# **STUDY MATERIAL for Bootstrap Program**

**Session 1 -11**

## CONTENTS

# Session 1

## History of Computers

Before computers were developed people used sticks, stones, and bones as counting tools. As technology advanced and the human mind improved with time more computing devices were developed like Abacus, Napier's Bones, etc. These devices were used as computers for performing mathematical computations but not very complex ones.

The concept of computation dates back to ancient civilizations. The abacus, developed around 2000 BCE, was one of the earliest tools used for arithmetic calculations. Other devices, such as the Antikythera mechanism from ancient Greece, demonstrated a remarkable understanding of astronomical calculations.

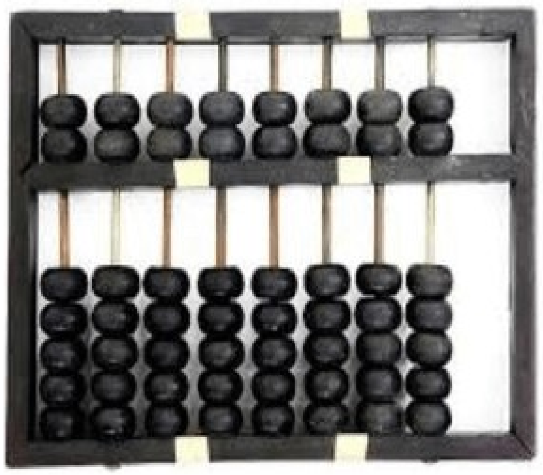
Some of the popular computing devices are described below, starting from the oldest to the latest or most advanced technology developed:

**Abacus**

Around 4000 years ago, the Chinese invented the Abacus, and it is believed to be the first computer. The history of computers begins with the birth of the abacus.

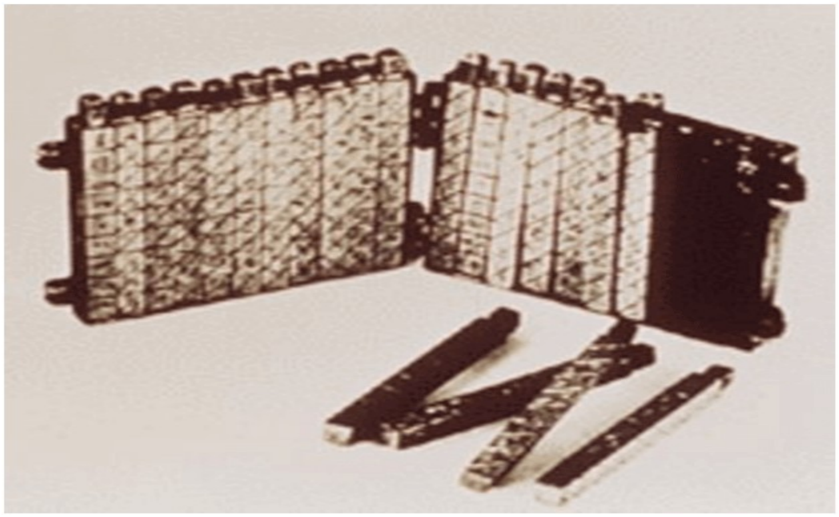
**Structure:** Abacus is basically a wooden rack that has metal rods with beads mounted on them.

**Working of abacus:** In the abacus, the beads were moved by the abacus operator according to some rules to perform arithmetic calculations. The beads represented the columns . The right-most column represented the unit , the next for ten and so on. Numbers were represented by moving beads at appropriate column. Abacus could perform simple addition and subtraction.



**Napier's Bones**

Napier's Bones was a manually operated calculating device and as the name indicates, it was invented by John Napier. In this device, he used 9 different ivory strips (bones) marked with numbers to multiply and divide for calculation. It was also the first machine to use the decimal point system for calculation.



**Pascaline**

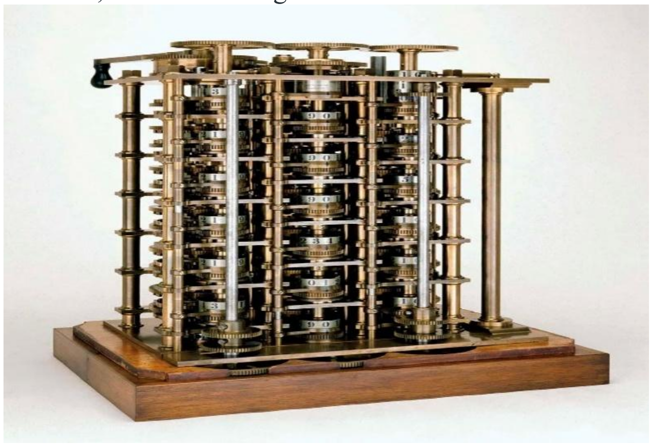
It is also called an Arithmetic Machine or Adding Machine. A French mathematician-philosopher Blaise Pascal invented this between 1642 and 1644. It was the first mechanical and automatic calculator. It was invented by Pascal to help his father, a tax accountant, in his work or calculation. It could perform addition and subtraction in quick time. It was basically a wooden box with a series of gears and wheels. It is worked by a rotating wheel like when a wheel is rotated one revolution, it rotates the neighbouring wheel and a series of windows is given on the top of the wheels to read the totals.

**Stepped Reckoner or Leibniz wheel**

A German mathematician-philosopher Gottfried Wilhelm Leibniz in 1673 developed this device by improving Pascal's invention to develop this machine. It was basically a digital mechanical calculator, and it was called the stepped reckoner as it was made of fluted drums instead of gears (used in the previous model of Pascaline).

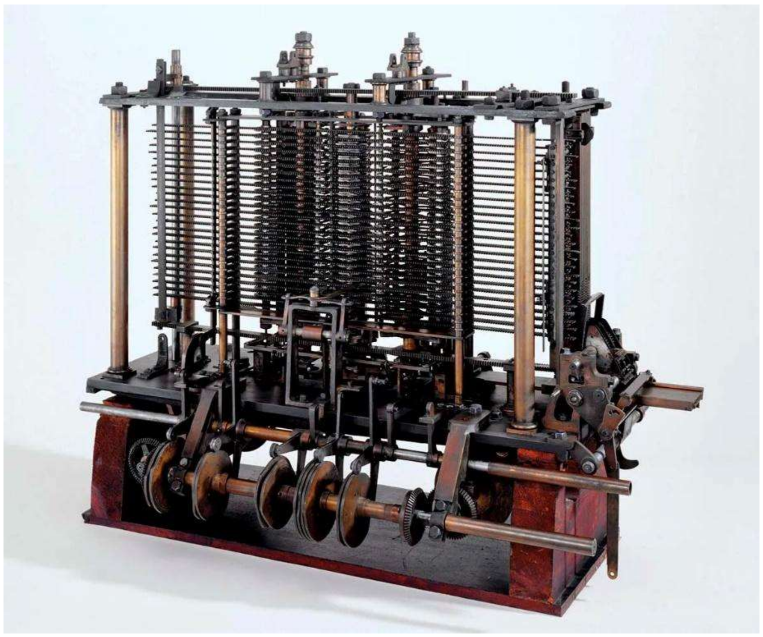
**Difference Engine**

Charles Babbage who is also known as the “Father of Modern Computer” designed the Difference Engine in the early 1820s. Difference Engine was a mechanical computer which is capable of performing simple calculations. It works with help of steam as it was a steam-driven calculating machine, and it was designed to solve tables of numbers like logarithm tables.



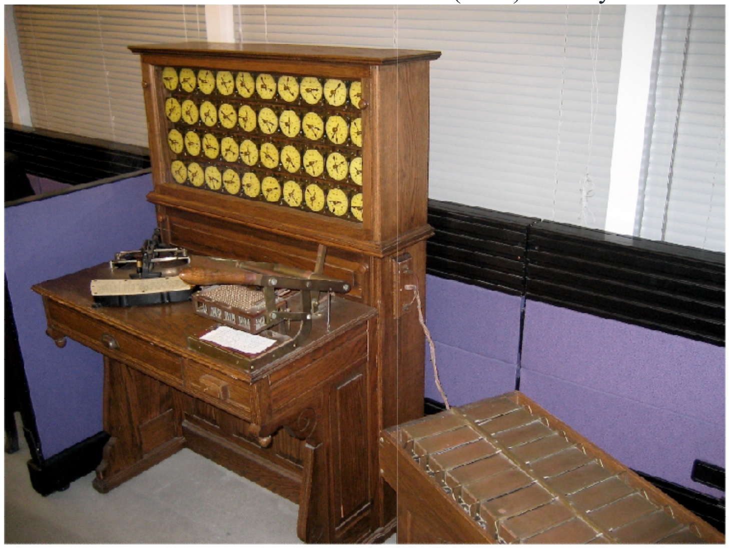
**Analytical Engine**

Again in 1830 Charles Babbage developed another calculating machine which was Analytical Engine. Analytical Engine was a mechanical computer that used punch cards as input. It was capable of performing or solving any mathematical problem and storing information as a permanent memory (storage).



**Tabulating Machine**

Herman Hollerith, an American statistician invented this machine in the year 1890. Tabulating Machine was a mechanical tabulator that was based on punch cards. It was capable of tabulating statistics and record or sort data or information. This machine was used by U.S. Census in the year 1890. Hollerith's Tabulating Machine Company was started by Hollerith and this company later became International Business Machine (IBM) in the year 1924.

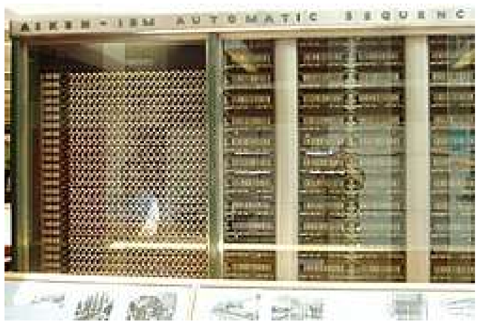


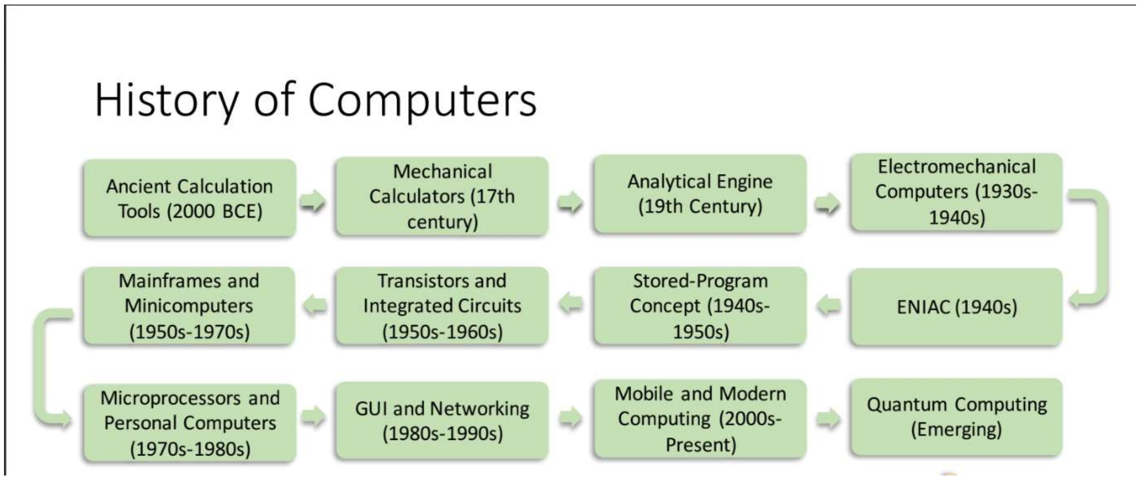
**Differential Analyzer**

Differential Analyzer was the first electronic computer introduced in the year 1930 in the United States. It was basically an analog device that was invented by Vannevar Bush. This machine consists of vacuum tubes to switch electrical signals to perform calculations. It was capable of doing 25 calculations in a few minutes.

**Mark I**

In the year 1937, major changes began in the history of computers when Howard Aiken planned to develop a machine that could perform large calculations or calculations involving large numbers. In the year 1944, Mark I computer was built as a partnership between IBM and Harvard. It was also the first programmable digital computer marking a new era in the computer world.





## History of Computers

$$\begin{tikzcd}
\text{Ancient Calculation Tools (2000 BCE)} \arrow[r] & \text{Mechanical Calculators (17th century)} \arrow[r] & \text{Analytical Engine (19th Century)} \arrow[r] & \text{Electromechanical Computers (1930s-1940s)} \arrow[d] \\
\text{Mainframes and Minicomputers (1950s-1970s)} \arrow[d] & \text{Transistors and Integrated Circuits (1950s-1960s)} \arrow[l] & \text{Stored-Program Concept (1940s-1950s)} \arrow[l] & \text{ENIAC (1940s)} \arrow[l] \\
\text{Microprocessors and Personal Computers (1970s-1980s)} \arrow[r] & \text{GUI and Networking (1980s-1990s)} \arrow[r] & \text{Mobile and Modern Computing (2000s-Present)} \arrow[r] & \text{Quantum Computing (Emerging)}
\end{tikzcd}$$

## Generations of Computers

**First Generation Computers [ 1940-1956] : Vacuum Tubes**

These machines were slow, huge, and expensive. In this generation of computers, vacuum tubes were used as the basic components of CPU and memory. Also, they were mainly dependent on the batch operating systems and punch cards. Magnetic tape and paper tape were used as output and input devices. For example ENIAC-Electronic Numerical Integrator and Calculator , UNIVAC-1- Universal Automatic Computer , EDVAC- Electronic Discrete Variable Automatic Computer, etc.

**Second Generation Computers [ 1957-1963] : Transistors**

It was the time of the transistor computers. In the second generation of computers, transistors (which were cheap in cost) were used. Transistors are also compact and consume less power. Transistor computers are faster than first-generation computers. For primary memory, magnetic cores were used, and for secondary memory magnetic disc and tapes for storage purposes. In second-generation computers, COBOL and FORTRAN were used as Assembly language and programming languages, and Batch processing and multiprogramming operating systems were used in these computers.

For example IBM 1620, IBM 7094, CDC 1604, CDC 3600, etc.

**Third Generation Computers [ 1964 – Early 1970s ] : Integrated Circuits**

In the third generation of computers, integrated circuits (ICs) were used instead of transistors(in the second generation). A single IC consists of many transistors which increased the power of a computer and also reduced the cost. The third generation computers were more reliable, efficient, and smaller in size. It used remote processing, time-sharing, and multiprogramming as operating systems. FORTRON-II TO IV, COBOL, and PASCAL PL/1 were used which are high-level programming languages.

For example IBM-360 series, Honeywell-6000 series, IBM-370/168, etc.

**Fourth Generation Computers [ Early 1970s – Till date ] : Microprocessors**

The period of 1971-1980 was mainly the time of fourth generation computers. It used VLSI(Very Large Scale Integrated) circuits. VLSI is a chip containing millions of transistors and other circuit elements and because of these chips, the computers of this generation are more compact, powerful, fast, and affordable(low in cost). Real-time, time-sharing and distributed operating system are used by these computers. C and C++ are used as the programming languages in this generation of computers.

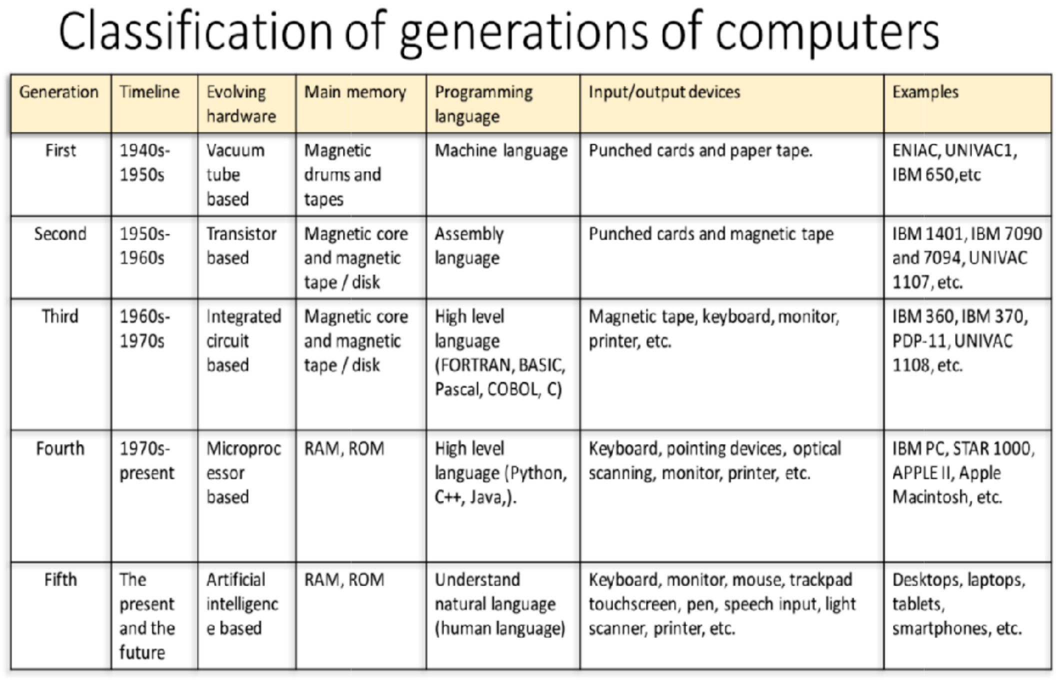
For example STAR 1000, PDP 11, CRAY-1, CRAY-X-MP, etc.

**Fifth Generation Computers [ Present and Beyond ] : Artificial Intelligence**

The ULSI (Ultra Large Scale Integration) technology is used in fifth-generation computers instead of the VLSI technology of fourth-generation computers. Use of fifth generation computers are currently in research level. Microprocessor chips with ten million electronic components are used in these computers. Parallel processing hardware and AI (Artificial Intelligence) software are also used in fifth-generation computers. The scientists are trying to use AI and KIPS to perform any complicated problem. The programming languages like C, C++, Java, .Net, etc. are used.

For example Desktop, Laptop, NoteBook, UltraBook, etc.

## Types of Computers



Classification of generations of computers

| Generation | Timeline | Evolving hardware | Main memory | Programming language | Input/output devices | Examples |
|---|---|---|---|---|---|---|
| First | 1940s-1950s | Vacuum tube based | Magnetic drums and tapes | Machine language | Punched cards and paper tape. | ENIAC, UNIVAC1, IBM 650,etc |
| Second | 1950s-1960s | Transistor based | Magnetic core and magnetic tape / disk | Assembly language | Punched cards and magnetic tape | IBM 1401, IBM 7090 and 7094, UNIVAC 1107, etc. |
| Third | 1960s-1970s | Integrated circuit based | Magnetic core and magnetic tape / disk | High level language (FORTRAN, BASIC, Pascal, COBOL, C) | Magnetic tape, keyboard, monitor, printer, etc. | IBM 360, IBM 370, PDP-11, UNIVAC 1108, etc. |
| Fourth | 1970s-present | Microprocessor based | RAM, ROM | High level language (Python, C++, Java,). | Keyboard, pointing devices, optical scanning, monitor, printer, etc. | IBM PC, STAR 1000, APPLE II, Apple Macintosh, etc. |
| Fifth | The present and the future | Artificial intelligence based | RAM, ROM | Understand natural language (human language) | Keyboard, monitor, mouse, trackpad touchscreen, pen, speech input, light scanner, printer, etc. | Desktops, laptops, tablets, smartphones, etc. |

**based on Data Type**

Based on the data type handling, computers can be categorized as Digital, Analog, and Hybrid.

**Digital**

Personal computers are an example of a digital computer. These computers accept input in the form of 0s and 1s. The computer processes binary input and provides the output. These computers perform all the logical & arithmetical operations. Any input given in any language is first converted into binary language and then the computer processes the information. Examples – laptops, PCs, mobile phones, desktops, etc.

**Analog**

These computers process analog data. Analog data keep varying. Hence, it does not have any discrete value. They read the continuous change in the input, process it, and then provide the output. Analog computers perform with equal diligence and accuracy. They are however slower than digital computers. They are also slightly less precise.
Analog computers are for a Speedometer, thermometer, frequency, and signal of voltage, measuring the resistance of a capacitor.

**Hybrid**

Hybrid computers are a mix of both analog and digital computers. These computers perform a high level of calculations. Hybrid computers are quick and efficient. They take input in analog form, convert it into digital form, and then process it to produce an output. scientists are also using hybrid computers for complex calculations. For example, in hospitals to measure the heartbeat of the patients, and at research institutes to measure earthquakes and other natural calamities.

**Based on the Purpose**

**Microcomputer**
Microcomputers are nothing but personal computers. These are single-chip systems. These are useful for personal use and can perform all the basic functions of the computer. Microcomputers require very little space and are comparatively inexpensive. Such computers have the most minimalistic requirement in terms of I/O devices. And have all the circuitry mounted on a single PCB. For example tablets, I pads, smartwatches, laptops, desktops
**Minicomputer**
Standing in between a microcomputer and a mainframe computer is the minicomputer. These computers are useful if people around 5 to 300. Those who want to operate the system at the same time. You can see such computers at the billing counters of malls or large institutions.
**Mainframe**
Mainframe computers are useful when a large number of people are involved. Like in the health care or retail sector who want to access data simultaneously. These computers process large amounts of data.

In addition, mainframe computers have evolved a lot over the years in terms of speed, size, and efficiency. These computers are just below the supercomputers. And sometimes are even more useful than a supercomputer. Examples – IBM z Series, System z9, etc.
**Supercomputers**
The biggest and fastest computers are supercomputers. Such computers can process trillions of functions within a few seconds. We generally use MPIS( Million Instructions Per Second) to measure their performance. These computers are specifically designed for scientific applications such as –

1. Encryption decryption of passwords
2. Weather forecasting
3. Testing of nuclear weapons
4. Scientific research of earth and other planetary systems, etc.

