9   # Creating an empty list
10 squares = []
11
12 # Creating a for loop
13 for value in numbers:
14 square = value ** 2
15 squares.append(square)
16 print("The list of squares is", squares)
```

Output:

The list of squares is [16, 4, 36, 49, 9, 25, 64, 100, 36, 1, 81, 4]

Control Statements, String Manipulation, Lists, Tuple, sets

Python Loop Control Statements

Loop control statements change execution from their normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements.

Python Continue

It returns the control to the beginning of the loop.

- Python3

```
# Prints all letters except 'e' and 's'
for letter in 'geeksforgeeks':
    if letter == 'e' or letter == 's':
        continue
    print('Current Letter :', letter)
```

Output:

Current Letter : g
Current Letter : k
Current Letter : f
Current Letter : o
Current Letter : r
Current Letter : g
Current Letter : k

Python Break

It brings control out of the loop.

- Python3

```
for letter in 'geeksforgeeks':
    # break the loop as soon it sees 'e'
    # or 's'
    if letter == 'e' or letter == 's':
        break
print('Current Letter :', letter)
```

Output:

Current Letter : e

Python Pass

We use pass statements to write empty loops. Pass is also used for empty control statements, functions, and classes.

- Python3

```
# An empty loop
for letter in 'geeksforgeeks':
    pass
print('Last Letter :', letter)
```

Output:

Last Letter :

Python String

Till now, we have discussed numbers as the standard data-types in Python. In this section of the tutorial, we will discuss the most popular data type in Python, i.e., string.

Python string is the collection of the characters surrounded by single quotes, double quotes, or triple quotes. The computer does not understand the characters; internally, it stores manipulated character as the combination of the 0's and 1's.

Each character is encoded in the ASCII or Unicode character. So we can say that Python strings are also called the collection of Unicode characters.

In Python, strings can be created by enclosing the character or the sequence of characters in the quotes. Python allows us to use single quotes, double quotes, or triple quotes to create the string.

Consider the following example in Python to create a string.

Syntax:

```
1  str = "Hi Python !"
```

Here, if we check the type of the variable str using a Python script

```
1  print(type(str)), then it will print a string (str).
```

In Python, strings are treated as the sequence of characters, which means that Python doesn't support the character data-type; instead, a single character written as 'p' is treated as the string of length 1.

Creating String in Python

We can create a string by enclosing the characters in single-quotes or double- quotes. Python also provides triple-quotes to represent the string, but it is generally used for multiline string or docstrings.

```
1  #Using single quotes
2  str1 = 'Hello Python'
3  print(str1)
4  #Using double quotes
5  str2 = "Hello Python"
6  print(str2)
7
8  #Using triple quotes
9  str3 = '''Triple quotes are generally used for
10 represent the multiline or
11 docstring'''
12 print(str3)
```

Output:

Hello Python

Hello Python

Triple quotes are generally used for represent the multiline or docstring.

Python List

In Python, the sequence of various data types is stored in a list. A list is a collection of different kinds of values or items. Since Python lists are mutable, we can change their elements after forming. The comma (,) and the square brackets [enclose the List's items] serve as separators.

Although six Python data types can hold sequences, the List is the most common and reliable form. A list, a type of sequence data, is used to store the collection of data. Tuples and Strings are two similar data formats for sequences.

Lists written in Python are identical to dynamically scaled arrays defined in other languages, such as Array List in Java and Vector in C++. A list is a collection of items separated by commas and denoted by the symbol [].

List Declaration

Code

```
1  # a simple list
2  list1 = [1, 2, "Python", "Program", 15.9]
3  list2 = ["Amy", "Ryan", "Henry", "Emma"]
4
5  # printing the list
6  print(list1)
7  print(list2)
8
9  # printing the type of list
10 print(type(list1))
11 print(type(list2))
```

Output:

[1, 2, 'Python', 'Program', 15.9]

['Amy', 'Ryan', 'Henry', 'Emma']

< class ' list ' >

< class ' list ' >

Characteristics of Lists

The characteristics of the List are as follows:

- The lists are in order.
- The list element can be accessed via the index.
- The mutable type of List is
- The rundowns are changeable sorts.
- The number of various elements can be stored in a list.

Ordered List Checking

Code