Examples – PARAM supercomputer series, Gravity Pipe for astrophysics, Deep Crack for deciphering codes, etc.
**Based on the Functionality**

**Workstations**
Workstation computers are for single usage and professional purposes. These are like our basic laptops and desktops but with added superior features. For example, double-processor motherboard, added graphic card, ECC RAM, etc. The workstations are more powerful as compared to generic PCs. These can handle heavy-duty functions. Like animation, CAD, audio & video editing, professional gaming, etc.
Examples: Apple PowerBook G4, SPARC CPU, MIPS CPU, etc.
**Servers**
These are hardware components or software programs that are built to assist other computers termed as clients. Together this architecture is called the client-server model. The client sends a request to the server and the server responds in return with a result or a solution. This proves that these server computers are more powerful than standard computers. The main purpose of these computers is to share data and resources with other computers.
Different types of servers are useful for different needs and applications. For example, cloud server, application server, database server, file server, etc. Each of these servers has a different purpose for different client needs.
**Embedded**
These computers are mainly microcontroller-based systems. Used for processing specific tasks. Embedded computers have a combination of software and hardware components. But, are usually a part of a larger system. Each of its components is designed from scratch to serve a specific purpose or complete a specific task. Another characteristic that distinguishes them from a standard PC is that all of its components are integrated into a single PCB or motherboard. They are most helpful for industrial use. this is because of their ruggedness.

Examples: GPS systems, centralized heating systems, fitness trackers, digital watches, electronic calculators, etc.

**Information appliances**

Mostly portable devices designed for specific functions come under this category. They can perform very restricted tasks for which they are built like text editors, music players, photography, videography, etc. The most common example is mobile phones. Many wearable devices are also available in the market.

This completes our article on the different types of computers. To sum up, it can be said that there is everything for everyone.

**Edge Devices**

An edge device is any piece of hardware that controls data flow at the boundary between two networks. Edge devices fulfill a variety of roles, depending on what type of device they are, but they essentially serve as network entry -- or exit -- points.

Traditional edge devices include edge routers, routing switches, firewalls, multiplexers, and other wide area network (WAN) devices. Intelligent edge devices have built-in processors with onboard analytics or artificial intelligence capabilities. Such devices might include sensors, actuators, and IoT gateways.

## Input and Output Devices of Computer with Examples

Nowadays computer directly or indirectly has become a part of our life from banking accounts to business deals. Do you know, How a computer works? Both input and output devices are important for a computer to work well and be easy to use. With the help of the input devices, you can send instructions to the system of the computer to execute an action based on your needs. With the help of the output devices, you can get the result or outcome of your inserting instructions on the screen or other output platforms. This article will provide you with information about the different input and output devices in a Computer system.

| Examples of Input Devices of Computer | |
|---|---|
| Keyboard | Mouse |
| Joysticks | Wii Remote |
| Light Pen | Game pad |
| Microphone | Webcam |
| Scanner | Digital camera |
| Barcode Reader | Portable Media Player |
| Trackball | Graphic Tablet |
| Magnetic Ink Card Reader (MICR) | Optical Character Reader (OCR) |
| Optical Character Reader (OCR) | Digitizer |

An input device is a computer device that allows computer users to enter data into a system and send instructions to the system to execute tasks accordingly. These devices are mainly hardware like keyboards, mouses, joysticks, etc. It is the first or primary step in the processing of computer data that is done at the Central Processing Unit (CPU). The delivered signals are received by the CPU which processes them. Input Devices of Computer can be classified further as per the modality like visual or audio, discrete or continuous, and direct or indirect. Some of the main input devices are explained below with their descriptions.Keyboard

The keyboard is the most fundamental input device of the computer. It is commonly used to insert data on the computer by using keys mounted on a keyboard. It is connected to the computer system through wifi or a USB cable. There are different varieties of keys for different purposes like numerals, letters, special characters, and functions. It is the main input device to command the computer system.

## Mouse

A Mouse is a hand-supported device that enables computer users to move the cursor point on the computer screen. A mouse consists of two buttons namely left and right on its top portion and one trackball at its bottom. It works on a flat surface to select and move the mouse around. The mouse as an input device was invented by Douglas C. Engelbart in 1063.

## Joy Stick

Joystick comprises a stick connected to the base at an angle so that it can be easily moved and controlled. It is mainly applied in controlling the movement of characters in video games. It is also employed in the cockpit of an airplane, wheelchairs, cranes, and trucks to regulate them properly. Its function is also to move the cursor on the screen but it is not like a mouse.

## Light Pen

A light pen is a pointing input device that is in a pen-like structure. It allows computer users to select options on the screen and even draw on the screen. It is light-sensitive equipment as photocells are inbuilt in this device that allows the flow of instructions to the CPU. It is generally used with a cathode ray tube (CRT) of the computer.

## Microphone

A microphone is an inbuilt voice input device in which different sounds can be collected in their digital form. This input device converts sound instructions into electrical signals. It has to be linked with an amplifier for recording or reproducing the sound.

## Scanner

A scanner is an input device that can scan different types of files of different formats like images or text and then convert them into digital signals. It can bring the document files to the computer screen by converting these files into a digital format. It operates like a photocopy machine to make the written information or data on a computer screen from paper.

## Barcode Reader

A barcode reader is a type of optical scanner device. It can read and understand the bar code data attached to goods, books, etc. It is like a reading device that throws a ray of light on a bar code and then the aspects and details of that particular item are displayed on the screen.

## Output Devices of Computer with Examples

An output device is a computer device that displays the result of the inserted input data after getting processed from the CPU. CPU of a computer converts received information from machine language to a human-friendly language and then sends these signals to output devices to execute the tasks as per entered data. These output devices are mainly hardware like printers, monitors, speakers, etc. When the data entered by the input devices is processed by the CPU of the computer then the output devices take care of the result. These results after processing of data may be in image, graphic, textual or audio form. The output devices display the visual contents on the monitor screen while sound contents are delivered to the speaker connected to a computer. Some of the main Output devices of computer are explained below with their descriptions.

| Examples of Output Devices of Computer | |
|---|---|
| Monitor | Speaker |
| Printer | Projector |
| Plotter | Braille Reader |
| Television | Global Positioning System |
| Headphones | Video Card |

## Monitor

The monitor is the main output device that displays all the data related to icons, text, images, etc. on its screen. When we enter the command to the computer to execute an action, then the outcome of that action is displayed on the screen of the monitor. Different types of monitors have been developed over time like CRT (Cathode Ray Tube) monitors, Flat-Panel display monitors, etc.

Printer

A Printer is an output device that produces a copy of the pictorial or textual files generally over a page. Its primary function is to print the information on paper. For example, an writer types a complete book on his computer system. He has to take a printout of it in the form of paper so that the book can be reviewed and later published. There are different types of printers in the market for different purposes like impact printers, character printers, line printers, laser printers, etc.

Speakers

A Speaker is an output device that converts electrical instructions into a sound signal. It helps you to listen to sound signals as an outcome of what you enter data into a computer. It is a hardware device that may or may not be attached to the computer system. Now, speakers are becoming wireless devices and can be linked with systems with the help of BlueTooth or else.

Projector

A Projector is the output and optical device that presents visual contents like moving or stationary images on the screen. These projecting devices are generally applied in auditoriums and cinema theatres for screening videos and lighting effects. Once a projector is linked to a computer system then the content displayed on the projection screen will be the same as displayed on the Monitor screen. The difference here is that projector displays contents on the bigger screens.

Headphones

The headphones operate on the same principles as a speaker operates. The only difference between headphones and speaker is the frequency of sound. With the help of speakers, the released sound can cover a larger area while with the help of headphones, the released sound can cover only a smaller area to make it audible only to a person who is wearing these headphones. These are also called earphones or headsets.

**Difference Between Input and Output Devices of Computer**

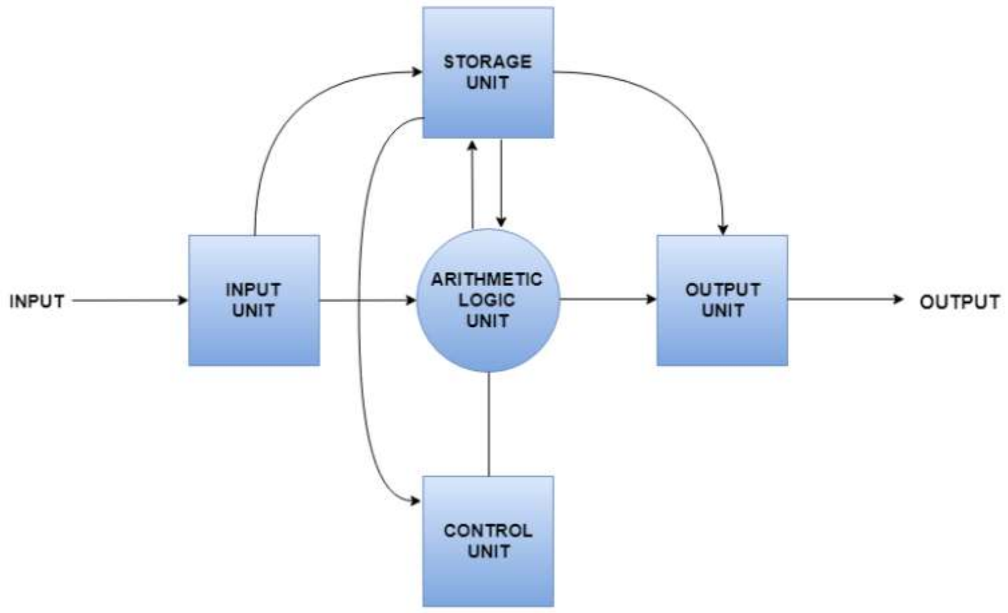
The computer operates smoothly with the help of both input and output devices. CPU plays a vital role in making a bridge between the input and output devices. Here the complete comparison of input and output devices of computers is explained below.

| Difference Between Input and Output Devices of Computer | |
|---|---|
| **Input Devices** | **Output Devices** |
| Accept computer user's data | Reflect computer user's data |
| The computer users command them | The computer processors command them |
| Conversion of Human-friendly signals into a machine-friendly language | Conversion of machine-friendly language to user-friendly<br>language |
| Send signals to the CPU for execution | Send processed signals to the computer user |
| Assist computer system in receiving data | Assist computer system in displaying data |
| It is complex in design comparatively | It is lesser complex in design |
| Example: Keyboard, Scanner, Mouse, etc. | Example: Monitor, Printers, Speaker, etc., |

## Computer system Architecture

A computer system is basically a machine that simplifies complicated tasks. It should maximize performance and reduce costs as well as power consumption.The different components in the Computer System Architecture are Input Unit, Output Unit, Storage Unit, Arithmetic Logic Unit, Control Unit etc.

A diagram that shows the flow of data between these units is as follows −



The input data travels from input unit to ALU. Similarly, the computed data travels from ALU to output unit. The data constantly moves from storage unit to ALU and back again. This is because stored data is computed on before being stored again. The control unit controls all the other units as well as their data.

Details about all the computer units are −

- **Input Unit**
  The input unit provides data to the computer system from the outside. So, basically it links the external environment with the computer. It takes data from the input devices, converts it into machine language and then loads it into the computer system. Keyboard, mouse etc. are the most commonly used input devices.
- **Output Unit**
  The output unit provides the results of computer process to the users i.e it links the computer with the external environment. Most of the output data is the form of audio or video. The different output devices are monitors, printers, speakers, headphones etc.
- **Storage Unit**
  Storage unit contains many computer components that are used to store data. It is traditionally divided into primary storage and secondary storage.Primary storage is also known as the main memory and is the memory directly accessible by the CPU. Secondary or external storage is not directly accessible by the CPU. The data from secondary storage needs to be brought into the primary storage before the CPU can use it. Secondary storage contains a large amount of data permanently.
- **Arithmetic Logic Unit**
  All the calculations related to the computer system are performed by the arithmetic logic unit. It can perform operations like addition, subtraction, multiplication, division

etc. The control unit transfers data from storage unit to arithmetic logic unit when calculations need to be performed. The arithmetic logic unit and the control unit together form the central processing unit.

- **Control Unit**
  This unit controls all the other units of the computer system and so is known as its central nervous system. It transfers data throughout the computer as required including from storage unit to central processing unit and vice versa. The control unit also dictates how the memory, input output devices, arithmetic logic unit etc. should behave.

**Computer architecture** can be defined as a set of rules and methods that describe the functionality, management and implementation of computers. To be precise, it is nothing but rules by which a system performs and operates.

**Sub-divisions**

Computer Architecture can be divided into mainly three categories, which are as follows −

- **Instruction set Architecture or ISA** − Whenever an instruction is given to processor, its role is to read and act accordingly. It allocates memory to instructions and also acts upon memory address mode (Direct Addressing mode or Indirect Addressing mode).
- **Micro Architecture** − It describes how a particular processor will handle and implement instructions from ISA.
- **System design** − It includes the other entire hardware component within the system such as virtualization, multiprocessing.

**Role of computer Architecture**

The main role of Computer Architecture is to balance the performance, efficiency, cost and reliability of a computer system.

**For Example** − Instruction set architecture (ISA) acts as a bridge between computer's software and hardware. It works as a programmer's view of a machine.

Computers can only understand binary language (i.e., 0, 1) and users understand high level language (i.e., if else, while, conditions, etc). So to communicate between user and computer, Instruction set Architecture plays a major role here, translating high level language to binary language.

## Computer Storage

The storage unit is a part of the computer system which is employed to store the information and instructions to be processed. A storage device is an integral part of the computer hardware which stores information/data to process the result of any computational work. Without a storage device, a computer would not be able to run or even boot up. Or in other words, we can say that a storage device is hardware that is used for storing, porting, or extracting data files. It can also store information/data both temporarily and permanently.

Types of Computer Memory

1. Primary Memory
2. Secondary Memory
3. Tertiary Memory

**1. Primary Memory:** It is also known as internal memory and main memory. This is a section of the CPU that holds program instructions, input data, and intermediate results. It is generally smaller in size. RAM (Random Access Memory) and ROM (Read Only Memory) are examples of primary storage.

**2. Secondary Memory:** Secondary storage is a memory that is stored external to the computer. It is mainly used for the permanent and long-term storage of programs and data. Hard Disks, CDs, DVDs, Pen/Flash drives, SSD, etc, are examples of secondary storage.
**3. Tertiary Memory:** Tertiary Memory is a type of Memory that is rarely used in personal computers and due to this, tertiary memory is not considered to be an important one. Tertiary memory works automatically without human intervention.
Types of Computer Storage Devices
Now we will discuss different types of storage devices available in the market. These storage devices have their own specification and use. Some of the commonly used storage devices are:

1. Primary Storage Devices
2. Magnetic Storage Devices
3. Flash memory Devices
4. Optical Storage Devices
5. Cloud and Virtual Storage

**1. Primary Storage Devices**

- **RAM:** It stands for Random Access Memory. It is used to store information that is used immediately or we can say that it is a temporary memory. Computers bring the software installed on a hard disk to RAM to process it and to be used by the user. Once, the computer is turned off, the data is deleted. With the help of RAM, computers can perform multiple tasks like loading applications, browsing the web, editing a spreadsheet, experiencing the newest game, etc. It allows you to modify quickly among these tasks, remembering where you're in one task once you switch to a different task. It is also used to load and run applications, like your spreadsheet program, answers commands, like all edits you made within the spreadsheet, or toggle between multiple programs, like once you left the spreadsheet to see the email. Memory is nearly always actively employed by your computer. It ranges from 1GB – 32GB/64GB depending upon the specifications. There are different types of RAM, and although they all serve the same purpose, the most common ones are :
    - **SRAM:** It stands for Static Random Access Memory. It consists of circuits that retain stored information as long as the power supply is on. It is also known as volatile memory. It is used to build Cache memory. The access time of SRAM is lower and it is much faster as compared to DRAM but in terms of cost, it is costly as compared to DRAM.
    - **DRAM:** It stands for Dynamic Random Access Memory. It is used to store binary bits in the form of electrical charges that are applied to capacitors. The access time of DRAM is slower as compared to SRAM but it is cheaper than SRAM and has a high packaging density.
    - **SDRAM:** It stands for Synchronous Random Access Memory. It is faster than DRAM. It is widely used in computers and others. After SDRAM was introduced, the upgraded version of double data rate RAM, i.e., DDR1, DDR2, DDR3, and DDR4 was entered into the market and widely used in home/office desktops and laptops.

  **ROM:** It stands for Read Only Memory. The data written or stored in these devices are non-volatile, i.e, once the data is stored in the memory cannot be modified or deleted. The memory from which will only read but cannot write it. This type of memory is non-volatile. The information is stored permanently during manufacture only once. ROM stores instructions that are used to start a computer. This operation is referred to as bootstrap. It is also used in other electronic items like washers and microwaves. ROM chips can only store a few megabytes (MB) of data, which ranges between 4 and 8 MB per ROM chip. There are two types of ROM:
    - **PROM:** PROM is Programmable Read Only Memory. These are ROMs that can be programmed. A special PROM programmer is employed to enter the program on the PROM. Once the chip has been programmed, information on

the PROM can't be altered. PROM is non-volatile, that is data is not lost when power is switched off.

- **EPROM:** Another sort of memory is the Erasable Programmable Read Only Memory. It is possible to erase the info which has been previously stored on an EPROM and write new data onto the chip.
- **EEPROM:** EEPROM is Electrically Erasable Read Only Memory. Here, data can be erased without using ultraviolet light, with the use of just applying the electric field.

**2. Magnetic Storage Devices**

- **Floppy Disk:** Floppy Disk is also known as a floppy diskette. It is generally used on a personal computer to store data externally. A Floppy disk is made up of a plastic cartridge and secured with a protective case. Nowadays floppy disk is replaced by new and effective storage devices like USB, etc.
- **Hard Disk:** Hard Disk is a storage device (HDD) that stores and retrieves data using magnetic storage. It is a non-volatile storage device that can be modified or deleted n number of times without any problem. Most computers and laptops have HDDs as their secondary storage device. It is actually a set of stacked disks, just like phonograph records. In every hard disk, the data is recorded electromagnetically in concentric circles or we can say track present on the hard disk, and with the help of a head just like a phonograph arm(but fixed in a position) to read the information present on the track. The read-write speed of HDDs is not so fast but decent. It ranges from a few GBs to a few and more TB.
- **Magnetic Card:** It is a card in which data is stored by modifying or rearranging the magnetism of tiny iron-based magnetic particles present on the band of the card. It is also known as a swipe card. It is used like a passcode(to enter the house or hotel room), credit card, identity card, etc.
- **Tape Cassette:** It is also known as a music cassette. It is a rectangular flat container in which the data is stored in an analog magnetic tape. It is generally used to store audio recordings.
- **SuperDisk:** It is also called LS-240 and LS-120. It is introduced by Imation Corporation and it is popular with OEM computers. It can store data up to 240 MB.

**3. Flash Memory Devices**

It is a cheaper and more portable storage device. It is the most commonly used device to store data because is more reliable and efficient as compared to other storage devices. Some of the commonly used flash memory devices are:

- **Pen Drive:** It is also known as a USB flash drive that includes flash memory with an integrated USB interface. We can directly connect these devices to our computers and laptops and read/write data into them in a much faster and more efficient way. These devices are very portable. It ranges from 1GB to 256GB generally.
- **SSD:** It stands for Solid State drive, a mass storage device like HDD. It is more durable because it does not contain optical disks inside like hard disks. It needs less power as compared to hard disks, is lightweight, and has 10x faster read and writes speed as compared to hard disks. But, these are costly as well. While SSDs serve an equivalent function as hard drives, their internal components are much different. Unlike hard drives, SSDs don't have any moving parts and thus they're called solid-state drives. Instead of storing data on magnetic platters, SSDs store data using non-volatile storage. Since SSDs haven't any moving parts, they do not need to "spin up". It ranges from 150GB to a few more TB.
- **SD Card:** It is known as a Secure Digital Card. It is generally used with electronic devices like phones, digital cameras, etc. to store larger data. It is portable and the size of the SD card is also small so that it can easily fit into electronic devices. It is available in different sizes like 2GB, 4GB, 8GB, etc.

- **Memory Card:** It is generally used in digital cameras. printers, game consoles, etc. It is also used to store large amounts of data and is available in different sizes. To run a memory card on a computer you require a separate memory card reader.
- **Multimedia Card:** It is also known as MMC. It is an integrated circuit that is generally used in-car radios, digital cameras, etc. It is an external device to store data/information.

**4. Optical Storage Devices**

Optical Storage Devices is also secondary storage device. It is a removable storage device. Following are some optical storage devices:

- **CD:** It is known as Compact Disc. It contains tracks and sectors on its surface to store data. It is made up of polycarbonate plastic and is circular in shape. CD can store data up to 700MB. It is of two types:
    - **CD-R:** It stands for Compact Disc read-only. In this type of CD, once the data is written can not be erased. It is read-only.
    - **CD-RW:** It stands for Compact Disc Read Write. In this type of CD, you can easily write or erase data multiple times.

  **DVD:** It is known as Digital Versatile Disc. DVDs are circular flat optical discs used to store data. It comes in two different sizes one is 4.7GB single-layer discs and another one is 8.5GB double-layer discs. DVDs look like CDs but the storage capacity of DVDs is more than as compared to CDs. It is of two types:
    - **DVD-R:** It stands for Digital Versatile Disc read-only. In this type of DVD, once the data is written can not be erased. It is read-only. It is generally used to write movies, etc.
    - **DVD-RW:** It stands for Digital Versatile Disc Read Write. In this type of DVD, you can easily write or erase data multiple times.

  **Blu-ray Disc:** It is just like CD and DVD but the storage capacity of blu ray is up to 25GB. To run a Blu-ray disc you need a separate Blu-ray reader. This Blu-ray technology is used to read a disc from a blue-violet laser due to which the information is stored in greater density with a longer wavelength.

**5. Cloud and Virtual Storage**

Nowadays, secondary memory has been upgraded to virtual or cloud storage devices. We can store our files and other stuff in the cloud and the data is stored for as long as we pay for the cloud storage. There are many companies that provide cloud services largely Google, Amazon, Microsoft, etc. We can pay the rent for the amount of space we need and we get multiple benefits out of it. Though it is actually being stored in a physical device located in the data centers of the service provider, the user doesn't interact with the physical device and its maintenance. For example, Amazon Web Services offers AWS S3 as a type of storage where users can store data virtually instead of being stored in physical hard drive devices. These sorts of innovations represent the frontier of where storage media goes.

## Characteristics of Computer Storage Devices

- Data stored in the Memory can be changed or replaced in case of a requirement, because of the mobility of the storage devices.
- Storage Devices validate that saved data can be replaced or deleted as per the requirements because the storage devices are easily readable, writeable, and rewritable.
- Storage Devices are easy and convenient to access because they do not require much skill set to handle these resources.
- The storage capacity of these devices is an extra advantage to the system.
- Storage Devices have better performance and data can be easily transferred from one device to another.