```
1  # example
2  a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6 ]
3  b = [ 1, 2, 5, "Ram", 3.50, "Rahul", 6 ]
4  a == b
```

Output:

False

Python Lists are just like dynamically sized arrays, declared in other languages (vector in C++ and ArrayList in Java). In simple language, a list is a collection of things, enclosed in [ ] and separated by commas.

The list is a sequence data type which is used to store the collection of data. Tuples and String are other types of sequence data types.

Example of list in Python

Here we are creating Python **List** using [].

- Python3

```
Var = ['NSTI', 'Greenery']
print(Var)
```

Output:

['NSTI', 'Greenery']

Python Lists are just like dynamically sized arrays, declared in other languages (vector in C++ and ArrayList in Java). In simple language, a list is a collection of things, enclosed in [ ] and separated by commas.

The list is a sequence data type which is used to store the collection of data. Tuples and String are other types of sequence data types.

Python Lists

mylist = ["apple", "banana", "cherry"]

List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

Lists are created using square brackets:

Example

Create a List:

thislist = [“apple”, “banana”, “cherry”]

print(thislist)

OUTPUT

[‘apple’, ‘banana’, ‘cherry’]

List Items

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

Note: There are some list methods that will change the order, but in general: the order of the items will not change.

Changeable

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Allow Duplicates

Since lists are indexed, lists can have items with the same value:

Example

Lists allow duplicate values:

thislist = [“apple”, “banana”, “cherry”, “apple”, “cherry”]

print(thislist)

OUTPUT

[‘apple’, ‘banana’, ‘cherry’, ‘apple’, ‘cherry’]

Python Tuples

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable.**

Tuples are written with round brackets.

Example: Get your own Python Server

Create a Tuple:

thistuple = (“apple”, “banana”, “cherry”)

print(thistuple)

Python Sets

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.

A set is a collection which is unordered, unchangeable*, and unindexed.

Note: Set items are unchangeable, but you can remove items and add new items.

Sets are written with curly brackets.

Example:

Create a Set:

thisset = {“apple”, “banana”, “cherry”}

print(thisset)

{‘banana’, ‘cherry’, ‘apple’}

Note: Sets are unordered, so you cannot be sure in which order the items will appear.

Set Items

Set items are unordered, unchangeable, and do not allow duplicate values.

Unordered

Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

Unchangeable

Set items are unchangeable, meaning that we cannot change the items after the set has been created.

Once a set is created, you cannot change its items, but you can remove items and add new items.

Duplicates Not Allowed

Sets cannot have two items with the same value.

Dictionaries

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered*, changeable and do not allow duplicates.

As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

Dictionaries are written with curly brackets, and have keys and values:

Example

Create and print a dictionary:

```
thisdict = {
  “brand”: “Ford”,
  “model”: “Mustang”,
  “year”: 1964
}
print(thisdict)
```

OUTPUT

{‘brand’: ‘Ford’, ‘model’: ‘Mustang’, ‘year’

Dictionary Items

Dictionary items are ordered, changeable, and do not allow duplicates.

Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

Example

Print the “brand” value of the dictionary:

```
thisdict = {
  “brand”: “Ford”,
  “model”: “Mustang”,
  “year”: 1964
}
print(thisdict[“brand”]
```

OUTPUT

Ford

Arrays

Note: This page shows you how to use LISTS as ARRAYS, however, to work with arrays in Python you will have to import a library, like the NumPy library.

Arrays are used to store multiple values in one single variable:

Example

Create an array containing car names:

```
cars = [“Ford”, “Volvo”, “BMW”]
```

OUTPUT

[‘Ford’, ‘Volvo’, ‘BMW’]

What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
car1 = “Ford”
car2 = “Volvo”
car3 = “BMW”
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Access the Elements of an Array

You refer to an array element by referring to the index number.

Example:

Get the value of the first array item:

x = cars[0]

Example:

Modify the value of the first array item:

cars[0] = “Toyota”

OUTPUT

VE

The Length of an Array

Use the len() method to return the length of an array (the number of elements in an array).

Example:

Return the number of elements in the cars array:

x = len(cars)

Looping Array Elements

You can use the for in loop to loop through all the elements of an array.

Example:

Print each item in the cars array:

```
for x in cars:
  print(x)
```

Adding Array Elements

You can use the append() method to add an element to an array.

Example:

Add one more element to the cars array:

cars.append(“Honda”)

Removing Array Elements

You can use the pop() method to remove an element from the array.

Example:

Delete the second element of the cars array:

cars.pop(1)

Iterators, Modules, Dates, Math

Python Iterators

- An iterator is an object that contains a countable number of values.
- An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.
- Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods __iter__() and __next__().

Iterator vs Iterable

Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable containers which you can get an iterator from.

All these objects have a iter() method which is used to get an iterator:

Example: Get your own Python Server

Return an iterator from a tuple, and print each value:

```
mytuple = (“apple”, “banana”, “cherry”)
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

OUTPUT

APPLE

BABANA

CHERRY

Even strings are iterable objects, and can return an iterator:

Example:

Strings are also iterable objects, containing a sequence of characters:

```
mystr = “banana”
myit = iter(mystr)

print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
```

OUTPUT

B

A

N

A

N

A

Looping Through an Iterator

We can also use a for loop to iterate through an iterable object:

Create an Iterator

To create an object/class as an iterator you have to implement the methods __iter__() and __next__() to your object.

As you have learned in the Python Classes/Objects chapter, all classes have a function called __init__(), which allows you to do some initializing when the object is being created.

The __iter__() method acts similar, you can do operations (initializing etc.), but must always return the iterator object itself.

The __next__() method also allows you to do operations, and must return the next item in the sequence.

Example

Create an iterator that returns numbers, starting with 1, and each sequence will increase by one (returning 1,2,3,4,5 etc.):

```
class MyNumbers:
  def __iter__(self):
    self.a = 1
    return self

  def __next__(self):
    x = self.a
    self.a += 1
    return x

myclass = MyNumbers()
myiter = iter(myclass)

print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
```


Python Modules

Create a Module

To create a module just save the code you want in a file with the file extension .py:

Python Modules

In this tutorial, we will explain how to construct and import custom Python modules. Additionally, we may import or integrate Python's built-in modules via various methods.

What is Modular Programming?

Modular programming is the practice of segmenting a single, complicated coding task into multiple, simpler, easier-to-manage sub-tasks. We call these subtasks modules. Therefore, we can build a bigger program by assembling different modules that act like building blocks.

Modularizing our code in a big application has a lot of benefits.

Simplification: A module often concentrates on one comparatively small area of the overall problem instead of the full task. We will have a more manageable design problem to think about if we are only concentrating on one module. Program development is now simpler and much less vulnerable to mistakes.

Flexibility: Modules are frequently used to establish conceptual separations between various problem areas. It is less likely that changes to one module would influence other portions of the program if modules are constructed in a fashion that reduces interconnectedness. (We might even be capable of editing a module despite being familiar with the program beyond it.) It increases the likelihood that a group of numerous developers will be able to collaborate on a big project.

Reusability: Functions created in a particular module may be readily accessed by different sections of the assignment (through a suitably established api). As a result, duplicate code is no longer necessary.

Scope: Modules often declare a distinct namespace to prevent identifier clashes in various parts of a program.

In Python, modularization of the code is encouraged through the use of functions, modules, and packages.

What are Modules in Python?

A document with definitions of functions and various statements written in Python is called a Python module.

In Python, we can define a module in one of 3 ways:

- Python itself allows for the creation of modules.
- Similar to the re (regular expression) module, a module can be primarily written in C programming language and then dynamically inserted at run-time.
- A built-in module, such as the itertools module, is inherently included in the interpreter.

A module is a file containing Python code, definitions of functions, statements, or classes. An example_module.py file is a module we will create and whose name is example_module.

We employ modules to divide complicated programs into smaller, more understandable pieces. Modules also allow for the reuse of code.

Rather than duplicating their definitions into several applications, we may define our most frequently used functions in a separate module and then import the complete module.

Let's construct a module. Save the file as example_module.py after entering the following.

Example:

```
1  # Here, we are creating a simple Python program to show how to create a module.
2  # defining a function in the module to reuse it
3  def square( number ):
4  # here, the above function will square the number passed as the input
5  result = number ** 2
6  return result     # here, we are returning the result of the function
```

Here, a module called example_module contains the definition of the function square(). The function returns the square of a given number.

How to Import Modules in Python?

In Python, we may import functions from one module into our program, or as we say into, another module.

For this, we make use of the import Python keyword. In the Python window, we add the next to import keyword, the name of the module we need to import. We will import the module we defined earlier example_module.