## Overview of Software

## Software and its Types

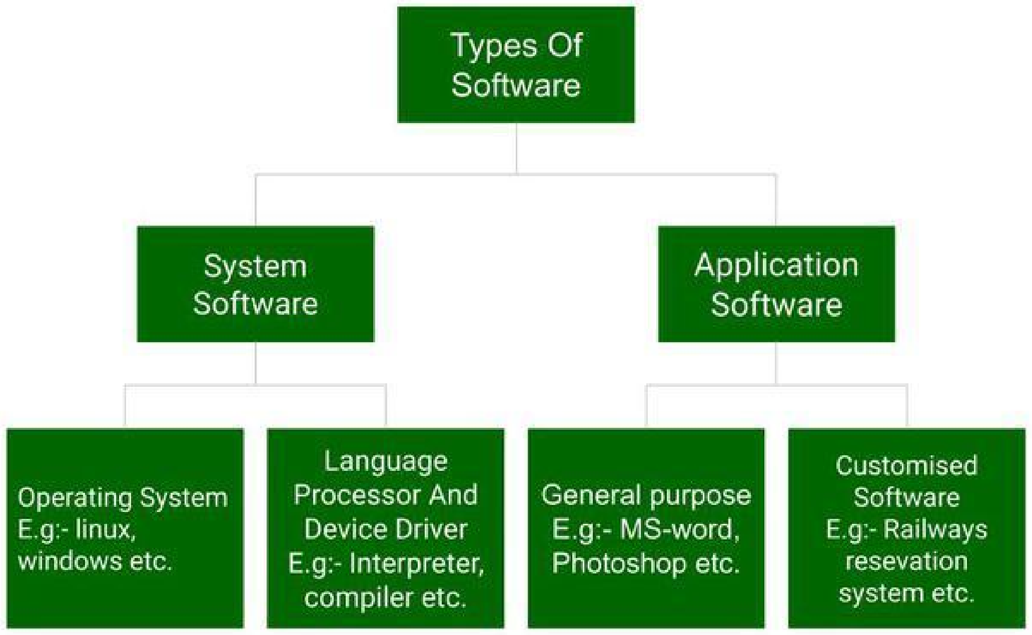
**Software** is a collection of instructions, data, or computer programs that are used to run machines and carry out particular activities. It is the antithesis of hardware, which refers to a computer's external components. A device's running programs, scripts, and applications are collectively referred to as "software" in this context.

**What is Software?**

In a Computer system, the software is basically a set of instructions or commands that tell a computer what to do. In other words, the software is a computer program that provides a set of instructions to execute a user's commands and tell the computer what to do. For example like MS-Word, MS-Excel,MS-Powerpoint, etc.

Types of Software

It is a collection of data that is given to the computer to complete a particular task. The chart below describes the types of software:


Types Of Software
System Software
Application Software
Operating System E.g:- linux, windows etc.
Language Processor And Device Driver E.g:- Interpreter, compiler etc.
General purpose E.g:- MS-word, Photoshop etc.
Customised Software E.g:- Railways resevation system etc.

Above is the diagram of types of software. Now we will briefly describe each type and its subtypes:

1. **System Software**
    - Operating System
    - Language Processor
    - Device Driver

0. **Application Software**
    - General Purpose Software
    - Customize Software
    - Utility Software

System Software

System Software is software that directly operates the computer hardware and provides the basic functionality to the users as well as to the other software to operate smoothly. Or in other words, system software basically controls a computer's internal functioning and also controls hardware devices such as monitors, printers, and storage devices, etc. It is like an interface between hardware and user applications, it helps them to communicate with each other because hardware understands machine language(i.e. 1 or 0) whereas user applications are work in human-readable languages like English, Hindi, German, etc. so system software converts the human-readable language into machine language and vice versa.

**Types of System Software**

It has two subtypes which are:

1. **Operating System:** It is the main program of a computer system. When the computer system is ON it is the first software that loads into the computer's memory. Basically, it manages all the resources such as computer memory ,CPU ,Printer , hard disk, etc., and provides an interface to the user, which helps the user to interact with the computer system. It also provides various services to other computer software. Examples of operating systems are Linux, Apple macOS, Microsoft Windows, etc.

   0. **Language Processor:** As we know that system software converts the human-readable language into a machine language and vice versa. So, the conversion is done by the language processor. It converts programs written in high-level programming languages like Java , C , C++, Python etc(known as source code), into sets of instructions that are easily readable by machines(known as object code or machine code).

   0. **Device Driver:** A device driver is a program or software that controls a device and helps that device to perform its functions. Every device like a printer, mouse, modem, etc. needs a driver to connect with the computer system eternally. So, when you connect a new device with your computer system, first you need to install the driver of that device so that your operating system knows how to control or manage that device.

**Features of System Software**

Let us discuss some of the features of System Software:

- System Software is closer to the computer system.
- System Software is written in a low-level language in general.
- System software is difficult to design and understand.
- System software is fast in speed(working speed).
- System software is less interactive for the users in comparison to application software.

Application Software

Software that performs special functions or provides functions that are much more than the basic operation of the computer is known as application software. Or in other words, application software is designed to perform a specific task for end-users. It is a product or a program that is designed only to fulfill end-users' requirements. It includes word processors, spreadsheets, database management, inventory, payroll programs, etc.

**Types of Application Software**

There are different types of application software and those are:

1. **General Purpose Software:** This type of application software is used for a variety of tasks and it is not limited to performing a specific task only. For example, MS-Word, MS-Excel, PowerPoint, etc.

   0. **Customized Software:** This type of application software is used or designed to perform specific tasks or functions or designed for specific organizations. For example, railway reservation system , airline reservation system, invoice management system, etc.

   0. **Utility Software:** This type of application software is used to support the computer infrastructure. It is designed to analyze, configure, optimize and maintains the system,

and take care of its requirements as well. For example, antivirus , disk fragmenter, memory tester, disk repair, disk cleaners, registry cleaners, disk space analyzer, etc.

**Features of Application Software**

Let us discuss some of the features of Application Software:

- An important feature of application software is it performs more specialized tasks like word processing, spreadsheets, email, etc.
- Mostly, the size of the software is big, so it requires more storage space.
- Application software is more interactive for the users, so it is easy to use and design.
- The application software is easy to design and understand.
- Application software is written in a high-level language in general.

Difference Between System Software and Application Software

Now, let us discuss some difference between system software and application software:

| **System Software** | **Application Software** |
|---|---|
| It is designed to manage the resources of the computer system, like memory and process management, etc. | It is designed to fulfil the requirements of the user for performing specific tasks. |
| Written in a low-level language. | Written in a high-level language. |
| Less interactive for the users. | More interactive for the users. |
| System software plays vital role for the effective functioning of a system. | Application software is not so important for the functioning of the system, as it is task specific. |
| It is independent of the application software to run. | It needs system software to run. |

.

Software and Its Type – FAQs

**1. What is the difference between a software and a program?**

*Both are for enabling the computer to perform specific task. The software is the collection of programs. we have created a program to perform the task and they compiled if there is no error in the program then program sent to create a software. Program is the set of instructions that are written by a programmer in a language while the software is a collection of programs that will enable the system to perform the specific task.*

**2. What is an example of Software?**

*You need software in order for the computer to operate effectively. It is a collection of data that is given to the computer to complete a particular task. three main categories of software are application software, programming software, and system software. All these three are important for the particular task to be performed.*

*Below are some software examples.*

- *Management Tools*
- *Compression Tools*
- *Internet Explorer*

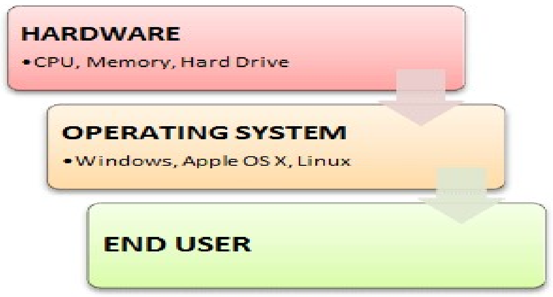
- *ERP( Enterprise Resource Planning )*
- *BI ( Business Intelligence)*
- *Customer Support Systems*

**3. Which software is used to control the operations of a computer?**
*There are two types of software system software and application software. System Software is used to control the operations and also controls a computer’s internal functioning and hardware devices.*
**4. Which software is designed to solve a specific problem or to do a specific task**
*Application Software is designed to solve a specific problem or to do a specific task Because a software that performs special functions or provides function which are much more than basic operation of the computer are application software.*

# Session 2

## Operating System

An **Operating System (OS)** is a software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs. Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks.
The OS helps you to communicate with the computer without knowing how to speak the computer's language. It is not possible for the user to use any computer or mobile device without having an operating system.



History Of OS

- Operating systems were first developed in the late 1950s to manage tape storage
- The General Motors Research Lab implemented the first OS in the early 1950s for their IBM 701
- In the mid-1960s, operating systems started to use disks
- In the late 1960s, the first version of the Unix OS was developed
- The first OS built by Microsoft was DOS. It was built in 1981 by purchasing the 86-DOS software from a Seattle company
- The present-day popular OS Windows first came to existence in 1985 when a GUI was created and paired with MS-DOS.

Types of Operating System (OS)
Following are the popular types of OS (Operating System):

- Batch Operating System
- Multitasking/Time Sharing OS
- Multiprocessing OS
- Real Time OS
- Distributed OS
- Network OS
- Mobile OS

**Batch Operating System**
Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs are batched together and run as a group.
The user of a batch operating system never directly interacts with the computer. In this type of OS, every user prepares his or her job on an offline device like a punch card and submit it to the computer operator.
**Multi-Tasking/Time-sharing Operating systems**

Time-sharing operating system enables people located at a different terminal(shell) to use a single computer system at the same time. The processor time (CPU) which is shared among multiple users is termed as time sharing.

**Real time OS**

A real time operating system allots time interval to process and respond to inputs which is very small. Examples: Military Software Systems, Space Software Systems are the Real time OS example.

**Distributed Operating System**

Distributed systems use many processors located in different machines to provide very fast computation to its users.

**Network Operating System**

Network Operating System runs on a server. It provides the capability to serve to manage data, user, groups, security, application, and other networking functions.

**Mobile OS**

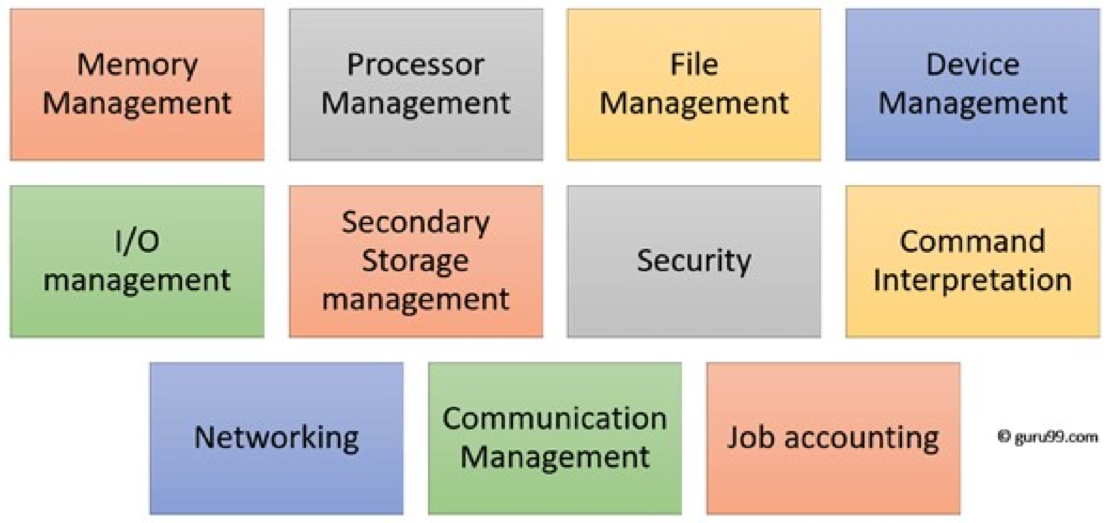
Mobile operating systems are those OS which is especially that are designed to power smartphones, tablets, and wearables devices.

Some most famous mobile operating systems are Android and iOS, but others include BlackBerry, Web, and watchOS.

**Functions of Operating System**

Some typical operating system functions may include managing memory, files, processes, I/O system & devices, security, etc.

Below are the main functions of Operating System:



Functions of Operating System

1. **Process management**: Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

0. **Memory management:** Memory management module performs the task of allocation and deallocation of memory space to programs in need of this resources.

0. **File management**: It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.

0. **Device Management**: Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and deallocation of the devices.

0. **I/O System Management:** One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.

0. **Secondary-Storage Management**: Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.

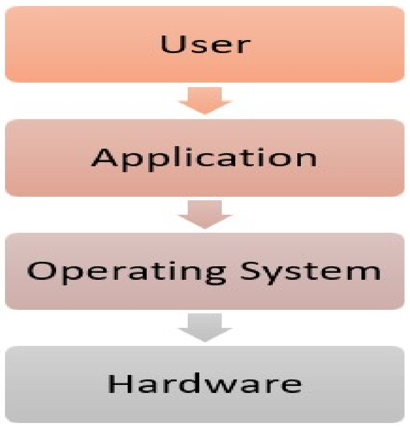
0. **Security**: Security module protects the data and information of a computer system against malware threat and authorized access.

0. **Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.

0. **Networking:** A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.

0. **Job accounting**: Keeping track of time & resource used by various job and users.

0. **Communication management**: Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

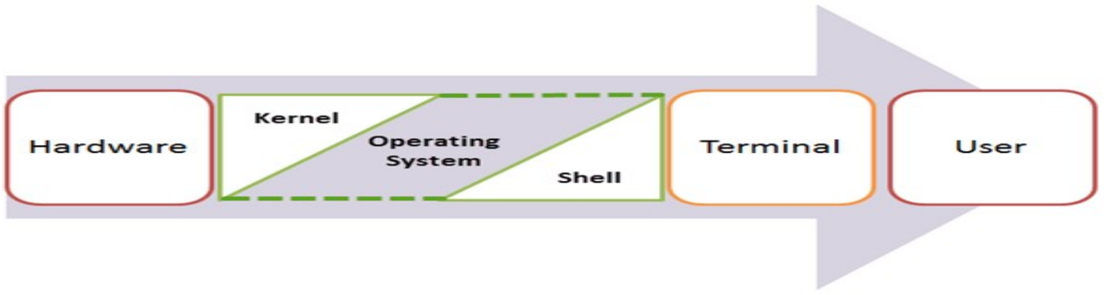


Advantage of Operating System

- Allows you to hide details of hardware by creating an abstraction
- Easy to use with a GUI
- Offers an environment in which a user may execute programs/applications
- The operating system must make sure that the computer system convenient to use
- Operating System acts as an intermediary among applications and the hardware components
- It provides the computer system resources with easy to use format
- Acts as an intermediary between all hardware and software of the system

What is Kernel in Operating Systems?

The kernel is the central component of a computer operating system. The only job performed by the kernel is to manage the communication between the software and the hardware. A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one.



**Major Operating systems :**

| Name | Features |
|---|---|
| MS- | stands for Microsoft Disk Operating system. |

| | |
|---|---|
| **DOS** | **does not take much space for installation ( about 8MB).<br>gives more control to the processes as it has a command-line user interface.<br>single-user, single-tasking operating system.<br>has limited graphics features.<br>not compatible with current browsers and the Internet.** |
| **Unix** | **was developed in 1969 by Ken Thompson, Dennis Ritchie and others at AT& T Bell Laboratories.<br>was earlier known as UNICS [ Uniplexed Information and Computing Service].<br>Was rewritten in C language. This made the OS portable.<br>one may use either the CUI that gives more control or flexibility or GUI which makes it simple.<br>very flexible and can be installed on micro , mini , mainframe and supercomputers.<br>very reliable, secured and robust.** |
| **Linux** | **developed by Linus Torvalds in 1992.<br>Unix-like OS and is available free.<br>like DOS, Linux is a command-line user interface.<br>It also gives GUI interfaces called desktop environments like GNOME and K Desktop Environment ( KDE ).<br>multi-user, multi-tasking OS.<br>reliable and secured OS.** |
| **Windows** | **first version of Windows was developed by Microsoft Corporation in 1985.<br>has a graphical user interface and is thus user-friendly.<br>Multi-tasking OS.<br>supports digital audio and video , CD and DVD drives .<br>contains built-in networking, which allows users to share files and applications with each other, if the computers are connected in a network.<br>can support a variety of application software.** |
| **MAC** | **It is a series of graphical user interface-based OS developed by Apple Inc. for their Macintosh line of computer systems.<br>extremely visually appealing.<br>multi-user, multi-tasking OS.<br>requires a low level of maintenance with fewer occurrence of worms, viruses and spyware.** |

**Mobile Operating Systems :**

| **Name** | **Features** |
|---|---|
| **Android** | **OS for mobile devices such as smart phones and tablet PCs.<br>developed by the Open Handset Alliance led by Google.<br>based on the Linux kernel and most applications are written in Java.<br>has an integrated browser based on the open source Webkit engine.<br>gives media support for common audio , video and image formats.<br>Was purchased by google in 2005.** |
| **Symbian** | **OS for mobile devices such as smartphones.<br>originally developed by Symbian Ltd but is currently manintained by Nokia.<br>has a graphics toolkit known as AVKON.<br>was the first mobile platform to make use of WebKit browser.<br>most applications are written in C++, though Symbian devices can be programmed using Python , Java ME , Flash Lite , .NET and so on.** |
| **BlackBerry** | **It is a proprietary mobile OS.<br>has been developed by Research In Motion for the BlackBerry line of smartphones.** |

| | |
|---|---|
| | **multitasking mobile OS.**<br>**supports specialised input devices like trackwheel , trackball , trackpad, and touchscreen.**<br>**BlackBerry platform is best known for its native support for corporate email.** |
| **Windows Mobile** | **has been developed by Microsoft.**<br>**Internet Explorer Mobile is the default browser.**<br>**contains Office Mobile- a suite of mobile versions of Microsoft Office.**<br>**based on Windows CE 5.2 kernel.**<br>**Internet Explorer Mobile 11 is the browser now used in Windows Mobile and new browser Microsoft Edge replaced IE in Windows 10 mobile.** |

## Binary Information and Representation: Bits, Bytes, Nibbles, Octets and Characters

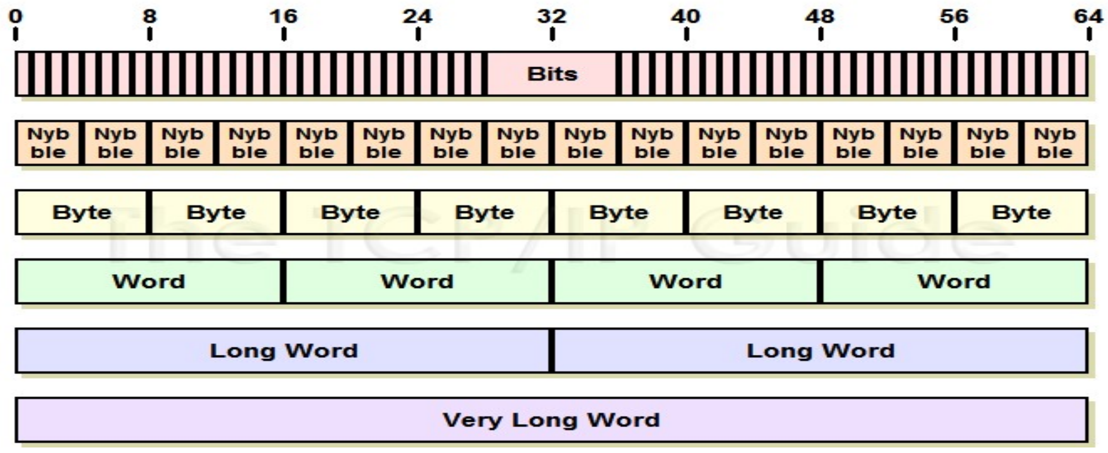
- The fundamental building block of computer information is the *bit* (a contraction of *binary digit*). Every bit can be either 0 or 1. Making the value of a bit 1 is commonly called *setting* the bit; changing it to 0 is *resetting* or *clearing* it.
- Of course, bits represent only a very small amount of information: a single fact or value. We must make collections of these bits so we can use them to store large amounts of information and more complex data types. The most common grouping is to take eight bits and reference them as a single unit. A collection of eight bits is technically called an *octet*, but is more commonly called a *byte*. (More on that in a moment.)
- "Byte" is a jocular play on the term "bit". Over time, other sizes of "bit collections" have also been defined. Some geek comedian decided that if eight bits made a "byte", then four bits must be a "nybble" (or "nibble").

**Table 1: Binary Information Group Representations and Terms**

| **Number of Bits** | **Common Representation Terms** |
|---|---|
| **1** | Bit / Digit / Flag |
| **4** | Nibble |
| **8** | Byte / Octet / Character |
| **16** | Double Byte / Word |
| **32** | Double Word / Long Word |
| **64** | Very Long Word |

- A few of these terms are worth special mention. Bit and byte we have already discussed, of course. A bit is also sometimes called a *flag*; this term is most often heard when a bit is used by itself to represent a particular information state. For example, a computer might use a "changed flag" to represent whether a particular file has been modified; this is an analogy to a flag either being raised or lowered to indicate a condition. These "flags" are often seen in networking message formats.

- The term *character* is also used to express a set of eight bits. This use comes from the fact that computers often store alphanumeric characters, such as letters and numbers, one to a byte. The 16-bit *word* is fairly often used, but not nearly as much as "byte". The larger collections of bits, such as double word and so on, are not often encountered in every-day parlance; they are used to represent chunks of data in technical fields such as hardware design or programming.



**Binary Information Representations and Terms**

<u>**Number System**</u>

The technique to represent and work with numbers is called Number system. Decimal number system is the most common number system. Other popular number systems include binary number system, octal number system, hexadecimal number system, etc.