Syntax:

```
1  import example_module
```

The functions that we defined in the example_module are not immediately imported into the present program. Only the name of the module, i.e., example_ module, is imported here.

We may use the dot operator to use the functions using the module name. For instance:

Example:

```
1  # here, we are calling the module square method and passing the value 4
2  result = example_module.square(  4  )
3  print("By using the module square of number is: ", result )
```

Output:

By using the module square of number is: 16

There are several standard modules for Python. The complete list of Python standard modules is available. The list can be seen using the help command.

Similar to how we imported our module, a user-defined module, we can use an import statement to import other standard modules.

Importing a module can be done in a variety of ways. Below is a list of them.

Python import Statement

Using the import Python keyword and the dot operator, we may import a standard module and can access the defined functions within it. Here's an illustration.

Code

```
1  # Here, we are creating a simple Python program to show how to import a standard module
2  # Here, we are import the math module which is a standard module
3  import math
4  print( "The value of euler's number is", math.e )
5  # here, we are printing the euler's number from the math module
```

Output:

The value of euler's number is 2.718281828459045

Python Date

Python provides the **datetime** module work with real dates and times. In real-world applications, we need to work with the date and time. Python enables us to schedule our Python script to run at a particular timing.

In Python, the date is not a data type, but we can work with the date objects by importing the module named with **datetime, time, and calendar.**

In this section of the tutorial, we will discuss how to work with the date and time objects in Python.

The **datetime** classes are classified in the six main classes.

- **date** - It is a naive ideal date. It consists of the year, month, and day as attributes.
- **time** - It is a perfect time, assuming every day has precisely 24*60*60 seconds. It has hour, minute, second, microsecond, and tzinfo as attributes.
- **datetime** - It is a grouping of date and time, along with the attributes year, month, day, hour, minute, second, microsecond, and tzinfo.
- **timedelta** - It represents the difference between two dates, time or datetime instances to microsecond resolution.
- **tzinfo** - It provides time zone information objects.
- **timezone** - It is included in the new version of Python. It is the class that implements the tzinfo abstract base class.

❖ **Tick**

In Python, the time instants are counted since 12 AM, 1st January 1970. The function time() of the module time returns the total number of ticks spent since 12 AM, 1st January 1970. A tick can be seen as the smallest unit to measure the time.

Consider the following example

```
1  import time;
2  #prints the number of ticks spent since 12 AM, 1st January 1970
3  print(time.time())
```

Output:

1585928913.6519969

How to get the current time?

The localtime() functions of the time module are used to get the current time tuple. Consider the following exam.

Python Math

Python Math Module

Introduction

In this article, we are discussing Math Module in Python. We can easily calculate many mathematical calculations in Python using the Math module. Mathematical calculations may occasionally be required when dealing with specific fiscal or rigorous scientific tasks. Python has a math module that can handle these complex calculations. The functions in the math module can perform simple mathematical calculations like addition (+) and subtraction (-) and advanced mathematical calculations like trigonometric operations and logarithmic operations.

This tutorial teaches us about applying the math module from fundamentals to more advanced concepts with the support of easy examples to understand the concepts fully. We have included the list of all built-in functions defined in this module for better understanding.

What is Math Module in Python?

Python has a built-in math module. It is a standard module, so we do not need to install it separately. We only must import it into the program we want to use. We can import the module, like any other module of Python, using import math to implement the functions to perform mathematical operations.

Since the source code of this module is in the C language, it provides access to the functionalities of the underlying C library. Here we have given some basic examples of the Math module in Python. The examples are written below -

Program Code 1:

Here we give an example of a math module in Python for calculating the square root of a number. The code is given below -

```
1  # This program will show the calculation of square root using the math module
2  # importing the math module
3  import math
4  print(math.sqrt( 9 ))
```

Output:

Now we compile the above code in Python, and after successful compilation, we run it. Then the output is given below -

3.0

This Python module does not accept complex data types. The more complicated equivalent is the math module.

We can, for example, calculate all trigonometric ratios for any given angle using the built-in functions in the math module. We must provide angles in radians to these trigonometric functions (sin, cos, tan, etc.). However, we are accustomed to measuring angles in terms of degrees. The math module provides two methods to convert angles from radians to degrees and vice versa.