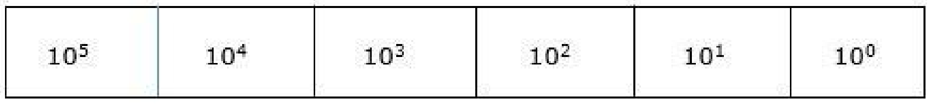
Decimal Number System

Decimal number system is a base 10 number system having 10 digits from 0 to 9. This means that any numerical quantity can be represented using these 10 digits. Decimal number system is also a positional value system. This means that the value of digits will depend on its position. Let us take an example to understand this.

Say we have three numbers – 734, 971 and 207. The value of 7 in all three numbers is different−

- In 734, value of 7 is 7 hundreds or 700 or 7 × 100 or 7 × $10^2$
- In 971, value of 7 is 7 tens or 70 or 7 × 10 or 7 × $10^1$
- In 207, value 0f 7 is 7 units or 7 or 7 × 1 or 7 × $10^0$

The weightage of each position can be represented as follows −



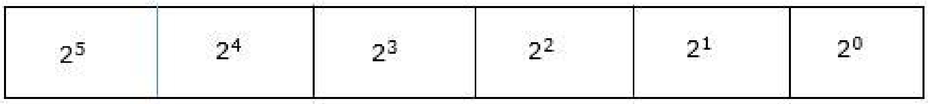
| | | | | | |
|---|---|---|---|---|---|
| $10^5$ | $10^4$ | $10^3$ | $10^2$ | $10^1$ | $10^0$ |

In digital systems, instructions are given through electric signals; variation is done by varying the voltage of the signal. Having 10 different voltages to implement decimal number system in digital equipment is difficult. So, many number systems that are easier to implement digitally have been developed. Let's look at them in detail.

Binary Number System

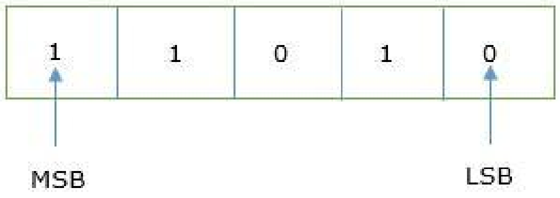
The easiest way to vary instructions through electric signals is two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. The number system having just these two digits – 0 and 1 – is called binary number system.

Each binary digit is also called a bit. Binary number system is also positional value system, where each digit has a value expressed in powers of 2, as displayed here.



| | | | | | |
|---|---|---|---|---|---|
| $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

In any binary number, the rightmost digit is called least significant bit (LSB) and leftmost digit is called most significant bit (MSB).



And decimal equivalent of this number is sum of product of each digit with its positional value.

$11010_2 = 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 16 + 8 + 0 + 2 + 0$

$= 26_{10}$

Computer memory is measured in terms of how many bits it can store. Here is a chart for memory capacity conversion.

- 1 byte (B) = 8 bits
- 1 Kilobytes (KB) = 1024 bytes
- 1 Megabyte (MB) = 1024 KB
- 1 Gigabyte (GB) = 1024 MB
- 1 Terabyte (TB) = 1024 GB
- 1 Exabyte (EB) = 1024 PB
- 1 Zettabyte = 1024 EB
- 1 Yottabyte (YB) = 1024 ZB

Octal Number System

Octal number system has eight digits – 0, 1, 2, 3, 4, 5, 6 and 7. Octal number system is also a positional value system with where each digit has its value expressed in powers of 8, as shown here −

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| $8^5$ | $8^4$ | $8^3$ | $8^2$ | $8^1$ | $8^0$ |

Decimal equivalent of any octal number is sum of product of each digit with its positional value.

$726_8 = 7\times8^2 + 2\times8^1 + 6\times8^0$

$= 448 + 16 + 6$

$= 470_{10}$

Hexadecimal Number System

Octal number system has 16 symbols – 0 to 9 and A to F where A is equal to 10, B is equal to 11 and so on till F. Hexadecimal number system is also a positional value system with where each digit has its value expressed in powers of 16, as shown here −

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| $16^5$ | $16^4$ | $16^3$ | $16^2$ | $16^1$ | $16^0$ |

Decimal equivalent of any hexadecimal number is sum of product of each digit with its positional value.

$27FB_{16} = 2\times16^3 + 7\times16^2 + 15\times16^1 + 10\times16^0$

$= 8192 + 1792 + 240 +10$

$= 10234_{10}$

ASCII

Besides numerical data, computer must be able to handle alphabets, punctuation marks, mathematical operators, special symbols, etc. that form the complete character set of English language. The complete set of characters or symbols are called alphanumeric codes. The complete alphanumeric code typically includes −

- 26 upper case letters
- 26 lower case letters
- 10 digits
- 7 punctuation marks
- 20 to 40 special characters

Now a computer understands only numeric values, whatever the number system used. So all characters must have a numeric equivalent called the alphanumeric code. The most widely used alphanumeric code is American Standard Code for Information Interchange (ASCII). ASCII is a 7-bit code that has 128 (27) possible codes.

ISCII

ISCII stands for Indian Script Code for Information Interchange. IISCII was developed to support Indian languages on computer. Language supported by IISCI include Devanagari, Tamil, Bangla, Gujarati, Gurmukhi, Tamil, Telugu, etc. IISCI is mostly used by government departments and before it could catch on, a new universal encoding standard called Unicode was introduced.

## Unicode

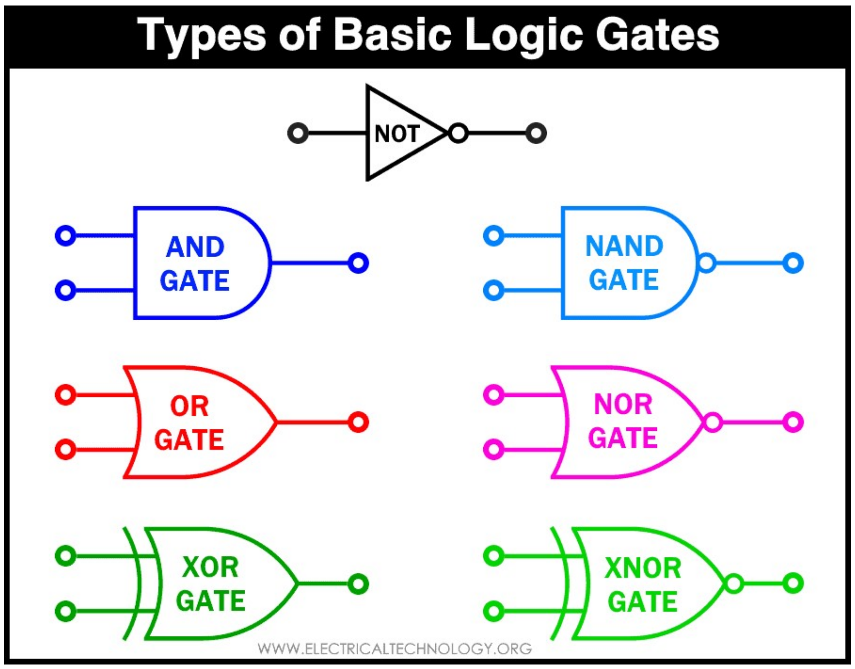
Unicode is an international coding system designed to be used with different language scripts. Each character or symbol is assigned a unique numeric value, largely within the framework of ASCII. Earlier, each script had its own encoding system, which could conflict with each other.

## Boolean Operators

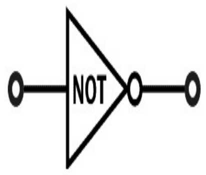
**What Is Boolean Logic?**
*Boolean logic* refers to the form of algebra where the variables have only 2 unique values i.e. TRUE or FALSE or ON and OFF. These values are often used as 1 or 0 in binary language or High and low logic respectively.

| | |
|---|---|
| 1 | 0 |
| True | False |
| High | Low |
| ON | OFF |

**Boolean Function**
A Boolean function is a logical operation of one or more than one variables whose resultant is a single binary bit. It can only be either TRUE or FALSE. Boolean functions are based on Boolean logic.
**What is a Digital Logic Gate?**
A digital logic gate is an electronic component which implements a Boolean function. these logic gates may have two or more than two binary inputs and provides a single binary output. Some of these basic logic gates are given below:



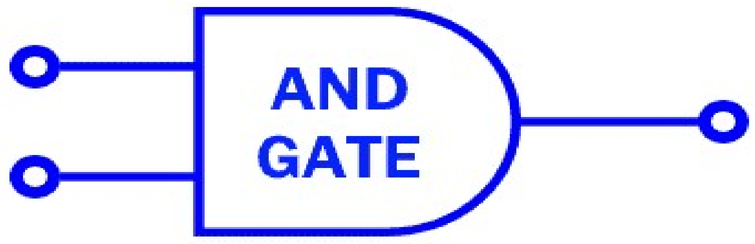
**Types of Basic Logic Gates**
*NOT Gate*



NOT gate is a basic digital logic gate. This logic gate inverts its input logic i.e LOW into HIGH and HIGH into LOW. It is also known as logic Inverter & is also sometimes referred to as negation buffer. NOT gate is a single input single output gate.
The logical operator for NOT is '!' and it inverts its operand's value.
The Truth table for NOT gate is:

| **Input** | **Output** |
|---|---|
| **1** | **0** |
| **0** | **1** |

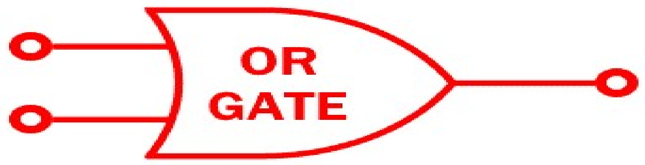
*AND Gate*



This digital logic gate implements the logical AND function, which is the Boolean product of two or more than two variables. AND operation is also known as a logical conjunction. In other words, the output of AND gate is TRUE when all of its inputs are TRUE.
AND function's operator, the logical conjunction is denoted by 'Λ' or '.'.
AND gate has a minimum of two inputs and a single output. The truth table for AND gate is:

| **Input 1** | **Input 2** | **Output** |
|---|---|---|
| **0** | **0** | **0** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

*OR Gate:*

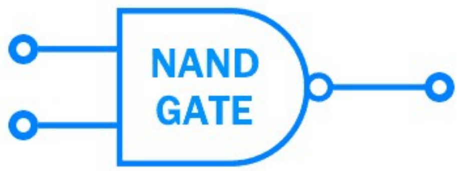


Digital logic OR Gate implements the logical OR function. Logical OR function gives TRUE output if any of its operands are TRUE. Thus, *t*he output of OR gate is True if any of its Input is TRUE.
OR logic function is also known as inclusive disjunction. And its operator is 'V' or '+'.
OR gate has a minimum of two inputs and a single output. The truth table for OR gate is:

| **Input 1** | **Input 2** | **Output** |
|---|---|---|
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |

| | | |
|---|---|---|
| **1** | **1** | **1** |

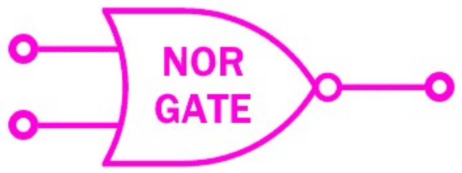
*NAND Gate:*



NAND Gate is a digital logic gate which performs negative AND function. As its name suggests, NAND (NOT of AND) operation inverts the output of AND operation. In other words, the output of the NAND gate is Low only when all of its inputs are high.
NAND gate has a minimum of two inputs and a single output.
NAND gate's truth table is

| **Input 1** | **Input 2** | **Output** |
|---|---|---|
| **0** | **0** | **1** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

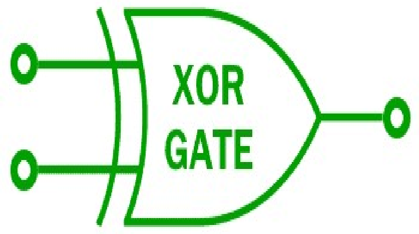
*NOR Gate:*



The digital NOR gate is a Negative-OR gate. The operation of NOR is negation or NOT of logical OR gate. In other words, the output of NOR gate is LOW when any of its input is HIGH and vice versa.
It has a minimum of two inputs and a single output.
The truth table of NOR gate is

| **Input 1** | **Input 2** | **Output** |
|---|---|---|
| **0** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **0** |

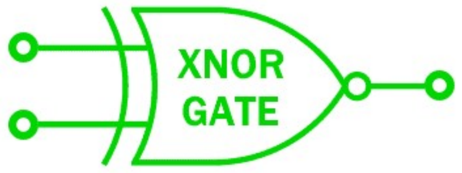
*XOR GATE:*



Exclusive-OR or XOR gate is a digital logic gate used as a parity checker. XOR gate provides output TRUE when the numbers of TRUE inputs are odd. For a two-input XOR gate, the output is TRUE if the inputs are different. Otherwise, the gate will produce FALSE output.
The truth table of XOR gate is following

| **Input 1** | **Input 2** | **Output** |
|---|---|---|
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

*XNOR GATE:*



XNOR gate or Exclusive NOR gate is the negative or inverse of XOR gate. Generally, XNOR gate give output TRUE if it has EVEN number of TRUE inputs.
The output of two-input XNOR gate is TRUE if the inputs are same. When the inputs are different, the 2-input XNOR gate produces FALSE output.
The following is the Truth table for XNOR gate

| **Input 1** | **Input 2** | **Output** |
|---|---|---|
| **0** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

## <u>Compiler and Interpreter</u>

The Compiler and Interpreter, both have similar works to perform. Interpreters and Compilers convert the Source Code (HLL) to Machine Code (understandable by Computer). In general, computer programs exist in High-Level Language that a human being can easily understand. But computers cannot understand the same high-level language, so for computers, we have to convert them into machine language and make them readable for computers. In this article, we are going to see the differences between them.

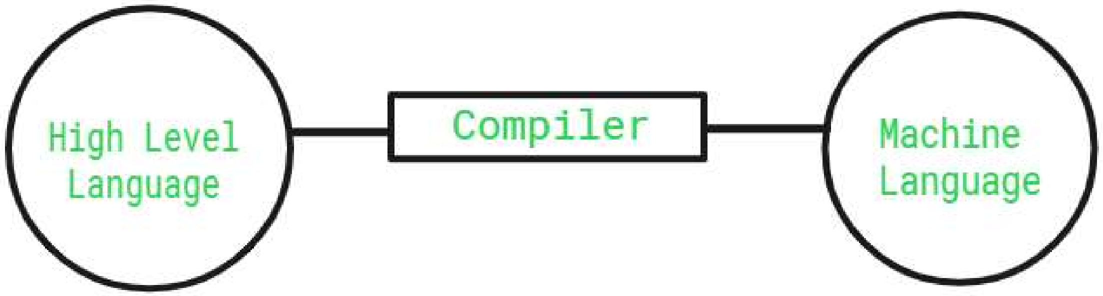
Compiler

The Compiler is a translator which takes input i.e., High-Level Language, and produces an output of low-level language i.e. machine or assembly language. The work of a Compiler is to transform the codes written in the programming language into machine code (format of 0s and 1s) so that computers can understand.

- A compiler is more intelligent than an assembler it checks all kinds of limits, ranges, errors, etc.
- But its program run time is more and occupies a larger part of memory. It has a slow speed because a compiler goes through the entire program and then translates the entire program into machine codes.

Role of a Compiler

For Converting the code written in a high-level language into machine-level language so that computers can easily understand, we use a compiler. Converts basically convert high-level language to intermediate assembly language by a compiler and then assembled into machine code by an assembler.



Advantages of Compiler

- Compiled code runs faster in comparison to Interpreted code.
- Compilers help in improving the security of Applications.
- As Compilers give Debugging tools, which help in fixing errors easily.

Disadvantages of Compiler

- The compiler can catch only syntax errors and some semantic errors.
- Compilation can take more time in the case of bulky code.

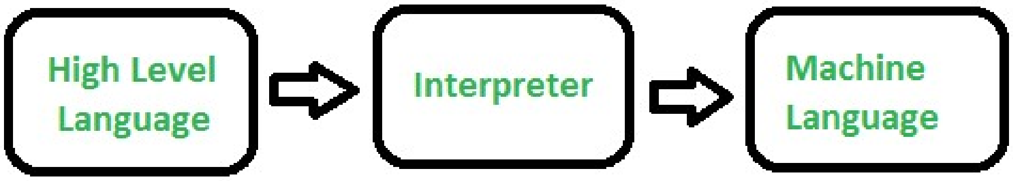
Interpreter

An Interpreter is a program that translates a programming language into a comprehensible language. The interpreter converts high-level language to an intermediate language. It contains pre-compiled code, source code, etc.

- It translates only one statement of the program at a time.
- Interpreters, more often than not are smaller than compilers.

Role of an Interpreter

The simple role of an interpreter is to translate the material into a target language. An Interpreter works line by line on a code. It also converts high level language to machine language.



Advantages of Interpreter

- Programs written in an Interpreted language are easier to debug.
- Interpreters allow the management of memory automatically, which reduces memory error risks.
- Interpreted Language is more flexible than a Compiled language.

Disadvantages of Interpreter

- The interpreter can run only the corresponding Interpreted program.
- Interpreted code runs slower in comparison to Compiled code.

Difference Between Compiler and Interpreter

| Compiler | Interpreter |
|---|---|
| Steps of Programming:<br>• Program Creation.<br>• Analysis of language by the compiler and throws errors in case of any incorrect statement.<br>• In case of no error, the Compiler converts the source code to Machine Code.<br>• Linking of various code files into a runnable program.<br>• Finally runs a Program. | Steps of Programming:<br>• Program Creation.<br>• Linking of files or generation of Machine Code is not required by Interpreter.<br>• Execution of source statements one by one. |
| The compiler saves the Machine Language in form of Machine Code on disks. | The Interpreter does not save the Machine Language. |
| Compiled codes run faster than Interpreter. | Interpreted codes run slower than Compiler. |
| Linking-Loading Model is the basic working model of the Compiler. | The Interpretation Model is the basic working model of the Interpreter. |

| Compiler | Interpreter |
|---|---|
| The compiler generates an output in the form of (.exe). | The interpreter does not generate any output. |
| Any change in the source program after the compilation requires recompiling the entire code. | Any change in the source program during the translation does not require retranslation of the entire code. |
| Errors are displayed in Compiler after Compiling together at the current time. | Errors are displayed in every single line. |
| The compiler can see code upfront which helps in running the code faster because of performing Optimization. | The Interpreter works by line working of Code, that's why Optimization is a little slower compared to Compilers. |
| It does not require source code for later execution. | It requires source code for later execution. |
| Execution of the program takes place only after the whole program is compiled. | Execution of the program happens after every line is checked or evaluated. |
| Compilers more often take a large amount of time for analyzing the source code. | In comparison, Interpreters take less time for analyzing the source code. |
| CPU utilization is more in the case of a Compiler. | CPU utilization is less in the case of a Interpreter. |
| The use of Compilers mostly happens in Production Environment. | The use of Interpreters is mostly in Programming and Development Environments. |
| Object code is permanently saved for future use. | No object code is saved for future use. |
| C, C++, C# etc are programming languages that are compiler-based. | Python , Ruby, Perl, SNOBOL, MATLAB, etc are programming languages that are interpreter-based. |

# Session 3

## Computer Network Architecture

A network model represents the organization of multiple computers in a network. It describes how individual computers are interconnected in a network.

The computer network architecture is as follows −

### Centralized Computing Architecture

In centralized computing architecture, one powerful computer is used to serve one or more low powered computers. In the centralized model, the nodes are not connected; they are only connected to the server.

The centralized computing architecture contains the following −

- All processing holds a function in the central, mainframe computer.
- Terminals are linked to the central computer and operation just as input/output devices.
- Networks can be employed to interconnect at least two mainframe computers. Terminals relate only to the mainframe, never to one another.

### Distributed Computing Architecture

Distributed architecture interconnects one or more personal computers known as nodes. It allows various features such as file sharing, hardware sharing or network sharing. In the distributed model, the nodes can handle their data and depend on the network for administration other than data processing.

For example, a network of Window 2000 can have a computer that is regarded as a Print server. The print server would handle all the printing jobs of the network.

Distributed computing architecture contains the following −

- Various computers are efficient in performing independently
- Tasks are finished locally on multiple computers.
- Networks allow the computers to transfer data and service but do not support processing assistance.

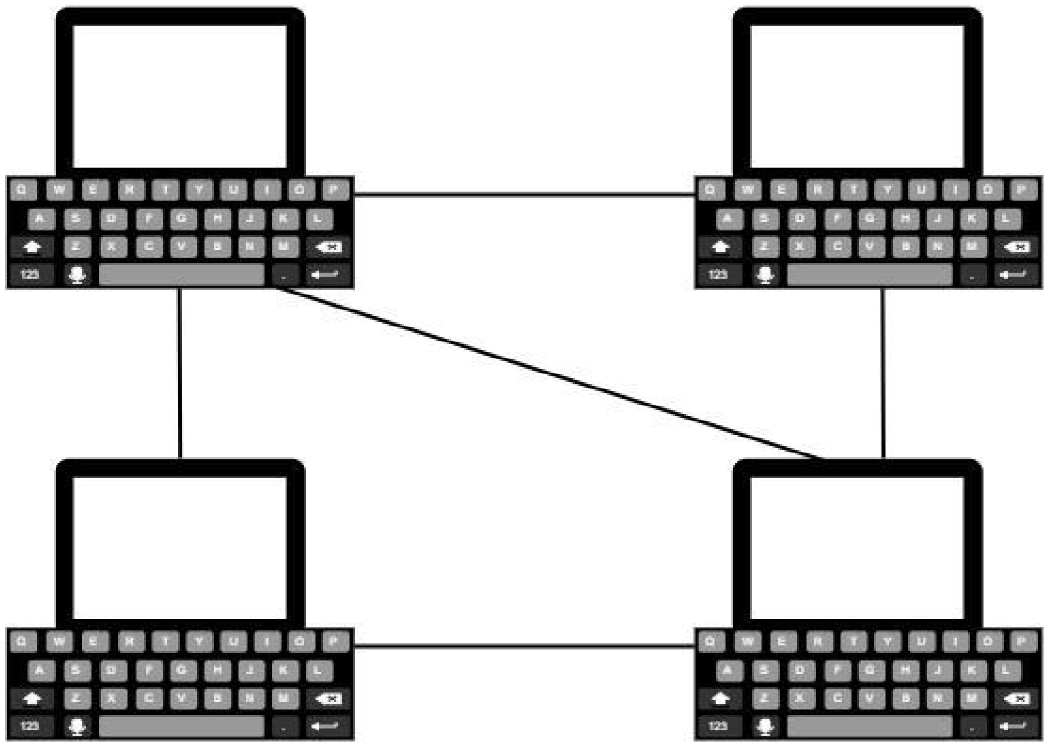
### Collaborative Computing Architecture

The collaborative computing architecture is a mixture of centralized and distributed computing. In the collaborative model, the individual members of a network can process their user's basic need.