Program code 2:

Here we give an example of a math module in Python for calculating the factorial of a number. The code is given below -

```
1  import math
2  n=int(input())
3  print(math.factorial(n))
```

Output:

Now we compile the above code in Python, and after successful compilation, we run it. Then the output is given below -

```
5
120
```

Python User Input

User Input

Python allows for user input.

That means we are able to ask the user for input.

The method is a bit different in Python 3.6 than Python 2.7.

Python 3.6 uses the input() method.

Python 2.7 uses the raw_input() method.

The following example asks for the username, and when you entered the username, it gets printed on the screen:

Python 3.6

```
username = input("Enter username:")
print("Username is: " + username)
```

OUTPUT

ENTER USERNAME:

Modules, Input and Output

Python input

How to Take Input from User in Python

Sometimes a developer might want to take user input at some point in the program. To do this Python provides an input() function.

Syntax:

input('prompt')

where prompt is an optional string that is displayed on the string at the time of taking input.

Example 1: Python get **user input with a message**

- Python3

```
# Taking input from the user
name = input("Enter your name: ")

# Output
print("Hello, " + name)
print(type(name))
```

Output:

```
Enter your name: NSTI
Hello, NSTI
<class 'str'>
```

Note: Python takes all the input as a string input by default. To convert it to any other data type we have to convert the input explicitly. For example, to convert the input to int or float we have to use the int() and float() method respectively.

Example 2: **Integer input in Python**

- Python3

```
# Taking input from the user as integer
num = int(input("Enter a number: "))
add = num + 1
# Output
print(add)
```

Output:

```
Enter a number: 25
26
```

How to take Multiple Inputs in Python :

we can take multiple inputs of the same data type at a time in python, using map() method in python.

- Python3

```
a, b, c = map(int, input("Enter the Numbers : ").split())
print("The Numbers are : ",end = " ")
print(a, b, c)
```

Output :

```
Enter the Numbers : 2 3 4
The Numbers are :  2 3 4
```

Python output

How to Display Output in Python

Python provides the print() function to display output to the standard output devices.

Syntax: print(value(s), sep= ' ', end = '\n', file=file, flush=flush)

Parameters:

value(s) : Any value, and as many as you like. Will be converted to string before printed

sep='separator' : (Optional) Specify how to separate the objects, if there is more than one.Default :' '

end='end': (Optional) Specify what to print at the end.Default : '\n'

file : (Optional) An object with a write method. Default :sys.stdout

flush : (Optional) A Boolean, specifying if the output is flushed (True) or buffered (False). Default: False

Returns: It returns output to the screen.

Example: **Python Print Output**

- Python3

```
# Python program to demonstrate
# print() method
print("NSTI")

# code for disabling the softspace feature
print('N', 'S', 'T', 'I')')
```

Output

```
NSTI
N S T I
```

In the above example, we can see that in the case of the 2nd print statement there is a space between every letter and the print statement always add a new line character at the end of the string. This is because after every character the sep parameter is printed and at the end of the string the end parameter is printed. Let's try to change this sep and end parameter.

Example: **Python Print output with custom sep and end parameter**

- Python3

```
# Python program to demonstrate
# print() method
print("NSTI", end = "@")

# code for disabling the softspace feature
print('N', 'S', 'T', 'I', sep="#")
```

Output

```
NSTI@N#S#T#I
```

Formatting Output

Formatting output in Python can be done in many ways. Let's discuss them below

Using formatted string literals

We can use formatted string literals, by starting a string with f or F before opening quotation marks or triple quotation marks. In this string, we can write Python expressions between { and } that can refer to a variable or any literal value.

Example: **Python String formatting using F string**

- Python3

```
# Declaring a variable
name = "Nsti"

# Output
print(f'Hello {name}! How are you?')
```

Output:

```
Hello Nsti! How are you?
```

Functions & arguments

In Python, functions are blocks of organized, reusable code that perform a specific task.. They allow you to break down your program into smaller, modular pieces, making your code more organized, easier to read, and easier to maintain. Functions can take arguments, which are inputs that the function operates on, and they can return a value as a result of their operation.