For example, a database server such as MSSQL server or ORACLE server sees or handles all the database related processing of all the nodes of networks. But requests other than the database will be processed by the model.

### Peer-to-Peer Architecture

In this architecture, there are no dedicated servers. All computers are similar and, therefore, are termed as a peer. Each of these machines functions both as a user and a server. This arrangement is applicable for environments with a finite number of clients (usually ten or less).



Advantages

- It is accessible to design and perpetuation.
- The network is not dependent on a precise computer.
- Linking new computers in a peer to peer network is significantly more straightforward.

What Do Computer Networks Do?

Computer Networks are one of the important aspects of Computer Science. In the early days, it is used for data transmission on telephone lines and had a very limited use, but nowadays, it is used in a variety of places.

Computer Networks help in providing better connectivity that helps nowadays. Modern computer networks have the following functionality like

- Computer Networks help in operating virtually.
- Computer Networks integrate on a large scale.
- Computer Networks respond very quickly in case of conditions change.
- Computer Networks help in providing data security.

Criteria of a Good Network

1. **Performance:** It can be measured in many ways, including transmit time and response time. Transit time is the amount of time required for a message to travel from one device to another. Response time is the elapsed time between an inquiry and a response. The performance of the network depends on a number of factors, including the number of users, the type of medium & Hardware
2. **Reliability:** In addition to accuracy is measured by frequency of failure, the time it takes a link to recover from failure, and the network's robustness in catastrophe.
3. **Security:** Network security issues include protecting data from unauthorized access, protecting data from damage and development, and implementing policies and procedures for recovery from breaches and data loss.

Goals of Computer Networking

- Programs do not have to execute on a single system because of resource and load sharing.
- Reduced costs – Multiple machines can share printers, tape drives, and other peripherals.

- Reliability – If one machine fails, another can take its place.
- Scalability (it's simple to add more processors or computers)
- Communication and mail (people living apart can work together)
- Information Access (remote information access, access to the internet, e-mail, video conferencing, and online shopping)
- Entertainment that is interactive (online games, videos, etc.)
- Social Networking

Types of Computer Networks

**Division Based on the Communication Medium**

- **Wired Network:** As we all know, "wired" refers to any physical medium made up of cables. Copper wire, twisted pair, or fiber optic cables are all options. A wired network employs wires to link devices to the Internet or another network, such as laptops or desktop PCs.
- **Wireless Network:** "Wireless" means without wire, media that is made up of electromagnetic waves (EM Waves) or infrared waves. Antennas or sensors will be present on all wireless devices. Cellular phones, wireless sensors, TV remotes, satellite dish receivers, and laptops with WLAN cards are all examples of wireless devices. For data or voice communication, a wireless network uses radio frequency waves rather than wires.

**Division Based on Area Covered**

- **Local Area Network (LAN):** A LAN is a network that covers an area of around 10 kilometers. For example, a college network or an office network. Depending upon the needs of the organization, a LAN can be a single office, building, or Campus. We can have two PCs and one printer in-home office or it can extend throughout the company and include audio and video devices. Each host in LAN has an identifier, an address that defines hosts in LAN. A packet sent by the host to another host carries both the source host's and the destination host's address.
- **Metropolitan Area Network (MAN): MAN** refers to a network that covers an entire city. For example: consider the cable television network.
- **Wide Area Network (WAN):** WAN refers to a network that connects countries or continents. For example, the Internet allows users to access a distributed system called www from anywhere around the globe. WAN interconnects connecting devices such as switches, routers, or modems. A LAN is normally privately owned by an organization that uses it. We see two distinct examples of WANs today: point-to-point WANs and Switched WANs
    - **Point To Point**: Connects two connecting devices through transmission media.
    - **Switched:** A switched WAN is a network with more than two ends.

**Based on Types of Communication**

- **Point To Point networks:** Point-to-Point networking is a type of data networking that establishes a direct link between two networking nodes.
  A direct link between two devices, such as a computer and a printer, is known as a point-to-point connection.
- **Multipoint**: is the one in which more than two specific devices share links. In the multipoint environment, the capacity of the channel is shared, either spatially or temporally. If several devices can use the link simultaneously, it is a spatially shared connection.
- **Broadcast networks:** In broadcast networks, a signal method in which numerous parties can hear a single sender. Radio stations are an excellent illustration of the "Broadcast Network" in everyday life. The radio station is a sender of data/signal in this scenario, and data is only intended to travel in one direction. Away from the radio transmission tower, to be precise.

**Based on the Type of Architecture**

- **P2P Networks:** Computers with similar capabilities and configurations are referred to as peers.
  "Peer to Peer" is the abbreviation for "peer to peer." The "peers" in a peer-to-peer network are computer systems that are connected to each other over the Internet. Without the use of a central server, files can be shared directly between systems on the network.
- **Client-Server Networks:** Each computer or process on the network is either a client or a server in a client-server architecture (client/server). The client asks for services from the server, which the server provides. Servers are high-performance computers or processes that manage disc drives (file servers), printers (print servers), or network traffic (network servers)
- **Hybrid Networks:** The hybrid model refers to a network that uses a combination of client-server and peer-to-peer architecture. Eg: Torrent.

Types of Computer Network Architecture

Computer Network Architecture is of two types. These types are mentioned below.

**1. Client-Server Architecture: It** is basically the architecture where the clients and the server are connected as two clients can communicate with each other and the devices present work as servers in the network.

**2. Peer-to-Peer Architecture:** computers are connected to each other and each computer is equally capable of working as there is no central server here. Each device present here can be used as a client or server.

Types of Enterprise Computer Networks

There are three main types of Enterprise Computer Networks which are mentioned below.

**1. Local Area Network (LAN): These** are small-scale networks used in small companies or as test networks. It has a limited size.

**2. Wide Area Networks (WAN): These** are networks that are used for a larger area than local area networks and are used for long-distance communication.

**3. Service Provider Networks:** Service Provider Networks are the networks that help in wireless communication, high-speed internet access, etc.

Network Topology

**Bus Topology**

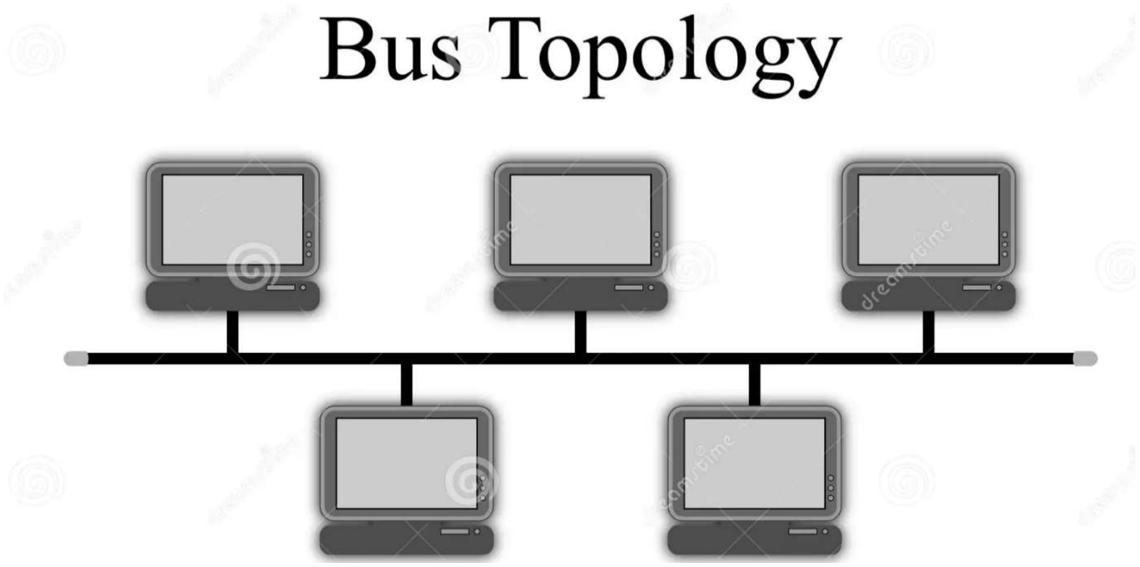
Every computer and network device is connected to a single cable in a bus topolgy network. Linear Bus topology is defined as having exactly two terminals.

**Advantages**

- Installation is simple.
- Compared to mesh, star, and tree topologies, the bus utilizes less cabling.

**Disadvantages**

- Difficulty in reconfiguring and isolating faults.
- A bus cable malfunction or break interrupts all communication.



*Bus Topology*

**Ring Topology**

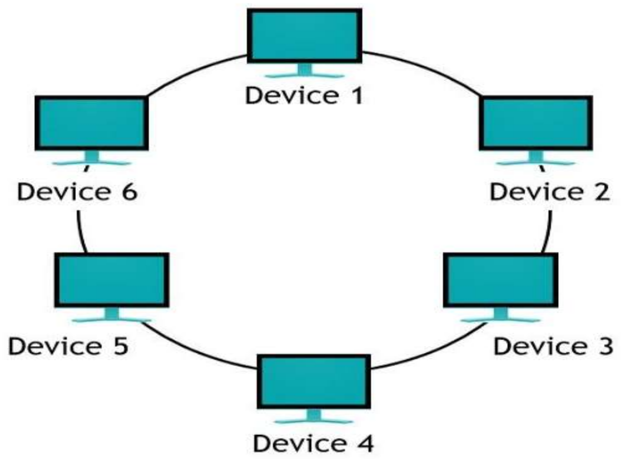
The topology is named ring topology because one computer is connected to another, with the final one being connected to the first. Exactly two neighbors for each device. A signal is passed along the ring in one direction. Each ring incorporates a repeater.

**Advantages**

- Data transmission is relatively straightforward because packets only move in one direction.
- There is no requirement for a central controller to manage communication between nodes.
- Easy installation & Reconfiguration
- Simplified Faulty connections

**Disadvantages**

- In a Unidirectional Ring, a data packet must traverse through all nodes.
- All computers must be turned on in order for them to connect with one another.



*Ring Topology*

**Star Topology**

Each device in a star topology has a dedicated point-to-point link to a central controller, which is commonly referred to as the HUB. There is no direct connection between the devices. Traffic between the devices is not allowed in this topology. As an exchange, the controller is used.

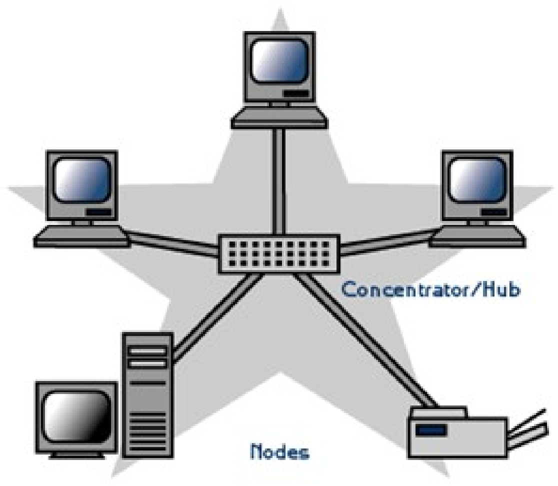
**Advantages**

- When attaching or disconnecting devices, there are no network interruptions.
- It's simple to set up and configure.
- Identifying and isolating faults is simple.
- Less Expensive than mesh
- Easy to install & configure

**Disadvantages**

- Nodes attached to the hub, switch, or concentrator is failed if they fail.
- Because of the expense of the hubs, it is more expensive than linear bus topologies.
- More cable is required compared to a bus or ring
- Too much dependency on Hub

.



*Star Topology*

**Mesh Topology**

Every device in a mesh topology has dedicated point-to-point connectivity to every other device. The term "dedicated" refers to the fact that the link exclusively transports data between the two devices it links. To connect n devices, a fully connected mesh network contains n *(n-1)/2 physical channels.

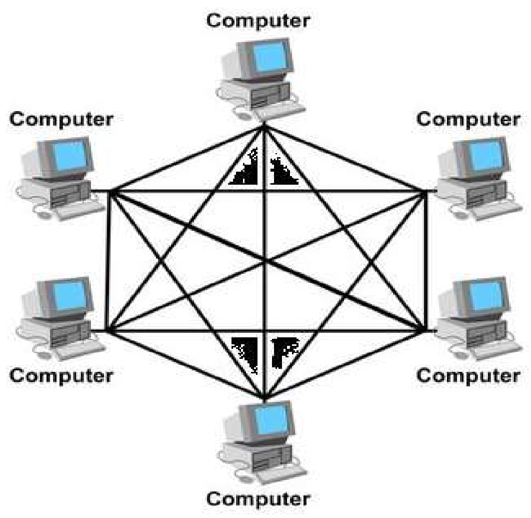
**Advantages**

- Data can be sent from multiple devices at the same time. This topology can handle a lot of traffic.
- Even if one of the connections fails, a backup is always available. As a result, data transit is unaffected.
- Physical boundaries prevent other users from gaining access to messages
- Point to Point links make fault transmission & fault isolation easy

**Disadvantages**

- The amount of cabling and the number of I/O ports that are necessary.
- The sheer bulk of wiring can be greater than the available space can accommodate.
- It is difficult to install and reconfigure.

**Example:** connection of telephone regional office in which each regional office needs to be connected to every other regional office.



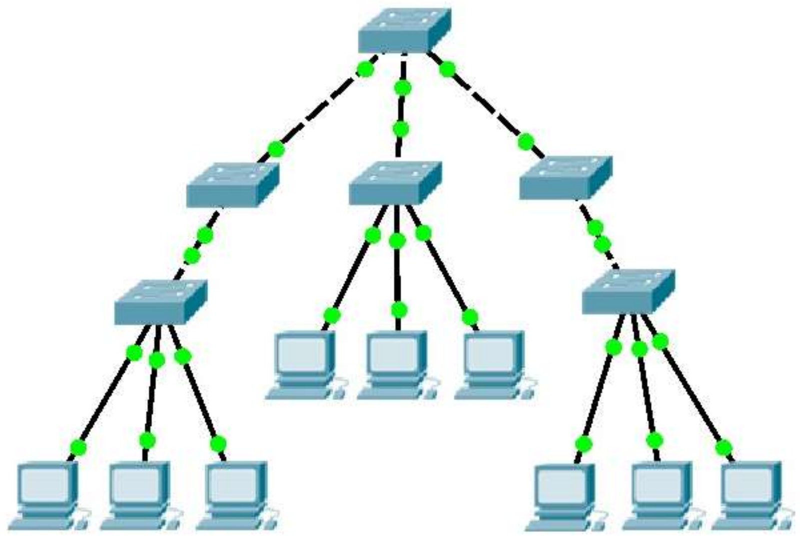
*Mesh Topology*

**Tree Topology**
The topology of a tree is similar to that of a star. Nodes in a tree, like those in a star, are connected to a central hub that manages network traffic. It has a root node, which is connected to all other nodes, producing a hierarchy. Hierarchical topology is another name for it. The number of Star networks is connected via Bus in Tree Topology.
**Advantages**

- Network expansion is both possible and simple.
- We partition the entire network into pieces (star networks) that are easier to manage and maintain.
- Other segments are unaffected if one segment is damaged.

**Disadvantages**

- Tree topology relies largely on the main bus cable because of its basic structure, and if it fails, the entire network is handicapped.
- Maintenance becomes more challenging when more nodes and segments are added.


*Tree Topology*

Networking Devices
Basic hardware interconnecting network nodes, such as Network Interface Cards (NICs), Bridges, Hubs, Switches, and Routers, are used in all networks. In addition, a mechanism for connecting these building parts is necessary, which is usually galvanic cable and optical cable are less popular ("optical fiber")The following are the network devices :

- **NIC (Network Interface Card):** A network card, often known as a network adapter or NIC (network interface card), is computer hardware that enables computers to communicate via a network. It offers physical access to networking media and, in many cases, MAC addresses serve as a low-level addressing scheme. Each network interface card has a distinct identifier. This is stored on a chip that is attached to the card.
- **Repeater:** A repeater is an electrical device that receives a signal, cleans it of unwanted noise, regenerates it, and retransmits it at a higher power level or to the opposite side of an obstruction, allowing the signal to travel greater distances without degradation. In the majority of twisted pair Ethernet networks, Repeaters are necessary for cable lengths longer than 100 meters in some systems. Repeaters are based on physics.
- **Hub:** A hub is a device that joins together many twisted pairs or fiber optic Ethernet devices to give the illusion of a formation of a single network segment. The device can be visualized as a multiport repeater. A network hub is a relatively simple broadcast device. Any packet entering any port is regenerated and broadcast out on all other ports, and hubs do not control any of the traffic that passes through them. Packet collisions occur as a result of every packet being sent out through all other ports, substantially impeding the smooth flow of communication.
- **Bridges:** Bridges broadcast data to all the ports but not to the one that received the transmission. Bridges, on the other hand, learn which MAC addresses are reachable through specific ports rather than copying messages to all ports as hubs do. Once a port and an address are associated, the bridge will only transport traffic from that address to that port.
- **Switches:** A switch differs from a hub in that it only forwards frames to the ports that are participating in the communication, rather than all of the ports that are connected. The collision domain is broken by a switch, yet the switch depicts itself as a broadcast domain. Frame-forwarding decisions are made by switches based on MAC addresses.
- **Routers:** are networking devices that use headers and forwarding tables to find the optimal way to forward data packets between networks. A router is a computer networking device that links two or more computer networks and selectively exchanges data packets between them. A router can use address information in each data packet to determine if the source and destination are on the same network or if the data packet has to be transported between networks. When numerous routers are deployed in a wide collection of interconnected networks, the routers share target system addresses so that each router can develop a table displaying the preferred pathways between any two systems on the associated networks.
- **Gateways:** To provide system compatibility, a gateway may contain devices such as protocol translators, impedance-matching devices, rate converters, fault isolators, or signal translators. It also necessitates the development of administrative procedures that are acceptable to both networks. By completing the necessary protocol conversions, a protocol translation/mapping gateway joins networks that use distinct network protocol technologies.

**Internet**
This is a larger network that allows computer networks controlled by enterprises, governments, colleges, and other organizations all over the world to communicate with one another. As a result, there is a tangle of cables, computers, data centers, routers, servers, repeaters, satellites, and Wi-Fi towers that allow digital data to go around the world.
The Internet is a vast network of networks that functions as a networking infrastructure. It links millions of computers throughout the world, creating a network in which any computer can talk with any other computer as long as they are both linked to the Internet.

The Internet is a global network of interconnected computers that communicate and share information using a standardized Internet Protocol Suite.

## Basic Internet Terms

Intranet

An Intranet is a private network of an organisation. It can be accessed by users authorised by the organisation. The purpose of the Intranet is to share information among authorised users only. The network is based on TCP/IP protocols. Intranet may be within a LAN or over a WAN via Internet. Igloo and Huddle are some Intranet services.

URL

URL is used to specify addresses on the World Wide Web. A URL is the fundamental network identification for any resources available on the web ( for example , hypertext pages , images and sound files ). A URL has the following format : *protocol://hostname/path* . It is unique in nature.

ISP

ISPs are the companies who provide services in the terms of the Internet connection to connect the web. ISPs also have the option of providing services like website hosting and development, email hosting and domain name registration. ISPs can either be regional or national.

IP Address

IP address is short form of Internet Protocol address. It uniquely identifies a computer or device on a TCP/IP network. IPv4 and IPv6 are the two IP addressing standards in use. IPv4 address is four bytes (32 bits) long. IT is written as four numbers separated by periods in the range 0.0.0.0 to 255.255.255.255. For example, 1.161.10.235 is IPv4 address. IPv6 address is 16 bytes (128 bits) long.

DNS(Domain Name Server)

Domain Name is a string of words for the numeric IP address of a computer on the Internet. Domain Name is used for convenience of the user, for example, google.com and yahoo.com. DNS server is a computer having a database that stores the IP addresses and their domain names. When an user uses the domain name, DNS translates it into its corresponding IP address, to access the computer in Internet. For example, DNS translates google.com to the IP address of the computer that houses Google.

# Session -4

## Basics of Web Design and Web Technology

### History of Web Design

- The history of web design began with the introduction of tables in websites to make the complex-looking structure simpler in the mid-1990s.
- Afterward, CSS was introduced in the late 1990s which separated the presentation part of the website from its structure and made websites more maintainable.
- In the early 2000s web standards were defined which helped in making websites look better and user friendly.
- Since the 2010s responsive web designs have been promoted so that websites can become accessible on various screens.
- In mid 2010s animations in websites became a trend that enhanced User Experience and engagement.
- In modern days website design a mobile-first approach is being preferred and various other user-centered designs like dark mode, and 3D elements are being added.

### What is web design?

Web design is the art of planning and arranging content on a website so that it can be shared and accessed online with the world. Being a combination of aesthetic and functional elements, Web design is a type of digital design that determines the look of a website—such as its colours, fonts, graphics and user interface.

Today, creating a website is one of the pillars of having an online presence. Because of this, the world of web design is as dynamic as ever. It is constantly evolving, including mobile apps and user interface design, to meet the growing needs of website owners and visitors alike.

Website - A website is a collection of files accessed through a web address, covering a particular theme or subject, and managed by a particular person or organization. Its opening page is called a home page.

It is very important to clarify the difference between web design and website development, since the two are closely related and often (mistakenly) used interchangeably:

**Web design** refers to the visual design and experiential aspects of a particular website. It is a field related to designing websites on the Internet. Without a good design, a website fails to perform well and cannot produce a good impact on the user. A good website design helps to create an engaging online presence that captivates the audience.
**Website development** refers to the building and maintenance of a website's structure, and involves intricate coding systems that ensure the website functions properly.