Here's a breakdown of functions and arguments in Python:

1 Defining a Function:

- You define a function in Python using the def keyword followed by the function name and parentheses (). If the function takes arguments, you list them inside the parentheses. The function body is then indented below the definition. Here's a basic example:

```
def greet():
    print("Hello, world!")
```

2 Function Arguments:

- Functions can take arguments, which are values passed into the function when it is called. These arguments provide input data to the function for it to operate on. Arguments are specified within the parentheses following the function name. There are different types of arguments in Python:
  - Positional Arguments: These are arguments that are matched based on their position in the function call. The order of the arguments matters.
  - Keyword Arguments: These are arguments preceded by a keyword when calling a function. The order of keyword arguments does not matter.
  - Default Arguments: These are arguments that have a default value specified in the function definition. If the caller doesn't provide a value for these arguments, the default value is used.
  - Variable-Length Arguments: Functions can accept a variable number of arguments. This is achieved using *args for positional arguments and **kwargs for keyword arguments.
- Here's an example illustrating these different types of arguments:

```
def greet(name, greeting="Hello"):
    print(f"{greeting}, {name}!")

greet("Alice")  # Output: Hello, Alice!
greet("Bob", "Hi")  # Output: Hi, Bob!
greet(greeting="Bonjour", name="Charlie")  # Output: Bonjour, Charlie!
```

- (name, g) # Output: H)) Outp

3 Returning Values:

- Functions can return a value using the return keyword. This value can be of any data type, including numbers, strings, lists, dictionaries, etc. If a function doesn't explicitly return a value, it implicitly returns None.

```
def add(x, y):
    return x + y

result = add(3, 5)
print(result)  # Output: 8
```

Functions and arguments are fundamental concepts in Python programming, enabling code reuse, modularity, and abstraction. They allow you to write efficient and maintainable code by breaking it down into smaller, manageable parts.

Modules

In Python, a module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended. Within a module, you can define functions, classes, and variables, and you can also include executable code.

Modules are used to organize code into reusable units. They help in organizing Python code logically and efficiently. You can think of a module as a library or a toolbox that contains related functions and classes.

Here's a basic example of a Python module:

```
# module.py

def greet(name):
    print("Hello, {}!".format(name))

def add(a, b):
    return a + b

pi = 3.14159
```

In this example, 'module.py' is a module containing a 'greet' function, an 'add' function, and a variable 'pi.' To use these definitions in another Python script or module, you can import them using the 'import' statement:

```
# main.py

import module

module.greet("Alice")
result = module.add(2, 3)
print("Result:", result)
print("Value of pi:", module.pi)
```

When 'main.py' is executed, it imports the 'module' module and then uses its functions and variables as needed.

Python also allows importing specific functions or variables from a module, rather than importing the entire module:

```
# main.py

from module import greet, add, pi

greet("Bob")
result = add(4, 5)
print("Result:", result)
print("Value of pi:", pi)
```

This way, only the specified functions and variables are imported into the current namespace.

Additionally, Python provides a special module called '__init__.py' which can be used to initialize packages. This module can contain initialization code or be left empty. It is automatically executed when the package is imported.

```
# __init__.py

# Initialization code for the package
```

These are the basics of modules in Python. They play a crucial role in structuring and organizing Python code into reusable and manageable units.

Exception Handling

Exception handling in Python allows you to gracefully manage errors or exceptional situations that may occur during the execution of your code. Python provides a way to catch and handle these exceptions using 'try', 'except', 'else', and 'finally' blocks.

Here's an overview of each component:

- try: The 'try' block is used to enclose the code that might raise an exception.

except: The 'except' block is used to catch and handle specific exceptions that occur within the corresponding 'try' block. You can specify the type of exception you want to handle, or you can use a generic 'except' block to catch any **exception.**

else: The 'else' block is executed if no exceptions occur in the 'try' block. It is optional.

finally: The 'finally' block is used to execute cleanup code that should be run regardless of whether an exception occurred. It is executed whether an exception occurred or not. It is optional.

Here's a basic example demonstrating how to use exception handling in Python:

```
try:
    # Code that might raise an exception
    x = int(input("Enter a number: "))
    result = 10 / x
    print("Result:", result)

except ZeroDivisionError:
    # Handle the specific exception (division by zero)
    print("Error: Cannot divide by zero.")

except ValueError:
    # Handle the specific exception (invalid input for conversion to int)
    print("Error: Invalid input. Please enter a valid integer.")

else:
    # This block is executed if no exceptions occur
    print("No exceptions occurred.")

finally:
    # Cleanup code (optional)
    print("Execution complete.")
```


In this example:

The 'try' block contains code that prompts the user to enter a number, performs a division operation, and prints the result.

The 'except' blocks handle specific exceptions ('ZeroDivisionError' and 'ValueError') that may occur within the try block.

The 'else' block prints a message if no exceptions occur.

The 'finally' block prints a message indicating that the execution is complete, regardless of whether an exception occurred or not.

Exception handling allows your program to recover from errors gracefully, ensuring that it doesn't crash unexpectedly and providing feedback or alternative paths when errors occur.

Built in Functions in Python

Certainly! Python comes with a wide range of built-in functions that are readily available for use without needing to import any modules. These functions serve various purposes and are foundational to programming in Python. Here are some of the most commonly used built-in functions in Python:

1 print(): Used to display the output to the standard output device (usually the console).

2 input(): Reads input from the user through the standard input device (usually the keyboard).

3 len(): Returns the length (the number of items) of an object. Works with sequences such as strings, lists, tuples, etc.

4 type(): Returns the type of an object.

5 int(), float(), str(), list(), tuple(), dict(), set(): These functions are used for type conversion. They convert the given value to the specified data type.

6 range(): Generates a sequence of numbers. It is often used with loops to iterate a certain number of times.

7 sum(): Returns the sum of all elements in a iterable (such as a list).

8 max(), min(): Returns the maximum or minimum value from a sequence or set of values.

9 abs(): Returns the absolute value of a number.

10 round(): Rounds a floating-point number to a specified number of decimal places.

11 sorted(): Returns a new sorted list from the elements of any iterable.

12 enumerate(): Returns an enumerate object that yields pairs of indexes and elements, useful for obtaining an index along with each element from an iterable.

13 zip(): Combines elements from multiple iterables into tuples.

14 map(): Applies a function to all the items in an input list or any other iterable.

15 filter(): Constructs an iterator from elements of an iterable for which a function returns true.

16 all(): Returns True if all elements of an iterable are true (or if the iterable is empty).

17 any(): Returns True if any element of an iterable is true. If the iterable is empty, returns False.

18 eval(): Evaluates a Python expression passed as a string.

19 format(): Formats a specified value into a specified format.

20 getattr(), setattr(), delattr(): Used to get, set, or delete an attribute from an object by name.

These are just a few examples of the many built-in functions available in Python. They provide a solid foundation for performing various operations and manipulations in Python programming.

File handling in Python

File handling in Python refers to the process of working with files on a computer's filesystem. Python provides several built-in functions and modules to facilitate file handling operations. Here's an overview of the basic file handling operations in Python:

- **1. Opening a File:** To open a file in Python, you use the 'open()' function, which returns a file object. The 'open()' function requires at least one argument, which is the path to the file you want to open. Optionally, you can specify the mode in which you want to open the file (read, write, append, etc.).

```
file = open('example.txt', 'r')  # Opens the file 'example.txt' in read mode
```

- **1. Reading from a File:** Once you have opened a file, you can read its contents using various methods such as' read()', 'readline()', or 'readlines()'.

```
content = file.read()  # Reads the entire content of the file
```

```
line = file.readline()  # Reads a single line from the file
```

```
lines = file.readlines()  # Reads all lines from the file and returns a list
```

- **1. Writing to a File:** To write data to a file, you need to open the file in write mode ('w'). You can then use the 'write()' method to write data to the file.

```
file = open('example.txt', 'w')  # Opens the file 'example.txt' in write mode
file.write('Hello, World!')  # Writes the string 'Hello, World!' to the file
```

- **1. Closing a File:** After performing file operations, it's important to close the file using the 'close()' method. This releases system resources associated with the file.

```
file.close()  # Closes the file
```

- **1. Appending to a File:** To append data to an existing file, you can open the file in append mode ('a') and use the' write()' method.

```
file = open('example.txt', 'a')  # Opens the file 'example.txt' in append mode
file.write('Appending this line.')  # Appends the string to the file
file.close()  # Don't forget to close the file
```

- **1. Using with Statement:** Python's with statement is preferred for file handling as it automatically closes the file when the block is exited, ensuring proper cleanup.

```
with open('example.txt', 'r') as file:
    content = file.read()
    # File is automatically closed when exiting the block
```

- These are the fundamental operations for handling files in Python. Additionally, Python also provides modules like 'os, shutil', and 'os.path' for performing more advanced file-related operations like file manipulation, directory handling, and path manipulations.