## Types of Web design

The following are the types in which you can do Web Design:
**Static website :** This type of design is used when little or no interaction is required from the user.
**Dynamic Website :** When user interaction is required and information is to be displayed according to the request then a dynamic design pattern is followed.
**E-Commerce Website:** This type of website design is required when a business wants to sell their products to the consumer.
**Flat Design :** Minimalist approach characterized by clean, simple elements, vibrant colours, and absence of textures or gradients.
**Neomorphic Design :** Mimics physical interactions and textures, creating interfaces that blend realism with digital functionality.
**Minimalism Design :** Focuses on stripping away unnecessary elements, favoring simplicity, clean lines, and ample white space for an uncluttered user experience.

*[ Subject teacher is being requested to introduce the students with some common websites for giving a demonstration of the types of web design ]*

## Work of a Web Designer

A web designer works on a **website's appearance, layout, and,** in some cases, **content**.

- **Appearance** relates to the colours, typography, and images used.
- **Layout** refers to how information is structured and categorized. A good web design is easy to use, aesthetically pleasing, and suits the user group and brand of the website.
- A well-designed website is simple and **communicates clearly** to avoid confusing users. It wins and fosters the target audience's trust, removing as many potential points of user frustration as possible.

The role of a web designer entails the task of designing a website's visual design and layout of a website, which includes the site's appearance, structure, navigation, and accessibility. They select colour palettes, create graphics, choose fonts, and layout content to create an aesthetically pleasing, user-friendly, and accessible design. Web designers also work closely with web developers to verify that the design is technically feasible and implemented correctly. They may be involved in user experience design, ensuring the website is intuitive, accessible, and easy to use.

## Some Basic Terms explained

### Web Server

A Web Server can be referred to as either the hardware ( the computer ) or the software ( the computer application ) that helps to deliver content that can be accessed through the internet. The primary function of a Web Server is to deliver web pages on request to the clients, to host websites, store data or run centralized applications. A Web Browser ( or client ) requests for a specific resource using HTTP and the server responds either with the content or an error message if unable to do so. It may also receive and store online forms and uploaded files. It may also store mails ( Mail Server ), share files ( File Server ) or share Printers ( Print Server). Major Web Servers are Netscape's iPlanet, Bea's Web Logic and IBM's Websphere. Some others include Apache HTTP Server, Microsoft's Internet Information Services, Sun Java System Web Server and Jigsaw Web Server.

**Web Browser**

A Web Browser is a computer program used for accessing websites or information on a Network or on the www. This helps us to view web pages on the Internet that contain hypertext documents. It also helps us to jump from one website to the other using links. The Web address, or the URL ( Uniform Resource Locator), that you type into the browser's address bar tells the browser where to obtain a page or pages from. It can also display a webpage if the address is typed in the browser's address bar. The following are some popular Web Browsers.

Internet explorer , Netscape , Firefox , Safari , Opera , Lynx , Chrome.

**Webpages**

A Web Page is a document, typically written in Hypertext Markup Language ( HTML) and Extensible HTML. Webpages are accessed and transported using the Hypertext Transfer Protocol (HTTP) or HTTP Secure (HTTPS) to provide security and privacy for the user of the webpage content. Many such webpages make a website. Web Pages may be static or dynamic. A Web Page may have a hyperlink to another website.

**Website**

A Website is a collection of various inter-related pages written in HTML. Each page available on the website is called a webpage and the first page of any website is called the homepage of that site. A Website should be hosted on at least one Web server.
Some advantages of a Website are :

- 24 hour global presence
- Increased customer base
- Tremendous cost saving

**Web Portal**

A Web portal is a website that presents information from different sources and makes them available in a unified way. A user can search for any type of information from a single location, that is , the home page of the Web portal. A Web portal generally consists of search engine, Email service, News , Advertisements , an extensive list of links to other sites etc.
www.msn.com and www.google.co.in are popular Web Portals.

**Link**

The collection of documents on WWW ( World Wide Web ) are inter-connected with each other via links. The linked documents may be located on one or more than one computer, worldwide.

**Hypertext**

Hypertext is text that has links to other texts. Hypertext format is used to create documents on the web. HTML is the language used to create a hypertext format document. A HTML can include text, pictures, video, images, sound, graphics, movies etc, and linked contents on the same or different documents using a hyperlink .

**Hyperlink**

A Hyperlink or simply a link is a selectable element in a hypertext document that serves as an access point to other pages. Typically, you click the hyperlink to access the linked resource. Once a page has loaded, move your mouse pointer over different areas of the page. The cursor will change from an arrow to a hand with the pointing finger when it is over a Hypertext link. Familiar hyperlinks include buttons, icons, image, maps and clickable text links.

**Bandwidth**

Bandwidth is the amount of data that can be transferred through a communication medium in a fixed amount of time. Bandwidth is measured in cycles per second [cps] or Hertz [Hz]. The speed of internet access increases with the increase in bandwidth. Broadband are the services with more bandwidth. Modem and Integrated Service Digital Network (ISDN) are some broadband connections.

## What Is Visual Design & Why Do You Need It?

In general, visual design is concerned with composing aesthetic visuals that achieve a desired aesthetic effect. You can find it in many fields and industries, but the term is most commonly used for design in visual communications. Web pages, landing pages, logos, infographics, icons and other UI elements, are all created to communicate with the user visually.

## Space

If you make a shape and color it in, it creates a space that’s called positive space. You can think of it as space that’s taken up by something. The space that remains untaken is called negative space. Dealing with negative space is one of the key basic principles of visual design.

## Visual Typography

The fonts you choose play a big role in how well you work. Their size, color, alignment, and spacing all need to come together to achieve the desired effect.

## Unity

When elements are arranged so that it appears like they belong together, it’s called unity. Unity is good because it allows elements to form groups (words are groups of elements that are in unity). However, if you have too much unity, the visual design might appear dull.

## Scale

The scale is the relative size of the elements. Playing around with it will show you how it affects many of the other principles.

## Hierarchy

Some elements are more important than others. By using different sizes, colors, and placements of elements, you can form a visual hierarchy. That way, the important elements will grab the attention first.

Visual design, or "look and feel", is the use of color, shapes, typography, and imagery on a website. It's similar to the curb appeal of a home and can evoke a visceral reaction, either positive or negative.

**Some key elements of effective web design include:**

- User-friendly design
- Mobile responsiveness
- A focus on SEO
- Speed
- Compelling content
- Calls-to-action
- Aesthetically pleasing imagery

## What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

## A Simple HTML Document

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading
- The <p> element defines a paragraph

## What is an HTML Element?

An HTML element is defined by a start tag, some content, and an end tag:

```
<tagname> Content goes here... </tagname>
```

The HTML element is everything from the start tag to the end tag:

<h1>My First Heading</h1>

<p>My first paragraph.</p>

| Start tag | Element content | End tag |
|---|---|---|
| <h1> | My First Heading | </h1> |
| <p> | My first paragraph. | </p> |
| <br> | none | none |

## HTML Attributes

- All HTML elements can have attributes
- Attributes provide additional information about elements
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: name="value"
- All HTML elements can have attributes
- The href attribute of <a> specifies the URL of the page the link goes to
- The src attribute of <img> specifies the path to the image to be displayed
- The width and height attributes of <img> provide size information for images
- The alt attribute of <img> provides an alternate text for an image
- The style attribute is used to add styles to an element, such as color, font, size, and more
- The lang attribute of the <html> tag declares the language of the Web page
- The title attribute defines some extra information about an element

## HTML Headings

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

```
<h1>Heading 1</h1>

<h2>Heading 2</h2>

<h3>Heading 3</h3>

<h4>Heading 4</h4>

<h5>Heading 5</h5>

<h6>Heading 6</h6>
```

## HTML Paragraphs

The HTML <p> element defines a paragraph.

A paragraph always starts on a new line, and browsers automatically add some white space (a margin) before and after a paragraph.

```
<p>This is a paragraph.</p>

<p>This is another paragraph.</p>
```

The **<p>** tag in HTML defines a paragraph. These have both opening and closing tags. So anything mentioned within **<p>** and **</p>** is treated as a paragraph. A paragraph is a block-level element so a new paragraph always begins on a new line, and browsers naturally put some space before and after a paragraph to make it look neat and easy to read.

### HTML Styles

Use the style attribute for styling HTML elements

- Use background-color for background color
- Use color for text colors
- Use font-family for text fonts
- Use font-size for text sizes
- Use text-align for text alignment

The HTML style attribute is used to add styles to an element, such as color, font, size, and more.

### HTML Text Formatting

HTML contains several elements for defining text with a special meaning.

### HTML Formatting Elements

Formatting elements were designed to display special types of text:

- <b> - Bold text
- <strong> - Important text
- <i> - Italic text
- <em> - Emphasized text
- <mark> - Marked text
- <small> - Smaller text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

## HTML Quotation and Citation Elements

In this chapter we will go through the <blockquote>,<q>, <abbr>, <address>, <cite>, and <bdo> HTML elements.

| **Tag** | **Description** |
|---|---|
| <abbr> | Defines an abbreviation or acronym |
| <address> | Defines contact information for the author/owner of a document |
| <bdo> | Defines the text direction |
| <blockquote> | Defines a section that is quoted from another source |
| <cite> | Defines the title of a work |
| <q> | Defines a short inline quotation |

## HTML Styles - CSS

CSS stands for Cascading Style Sheets.

CSS saves a lot of work. It can control the layout of multiple web pages all at once.

## What is CSS?

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

## Using CSS

CSS can be added to HTML documents in 3 ways:

- Inline - by using the style attribute inside HTML elements
- Internal - by using a <style> element in the <head> section
- External - by using a <link> element to link to an external CSS file

The most common way to add CSS, is to keep the styles in external CSS files. However, in this tutorial we will use inline and internal styles, because this is easier to demonstrate, and easier for you to try it yourself.

## CSS Solved a Big Problem

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to describe the content of a web page, like:

```
<h1>This is a heading</h1>

<p>This is a paragraph.</p>
```

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page.

## Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powder blue" background color:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
```

```
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

## External CSS

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet, add a link to it in the <head> section of each HTML page:

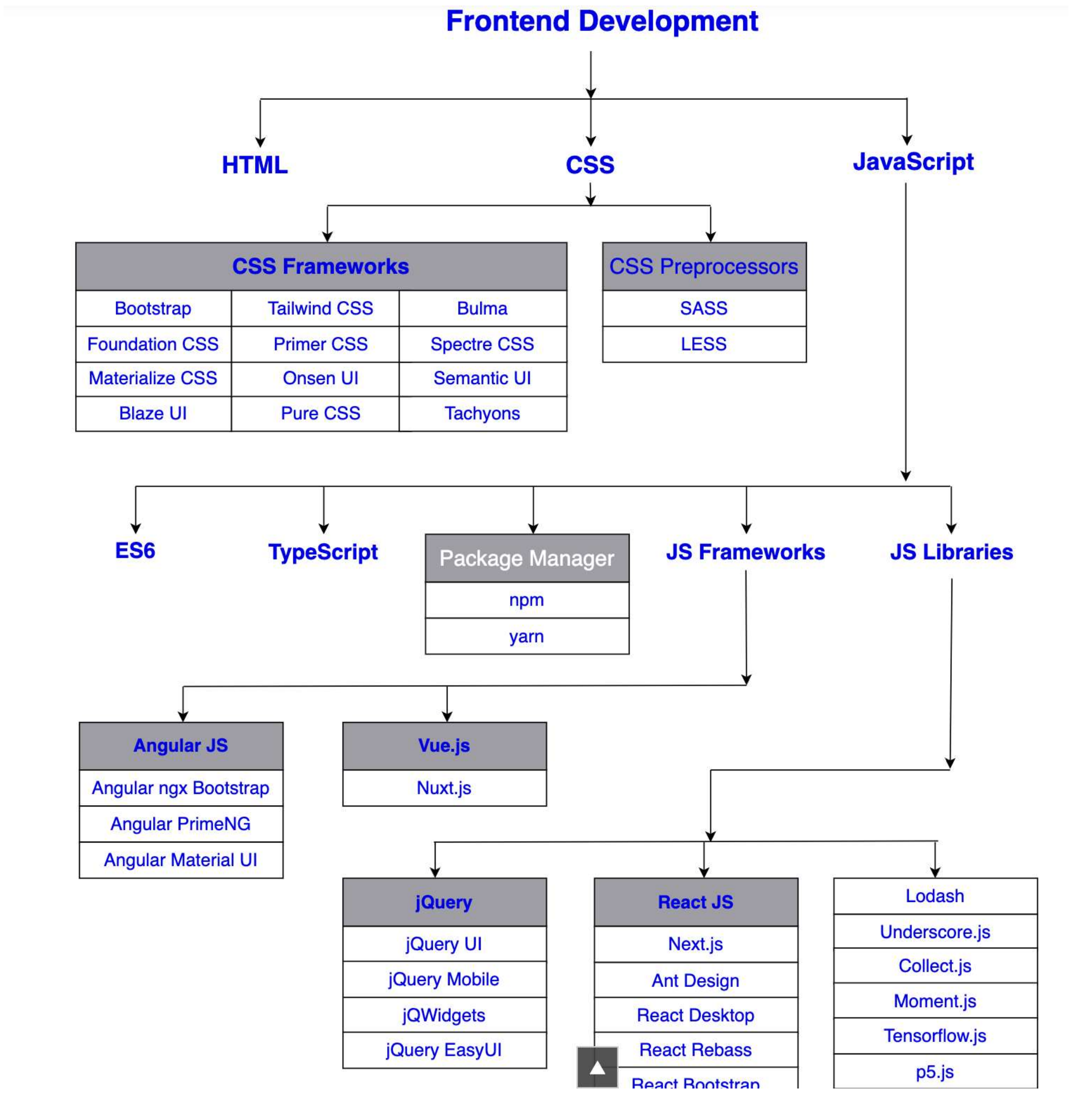
```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

- Use the HTML style attribute for inline styling
- Use the HTML <style> element to define internal CSS
- Use the HTML <link> element to refer to an external CSS file
- Use the HTML <head> element to store <style> and <link> elements
- Use the CSS color property for text colors
- Use the CSS font-family property for text fonts
- Use the CSS font-size property for text sizes
- Use the CSS border property for borders
- Use the CSS padding property for space inside the border
- Use the CSS margin property for space outside the border

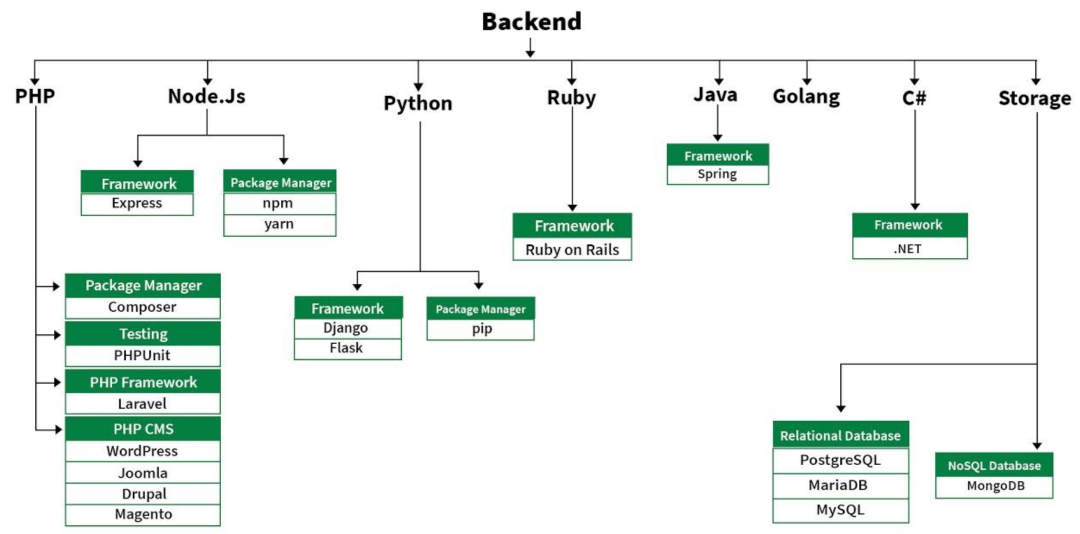
Keywords related to Web Technology.

## Web Development can be Classified into Two Ways:

- **Frontend Development:** The part of a website that the user interacts directly is termed as front end. It is also referred to as the 'client side' of the application.



**Backend Development:** Backend is the server side of a website. It is the part of the website that users cannot see and interact with. It is the portion of software that does not come in direct contact with the users. It is used to store and arrange data.



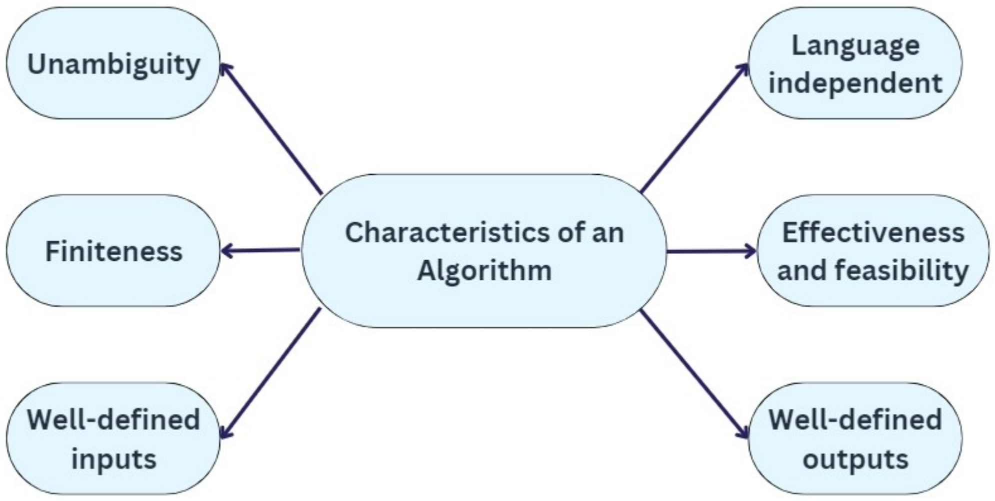
# Session 5

## Concept of Algorithm and Flowchart

The algorithm and flowchart are two types of tools to explain the process of a program. Algorithms and flowcharts are two different tools that are helpful for creating new programs, especially in computer programming. An algorithm is a step-by-step analysis of the process, while a flowchart explains the steps of a program in a graphical way.

Writing a logical step-by-step method to solve the problem is called the algorithm. In other words, an algorithm is a procedure for solving problems. In order to solve a mathematical or computer problem, this is the first step in the process.

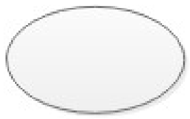
An algorithm includes calculations, reasoning, and data processing. Algorithms can be presented by natural languages, pseudocode, and flowcharts, etc.



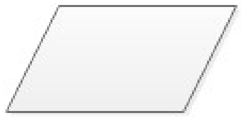
A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes, and arrows to demonstrate a process or a program. With algorithms, we can easily understand a program.
The main purpose of using a flowchart is to analyze different methods. Several standard symbols are applied in a flowchart:

Common Abbreviations Used in flowchart :

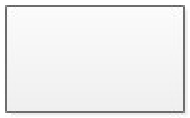
Terminal Box - Start / End



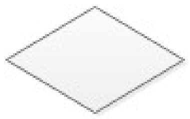
Input / Output



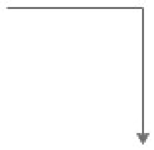
Process / Instruction



Decision



Connector / Arrow



The symbols above represent different parts of a flowchart. The process in a flowchart can be expressed through boxes and arrows with different sizes and colors. In a flowchart, we can easily highlight certain elements and the relationships between each part.

## Difference between Algorithm and Flowchart

If you compare a flowchart to a movie, then an algorithm is the story of that movie. In other words, an algorithm is the core of a flowchart. Actually, in the field of computer programming, there are many differences between algorithms and flowchart regarding various aspects, such as the accuracy, the way they display, and the way people feel about them. Below is a table illustrating the differences between them in detail.

Algorithm

- It is a procedure for solving problems.
- The process is shown in step-by-step instruction.
- It is complex and difficult to understand.
- It is convenient to debug errors.
- The solution is showcased in natural language.
- It is somewhat easier to solve complex problems.
- It costs more time to create an algorithm.

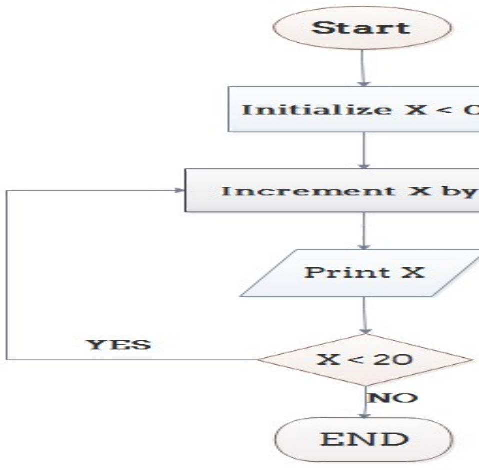
Flowchart

- It is a graphic representation of a process.
- The process is shown in a block-by-block information diagram.
- It is intuitive and easy to understand.
- It is hard to debug errors.
- The solution is showcased in pictorial format.
- It is hard to solve complex problems.
- It costs less time to create a flowchart.

Example 1: Print 1 to 20:
Algorithm:

- Step 1: Initialize X as 0,
- Step 2: Increment X by 1,
- Step 3: Print X,
- Step 4: If X is less than 20 then go back to step 2.

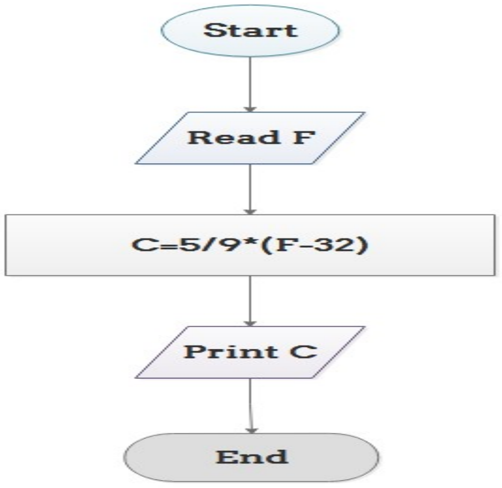
Flowchart:



Example 2: Convert Temperature from Fahrenheit (°F) to Celsius (°C)
Algorithm:

- Step 1: Read temperature in Fahrenheit,
- Step 2: Calculate temperature with formula C=5/9*(F-32),
- Step 3: Print C.

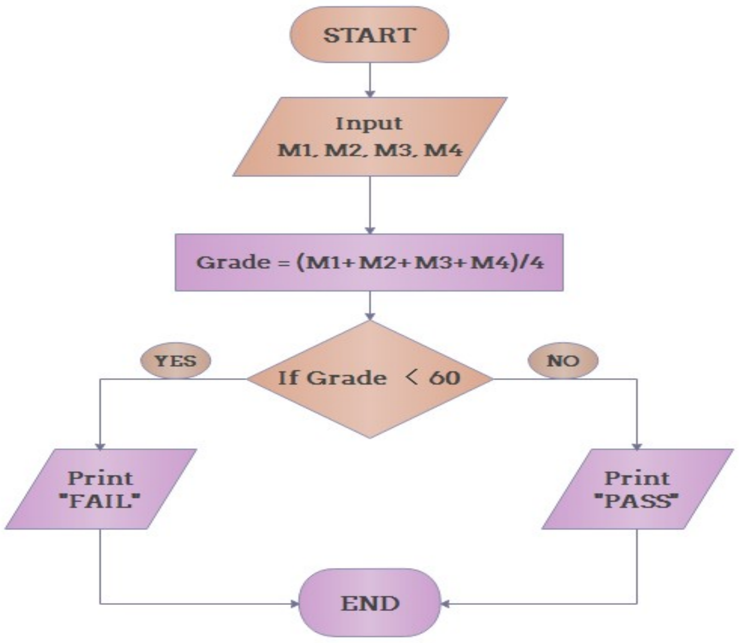
Flowchart:



Example 3: Determine Whether A Student Passed the Exam or Not:
Algorithm:

- Step 1: Input grades of 4 courses M1, M2, M3 and M4,
- Step 2: Calculate the average grade with formula "Grade=(M1+M2+M3+M4)/4"
- Step 3: If the average grade is less than 60, print "FAIL", else print "PASS".

Flowchart:



# Basics of Computer Programming

## Programming Language

As we know, to communicate with a person, we need a specific language, similarly to communicate with computers, programmers also need a language called Programming language.

Before learning the programming language, let's understand what language is?

## What is Language?

Language is a mode of communication that is used to share ideas, opinions with each other. For example, if we want to teach someone, we need a language that is understandable by both communicators.

## What is a Programming Language?

A programming language is a computer language that is used by programmers (developers) to communicate with computers. It is a set of instructions written in any specific language ( C, C++, Java, Python) to perform a specific task.

A programming language is mainly used to develop desktop applications, websites, and mobile applications.

## Types of programming language

### 1. Low-level programming language

Low-level language is machine-dependent (0s and 1s) programming language. The processor runs low- level programs directly without the need of a compiler or interpreter, so the programs written in low-level language can be run very fast.

Low-level language is further divided into two parts -

i. Machine Language

Machine language is a type of low-level programming language. It is also called as machine code or object code. Machine language is easier to read because it is normally displayed in binary or hexadecimal form (base 16) form. It does not require a translator to convert the programs because computers directly understand the machine language programs.

The advantage of machine language is that it helps the programmer to execute the programs faster than the high-level programming language.

ii. Assembly Language

Assembly language (ASM) is also a type of low-level programming language that is designed for specific processors. It represents the set of instructions in a symbolic and human-understandable form. It uses an assembler to convert the assembly language to machine language.

The advantage of assembly language is that it requires less memory and less execution time to execute a program.

### 2. High-level programming language

High-level programming language (HLL) is designed for developing user-friendly software programs and websites. This programming language requires a compiler or interpreter to translate the program into machine language (execute the program).

The main advantage of a high-level language is that it is easy to read, write, and maintain.

High-level programming language includes Python, Java, JavaScript, PHP, C#, C++, Objective C, Cobol, Perl, Pascal, LISP, FORTRAN, and Swift programming language.

A high-level language is further divided into three parts -

i. Procedural Oriented programming language

Procedural Oriented Programming (POP) language is derived from structured programming and based upon the procedure call concept. It divides a program into small procedures called routines or functions.

Procedural Oriented programming language is used by a software programmer to create a program that can be accomplished by using a programming editor like IDE, Adobe Dreamweaver, or Microsoft Visual Studio.

The advantage of POP language is that it helps programmers to easily track the program flow and code can be reused in different parts of the program.

Example: C, FORTRAN, Basic, Pascal, etc.

ii. Object-Oriented Programming language

Object-Oriented Programming (OOP) language is based upon the objects. In this programming language, programs are divided into small parts called objects. It is used to implement real-world entities like inheritance, polymorphism, abstraction, etc in the program to make the program reusable, efficient, and easy-to-use.
The main advantage of object-oriented programming is that OOP is faster and easier to execute, maintain, modify, as well as debug.
Example: C++, Java, Python, C#, etc.

iii. Natural language

Natural language is a part of human languages such as English, Russian, German, and Japanese. It is used by machines to understand, manipulate, and interpret human's language. It is used by developers to perform tasks such as translation, automatic summarization, Named Entity Recognition (NER), relationship extraction, and topic segmentation.
The main advantage of natural language is that it helps users to ask questions in any subject and directly respond within seconds.

### 3. Middle-level programming language

Middle-level programming language lies between the low-level programming language and high-level programming language. It is also known as the intermediate programming language and pseudo-language.
A middle-level programming language's advantages are that it supports the features of high-level programming, it is a user-friendly language, and closely related to machine language and human language.
Example: C, C++, language

## Most commonly used Programming Language

As we all know, the programming language makes our life simpler. Currently, all sectors (like education, hospitals, banks, automobiles, and more ) completely depend upon the programming language.
There are dozens of programming languages used by the industries. Some most widely used programming languages are given below -

### 1. Python

Python is one of the most widely used user-friendly programming languages. It is an open-source and easy to learn programming language developed in the 1990s. It is mostly used in Machine Learning, Artificial Intelligence, Big Data, GUI based desktop applications, and Robotics.
Advantages

- Python is easy to read, easy to understand, and easy to write.
- It integrates with other programming languages like C, C++, and Java.
- Python executes code line-by-line, so it is easy for the programmer to find the error that occurred in the code.
- Python is platform-independent means you can write code once and run it anywhere.

Disadvantages

- Python is not suitable for developing mobile applications and games.
- Python works with the interpreter. That's why it is slower than other programming languages like C and C++.

### 2. Java

Java is a simple, secure, platform-independent, reliable, architecture-neutral high-level programming language developed by Sun Microsystems in 1995. Now, Java is owned by Oracle. It is mainly used

to develop banking, retail, information technology, android, big data, research community, web, and desktop applications.
Advantages

- Java is easy to write, compile, learn, and debug as compared to other programming languages.
- It provides an ability to run the same program on different platforms.
- It is a highly secured programming language because in java, there is no concept of explicit pointers.
- It is capable of performing multiple tasks at the same time.

Disadvantages

- Java consumes more memory and is slower than other programming languages like C or C++.
- It does not provide a backup facility.

## 3. C

C is a popular, simple, and flexible general-purpose computer programming language. Dennis M Ritchie develops it in 1972 at AT&T. It is a combination of both low-level programming language as well as a high-level programming language. It is used to design applications like Text Editors, Compilers, Network devices, and many more.
Advantages

- C language is easy to learn.
- It is fast, efficient, portable, easy to extend, powerful, and flexible programming language.
- It is used to perform complex calculations and operations such as MATLAB.
- It provides dynamic memory allocation to allocate memory at the run time.

Disadvantages

- In the C programming language, it is very difficult to find the errors.
- C does not support the concepts of constructors, destructors, abstraction, polymorphism, encapsulation, and namespace like OOPs.

## 4. C++

C++ is one of the thousands of programming languages that we use to develop software. C++ programming language is developed by Bjarne Stroustrup in 1980. It is similar to the C programming language but also includes some additional features such as exception handling, object-oriented programming, type checking, etc.
Advantages

- C++ is a simple and portable structured programming language.
- It supports OOPs features such as Abstraction, Inheritance, Encapsulation.
- It provides high-level abstraction and useful for a low-level programming language, and more efficient for general-purpose.
- C++ is more compatible with the C language.

Disadvantages

- C++ programming language is not secured as compared to other programming languages like Java or Python.
- C++ can not support garbage collection.
- It is difficult to debug large as well as complex web applications.

## 5. C#

C# (pronounced as C sharp) is a modern, general-purpose, and object-oriented programming language used with XML based Web services on the .NET platform. It is mainly designed to improve productivity in web applications. It is easier to learn for those users who have sufficient knowledge of common programming languages like C, C++, or Java.
Advantages

- C# is a modern, type-safe, easy, fast, and open-source programming language that is easily integrated with Windows.
- The maintenance of C# (C sharp) is lower than the C++ programming language.
- C# is a pure object-oriented programming language.
- C# includes a strong memory backup facility. That's why it avoids the problem of memory leakage.

Disadvantages

- C# is less flexible because it is completely based on Microsoft .Net framework.
- In C#, it is difficult to write, understand, debug, and maintain multithreaded applications.

## 6. JavaScript

Javascript is a type of scripting language that is used on both client-side as well as a server-side. It is developed in the 1990s for the Netscape Navigator web browser. It allows programmers to implement complex features to make web pages alive. It helps programmers to create dynamic websites, servers, mobile applications, animated graphics, games, and more.
Advantage

- JavaScript helps us to add behaviour and interactivity on the web page.
- It can be used to decrease the loading time from the server.
- It has the ability to create attractive, dynamic websites, and rich interfaces.
- JavaScript is a simple, versatile, and lightweight programming language.
- JavaScript and its syntax are easy to understand.

Disadvantage

- JavaScript is completely based on the browser.
- It does not support multiple inheritance.
- It is less secure compared to other programming languages.

## 7. R

Currently, R programming is one of the popular programming languages that is used in data analytics, scientific research, machine learning algorithms, and statistical computing. It is developed in 1993 by Ross Ihaka and Robert Gentleman. It helps marketers and data scientists to easily analyze, present, and visualize data.
Advantages

- R programming provides extensive support for Data Wrangling.
- It provides an easy-to-use interface.
- It runs on any platform like Windows, Linux, and Mac.
- It is an open-source and platform-independent programming language.

Disadvantages

- R programming does not support 3D graphics.
- It is slower than other programming languages.

## 8. PHP

PHP stands for Hypertext Preprocessor. It is an open-source, powerful server-side scripting language mainly used to create static as well as dynamic websites. It is developed by Rasmus Laird in 1994. Inside the php, we can also write HTML, CSS, and JavaScript code. To save a php file, file extension .php is used.

Advantages

- PHP is a more secure and easy-to-use programming language.
- It supports powerful online libraries.
- It can be run on a variety of operating systems such as Windows, Linux, and Mac.
- It provides excellent compatibility with cloud services.

Disadvantages

- PHP is not capable of handling a large number of applications and not suitable for large applications.
- It is quite difficult to maintain.

## 9. Go

Go or Golang is an open-source programming language. It is used to build simple, reliable, and efficient software. It is developed by Robert Griesemer, Rob Pike, and Ken Thompson in 2007.

Advantages

- Go language is easy-to-learn and use.
- It comes with the in-built testing tools.
- Go is a fast programming language.

Disadvantages

- Go language does not support generics.
- It does not support error handling.
- It supports a lack of frameworks.

## 10. Ruby

Ruby is an open-source, general-purpose, and pure object-oriented programming language released in 1993. It is used in front-end and back-end web development. It is mainly designed to write CGI (Common Gateway Interface) scripts.

Advantages

- Ruby supports various GUI (Graphical User Interface) tools like GTK and OpenGL.
- It is used to develop both internet as well as intranet applications.
- The code written in Ruby is small and contains less number of lines.

Disadvantages

- Ruby is slower than other programming languages.
- It is very difficult for programmers to debug the code written in Ruby.

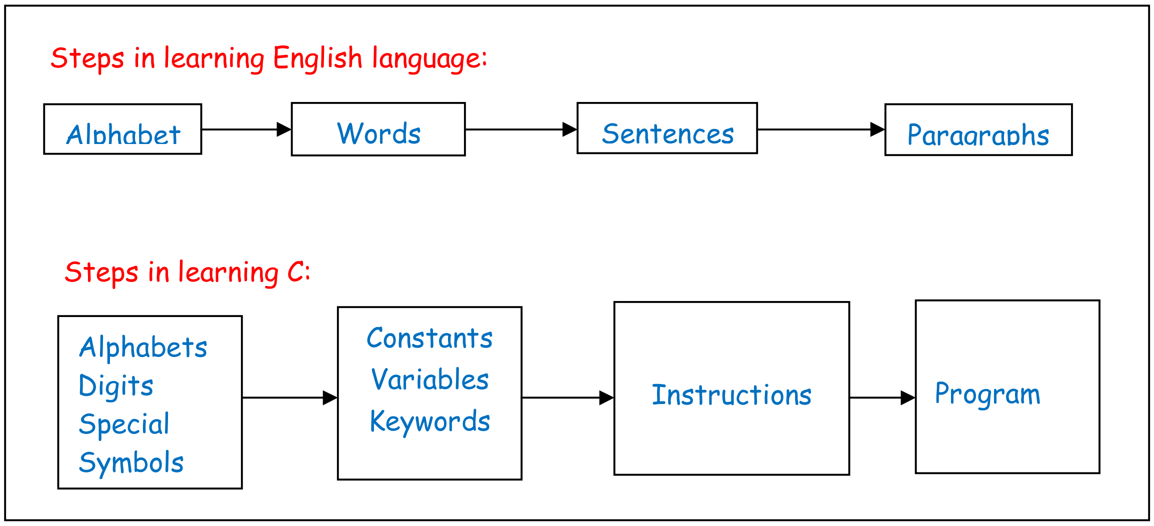
# Session 6

## What is C?

C is a programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL etc. Possibly why C seems so popular because it is reliable, simple and easy to use.

## Getting started with C

There is a close analogy between learning English language and learning C language. The classical method of learning English is to first learn the alphabets used in the language, then learn to combine these alphabets to form words, which in turn are combined to form paragraphs. Learning c is similar. We must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables and keywords are constructed, and finally how are these combined to form an instruction. A group of instructions would be combined later on to form a program. This is illustrated in Figure 1:


Steps in learning English language:
Alphabet
Words
Sentences
Paragraphs
Steps in learning C:
Alphabets Digits Special Symbols
Constants Variables Keywords
Instructions
Program

## C Character set

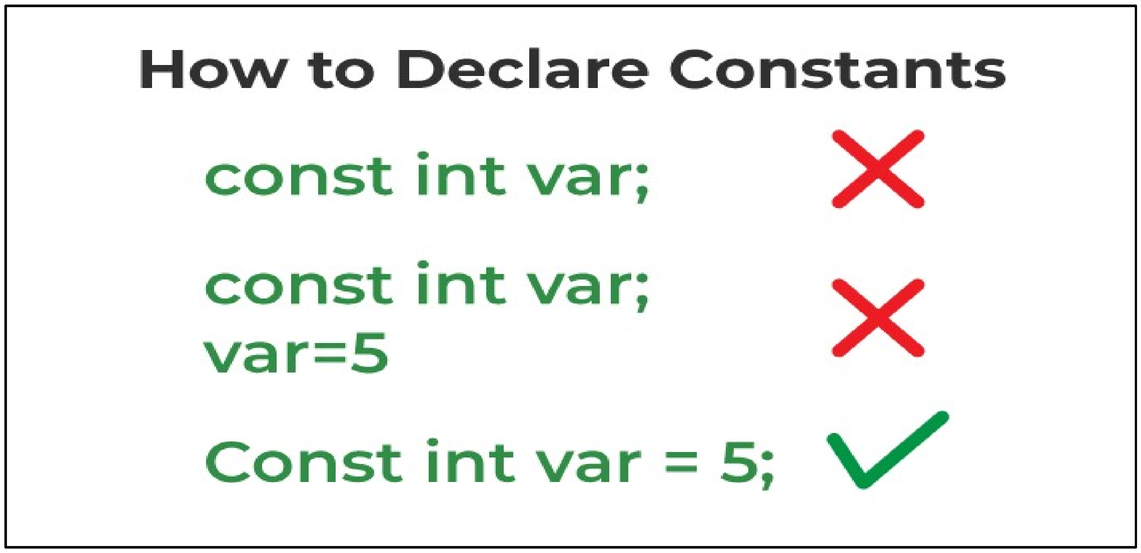
| | |
|---|---|
| **Alphabets** | A, B,......,Y, Z<br>a, b, .........y, z |

| | |
|---|---|
| **Digits** | 0,1,2,3,4,5,6,7,8,9 |
| **Special symbols** | ~ ’ ! @ # % ^ & * ( ) _ - + = \| \ { } [] : ; “ <> , . ? / |

Figure 2

## What is Constants

A constant is a name given to the variable whose values can’t be altered or changed. It means that once we assign value to the constant, then we can’t change it throughout the execution of a program- it stays fixed.



How to Declare Constants

```
const int var;          ✗

const int var;          ✗
var=5

Const int var = 5;      ✓
```

Figure 3

## Types of C constants

C constants can be divided into two major categories:

a) Primary Constants
b) Secondary Constants

These constants are further categorised as shown in Figure 4.1 and Figure 4.2

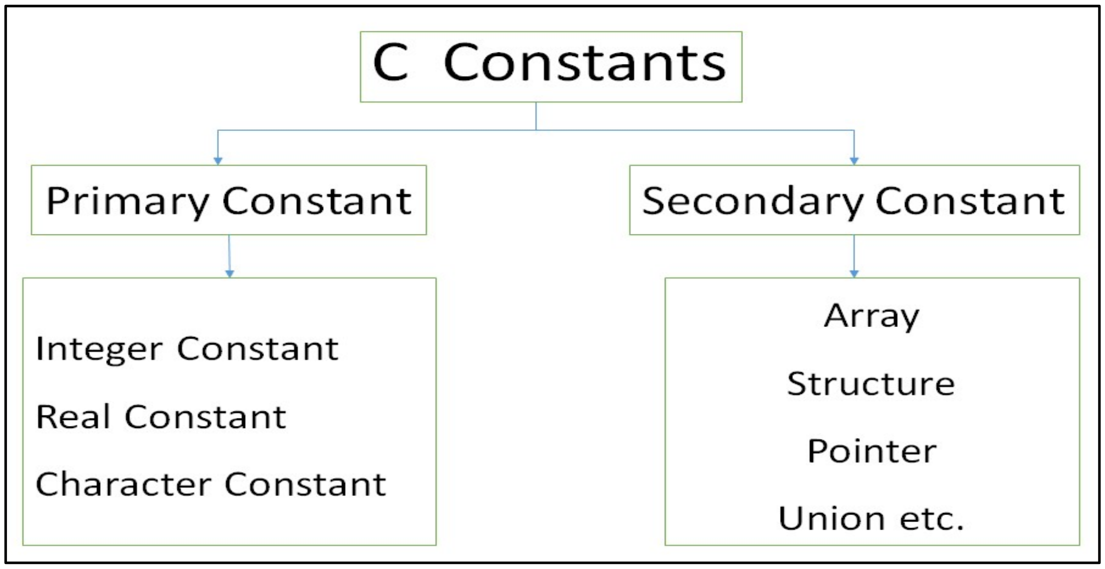


Figure 4.1

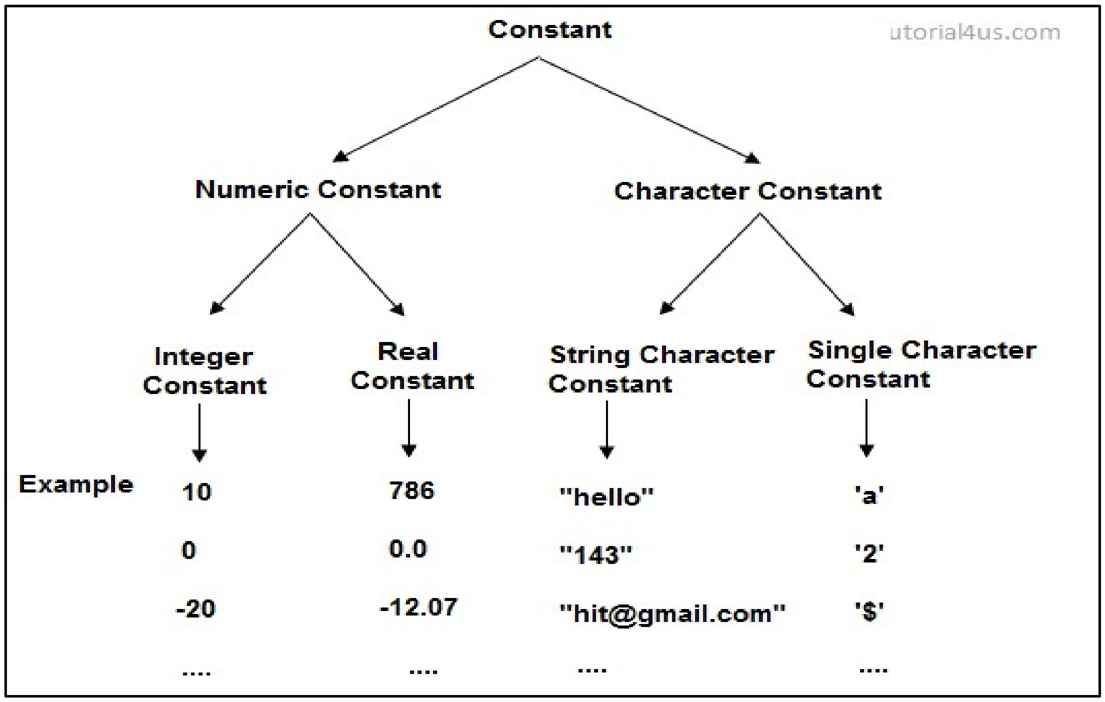


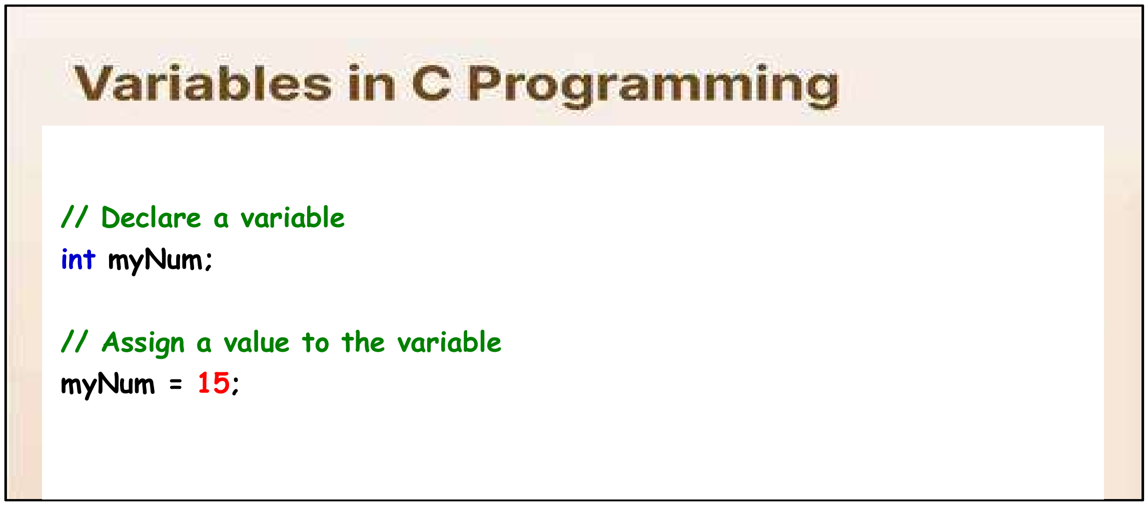
Figure 4.2

## What is Variable

Variables are containers for storing data values, like numbers and characters.

In C, there are different types of variables (defined with different keywords), for example:

- int - stores integers (whole numbers), without decimals, such as 123 or -123
- float - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Characters are surrounded by single quotes


Variables in C Programming