Python Architecture

Certainly! Below is a high-level description of Python's architecture written in Python-style pseudocode

```
class Python:

    def __init__(self):
        self.interpreter = Interpreter()
        self.standard_library = StandardLibrary()
        self.bytecode_compiler = BytecodeCompiler()
        self.parser = Parser()
        self.execution_engine = ExecutionEngine()
        self.memory_manager = MemoryManager()
        self.object_model = ObjectModel()
        self.garbage_collector = GarbageCollector()

class Interpreter:

    def execute(self, source_code):
        # Executes the Python code by parsing, compiling, and running it
        parsed_code = Python.parser.parse(source_code)
        compiled_bytecode = Python.bytecode_compiler.compile(parsed_code)
        result = Python.execution_engine.execute(compiled_bytecode)
        return result

class StandardLibrary:
```

```
    def __init__(self):
        # Contains built-in modules and functions
        pass

class BytecodeCompiler:

    def compile(self, parsed_code):
        # Transforms parsed Python code into bytecode
        bytecode = bytecode_generation(parsed_code)
        return bytecode

class Parser:

    def parse(self, source_code):
        # Converts source code into abstract syntax tree (AST)
        ast_tree = generate_ast(source_code)
        return ast_tree

class ExecutionEngine:
```

```
    def execute(self, compiled_bytecode):
        # Executes bytecode in a virtual machine
        result = virtual_machine.execute_bytecode(compiled_bytecode)
        return result

class MemoryManager:

    def allocate(self, size):
        # Allocates memory for Python objects
        memory = allocate_memory(size)
        return memory

class ObjectModel:

    def __init__(self):
        # Defines the structure and behavior of Python objects
        pass

class GarbageCollector:

    def collect(self, memory):
        # Frees memory of unused objects
        garbage = identify_garbage(memory)
        deallocate(garbage)
```


This Python-style pseudocode describes the main components of Python's architecture:

- Interpreter: Responsible for executing Python code by coordinating different components like parsing, compiling, and executing.
- Standard Library: Contains built-in modules and functions that provide a wide range of functionalities.
- Bytecode Compiler: Translates parsed Python code into bytecode, which is then executed by the virtual machine.
- Parser: Converts source code into an abstract syntax tree (AST) for further processing.
- Execution Engine: Executes bytecode within a virtual machine environment.
- Memory Manager: Allocates and manages memory for Python objects.
- Object Model: Defines the structure and behavior of Python objects.
- Garbage Collector: Identifies and frees memory occupied by unused objects to prevent memory leaks.

This architecture demonstrates how Python processes code from source to execution, handling memory management and providing a rich set of functionalities through its standard library and object model.

Documentation in Python

Certainly! Below is a Python-style pseudocode description of how you might implement documentation in Python:

```
class Documentation:

    def __init__(self):
        self.documentation_entries = {}

    def add_entry(self, name, description):
        """
        Add a new documentation entry.

        Args:
            name (str): The name of the entry.
            description (str): The description of the entry.

        Returns:
            None
        """
        self.documentation_entries[name] = description

    def remove_entry(self, name):
        """
        Remove a documentation entry.
```

```
        Args:
            name (str): The name of the entry to be removed.

        Returns:
            None
        """
        if name in self.documentation_entries:
            del self.documentation_entries[name]

    def get_entry(self, name):
        """
        Retrieve the description of a documentation entry.

        Args:
            name (str): The name of the entry to retrieve.

        Returns:
            str: The description of the entry.
        """
        return self.documentation_entries.get(name, "Entry not found")
```

```
# Example Usage:
doc = Documentation()

# Add entries
doc.add_entry("function1", "This is the documentation for function1")
doc.add_entry("class1", "This is the documentation for class1")

# Get entry
print(doc.get_entry("function1"))  # Output: This is the documentation for function1

# Remove entry
doc.remove_entry("class1")

# Trying to get removed entry
print(doc.get_entry("class1"))  # Output: Entry not found
```

In this pseudocode:

- The 'Documentation' class represents a collection of documentation entries.
- Each entry consists of a name and a description.
- The 'add_entry' method allows adding new documentation entries.
- The 'remove_entry' method allows removing documentation entries.
- The 'get_entry' method allows retrieving the description of a documentation entry by its name.

This structure provides a basic framework for managing documentation within a Python program. Actual Python documentation frameworks like Sphinx or tools like docstrings provide more comprehensive features and formatting options for documentation.