```
// Declare a variable
int myNum;

// Assign a value to the variable
myNum = 15;
```

Figure 5

## Rules for constructing variable names

a) A variable name is ant combinations of 1 to 31 alphabets, digits or underscores. Do not create unnecessarily long variable names as it adds to your typing effort.

b) Names can contain letters, digits and underscores.

c) Names must begin with a letter or an underscore (_)

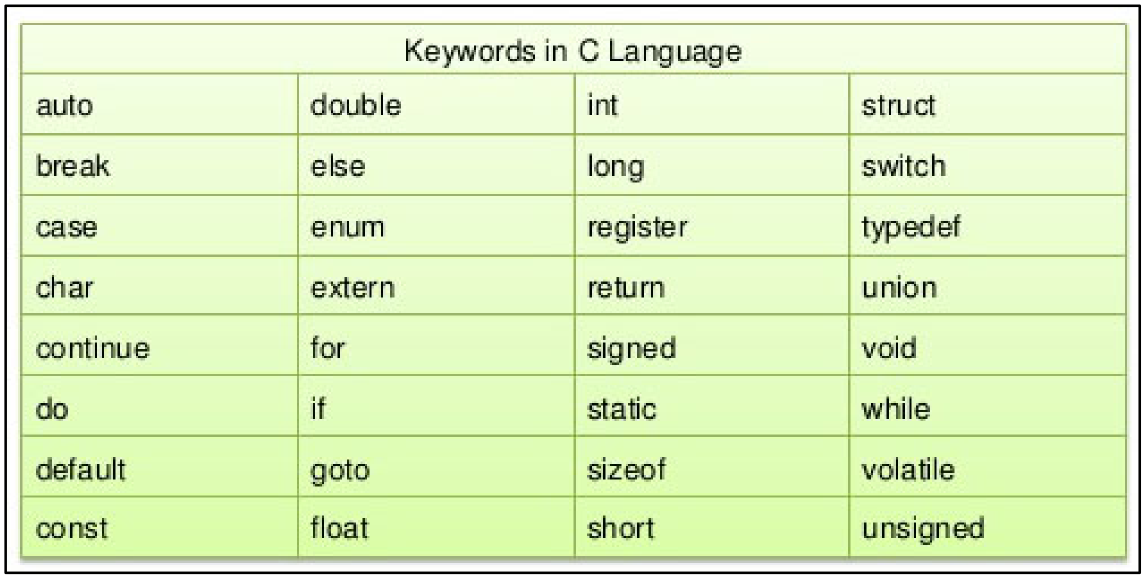
d) Names are case-sensitive ( myVar and myvar are different variables)

e) Names cannot contain whitespaces or special characters like !, #, %, etc.

f) Reserved words (such as int ) cannot be used as names.

## C keywords

Keywords are predefined or reserved words that have special meanings to the compiler. These are part of the syntax and cannot be used as identifiers in the program. There are only 32 keywords available in C. Figure 6 gives the list of these keywords.

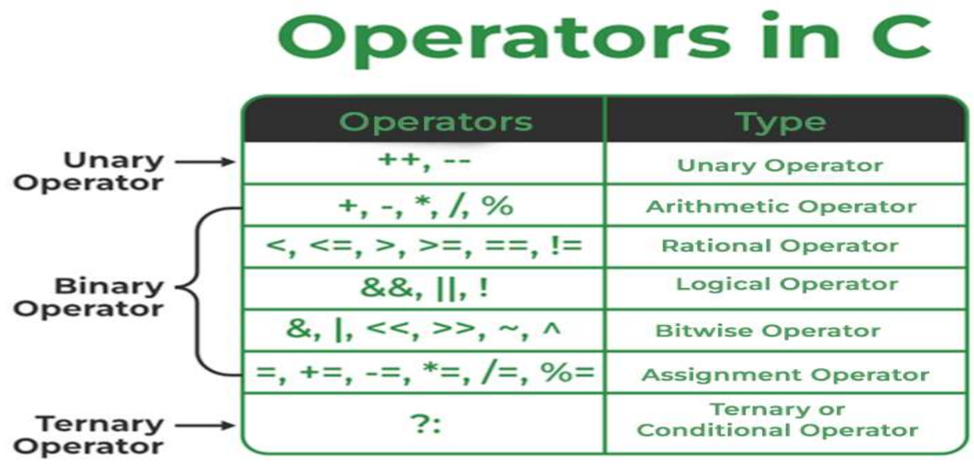


| Keywords in C Language | | | |
|---|---|---|---|
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | return | union |
| continue | for | signed | void |
| do | if | static | while |
| default | goto | sizeof | volatile |
| const | float | short | unsigned |

Figure 6

## C Operators

In C language, operators are symbols that represent operations to be performed on one or more operands. There are various types of operators as mentioned in Figure 7.


Operators in C

| | Operators | Type |
|---|---|---|
| Unary Operator | ++, -- | Unary Operator |
| Binary Operator | +, -, *, /, % | Arithmetic Operator |
| | <, <=, >, >=, ==, != | Rational Operator |
| | &&, \|\|, ! | Logical Operator |
| | &, \|, <<, >>, ~, ^ | Bitwise Operator |
| | =, +=, -=, *=, /=, %= | Assignment Operator |
| Ternary Operator | ?: | Ternary or Conditional Operator |

Figure 7

For example,

c = a + b

Here, '+' is the operator known as the addition operator, and 'a' and 'b' are operands. The addition operator tells the compiler to add both of the operands 'a' and 'b'.

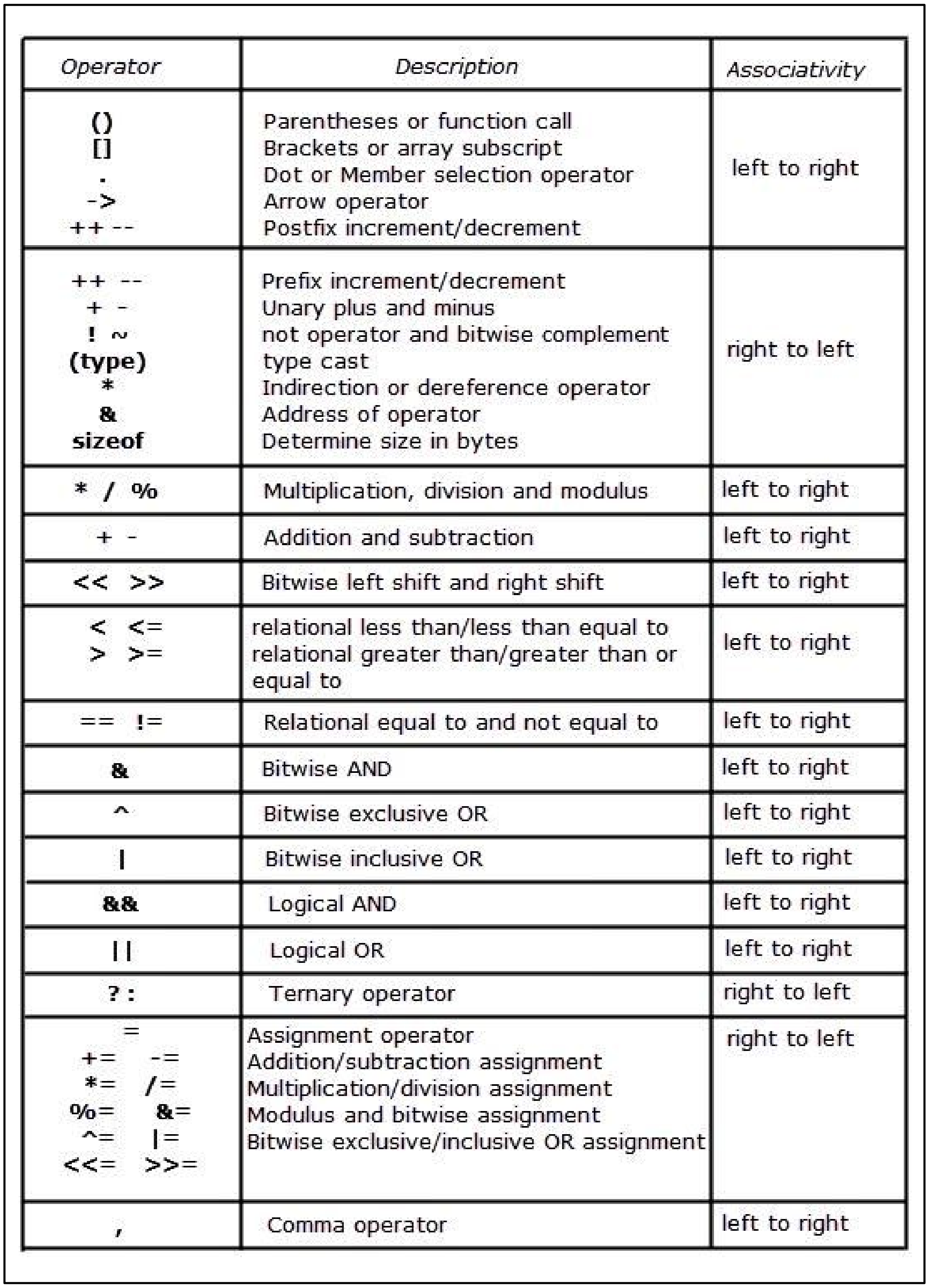
Apart from the above operators, there are some other operators available in C used to perform some specific tasks, like sizeof, comma (,), dot (.) and arraow (->), cast operator (type) etc.

## Operator Precedence and Associativity in C

In C, it is very common for an expression or statement to have multiple operators and in these expressions, there should be a fixed order or priority of operator evaluation to avoid ambiguity.

Operator Precedence and Associativity is the concept that decides which operator will be evaluated first in the case when there are multiple operators present in an expression.

The below table describes the precedence order and associativity of operators in C. The precedence of the operator decreases from top to bottom.



| Operator | Description | Associativity |
|---|---|---|
| () | Parentheses or function call | left to right |
| [] | Brackets or array subscript | |
| . | Dot or Member selection operator | |
| -> | Arrow operator | |
| ++ -- | Postfix increment/decrement | |
| ++ -- | Prefix increment/decrement | right to left |
| + - | Unary plus and minus | |
| ! ~ | not operator and bitwise complement | |
| (type) | type cast | |
| * | Indirection or dereference operator | |
| & | Address of operator | |
| sizeof | Determine size in bytes | |
| * / % | Multiplication, division and modulus | left to right |
| + - | Addition and subtraction | left to right |
| << >> | Bitwise left shift and right shift | left to right |
| < <= | relational less than/less than equal to | left to right |
| > >= | relational greater than/greater than or equal to | |
| == != | Relational equal to and not equal to | left to right |
| & | Bitwise AND | left to right |
| ^ | Bitwise exclusive OR | left to right |
| \| | Bitwise inclusive OR | left to right |
| && | Logical AND | left to right |
| \|\| | Logical OR | left to right |
| ? : | Ternary operator | right to left |
| = | Assignment operator | right to left |
| += -= | Addition/subtraction assignment | |
| *= /= | Multiplication/division assignment | |
| %= &= | Modulus and bitwise assignment | |
| ^= \|= | Bitwise exclusive/inclusive OR assignment | |
| <<= >>= | | |
| , | Comma operator | left to right |

Figure 8

## C- Loops

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code.

There are mainly two types of loops in C Programming:

1. Entry Controlled loops: In Entry controlled loops the test condition is checked before entering the main body of the loop. For Loop and While Loop is Entry-controlled loops.

2. Exit Controlled loops: In Exit controlled loops the test condition is evaluated at the end of the loop body. The loop body will execute at least once, irrespective of whether the condition is true or false. do-while Loop is Exit Controlled loop.

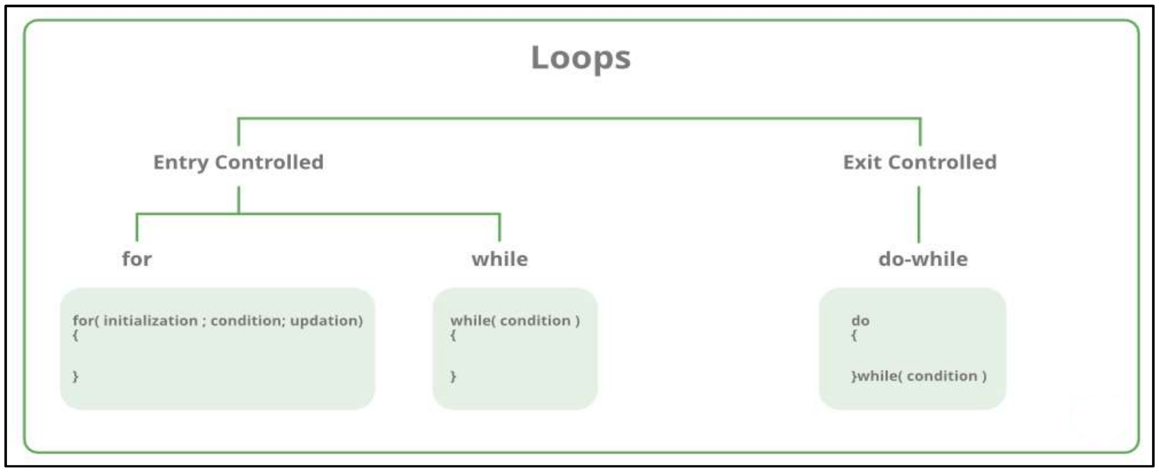


Figure 9

## C- Functions

A **function in C** is a set of statements that when called perform some specific task. It is the basic building block of a C program that provides modularity and code reusability. The programming statements of a function are enclosed within **{ } braces**, having certain meanings and performing certain operations. They are also called subroutines or procedures in other languages.

The syntax of function can be divided into 3 aspects:

1. **Function Declaration**
2. **Function Definition**

3. **Function Calls**

**Function Declarations**

In a function declaration, we must provide the function name, its return type, and the number and type of its parameters. A function declaration tells the compiler that there is a function with the given name defined somewhere else in the program.

Syntax:

```
return_type name_of_the_function (parameter_1, parameter_2);
```

The parameter name is not mandatory while declaring functions. We can also declare the function without using the name of the data variables.

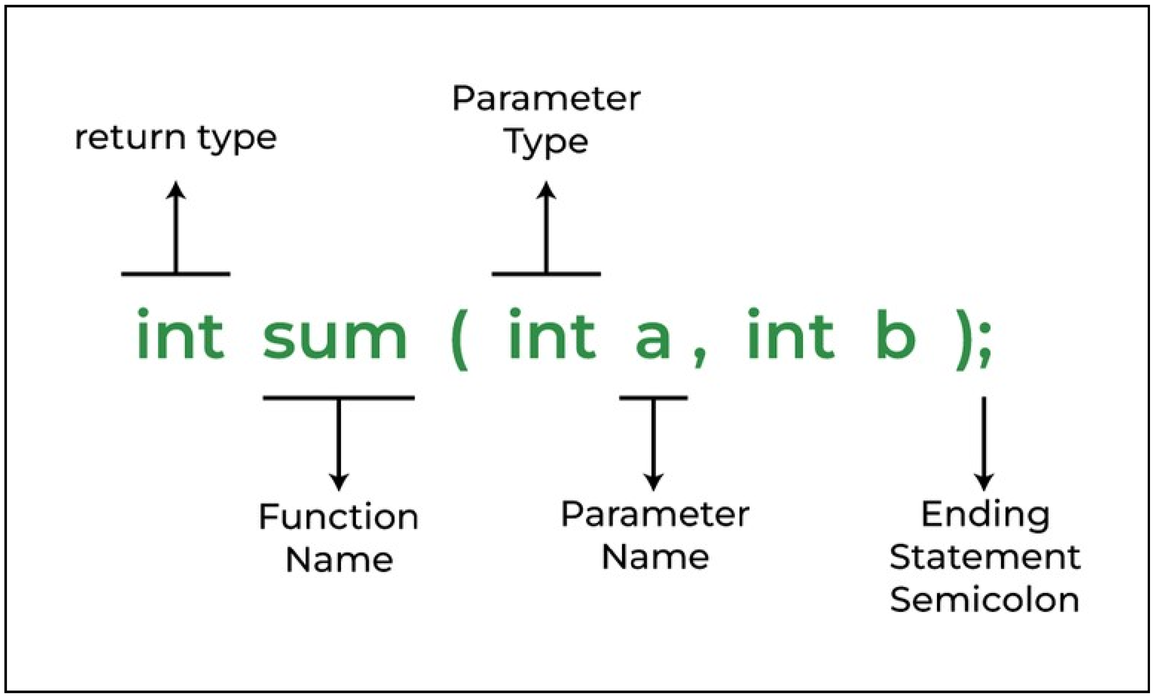


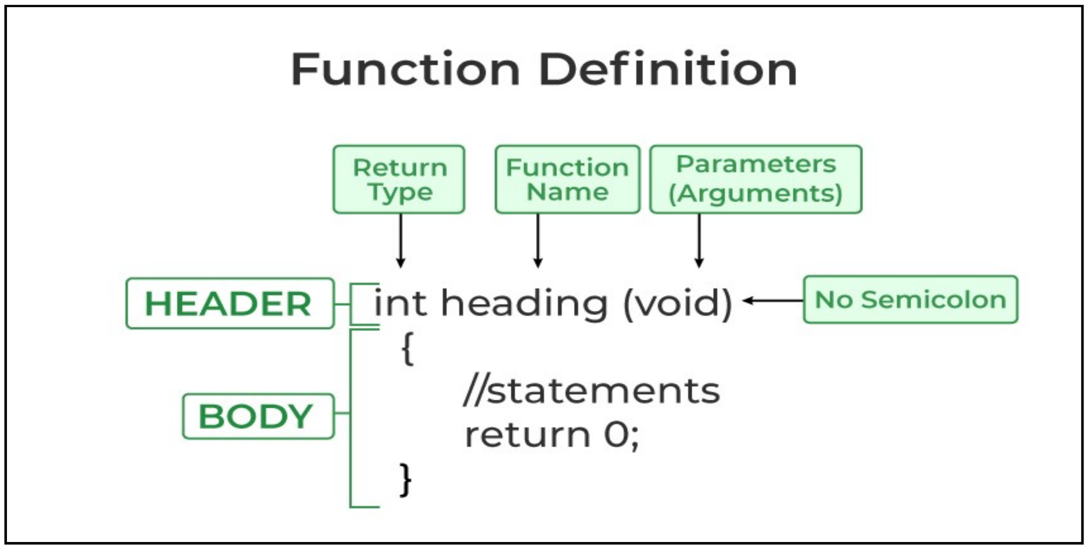
Figure 10

**Function Definition**

The function definition consists of actual statements which are executed when the function is called (i.e. when the program control comes to the function).

Syntax:

```
return_type function_name (para1_type para1_name, para2_type para2_name)
{
// body of the function
}
```

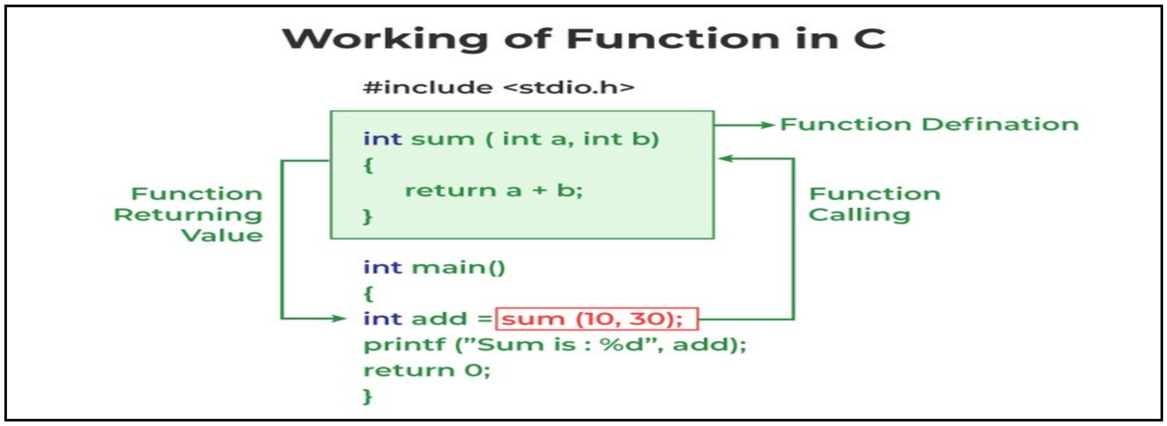


```
int heading (void)
{
    //statements
    return 0;
}
```

Figure 11

**Function Call**

A function call is a statement that instructs the compiler to execute the function. We use the function name and parameters in the function call.



```
#include <stdio.h>
int sum ( int a, int b)
{
    return a + b;
}
int main()
{
int add = sum (10, 30);
printf ("Sum is : %d", add);
return 0;
}
```

Figure 12

**Types of Functions**

There are two types of functions in C:

1. **Library Functions**
2. **User Defined Functions**

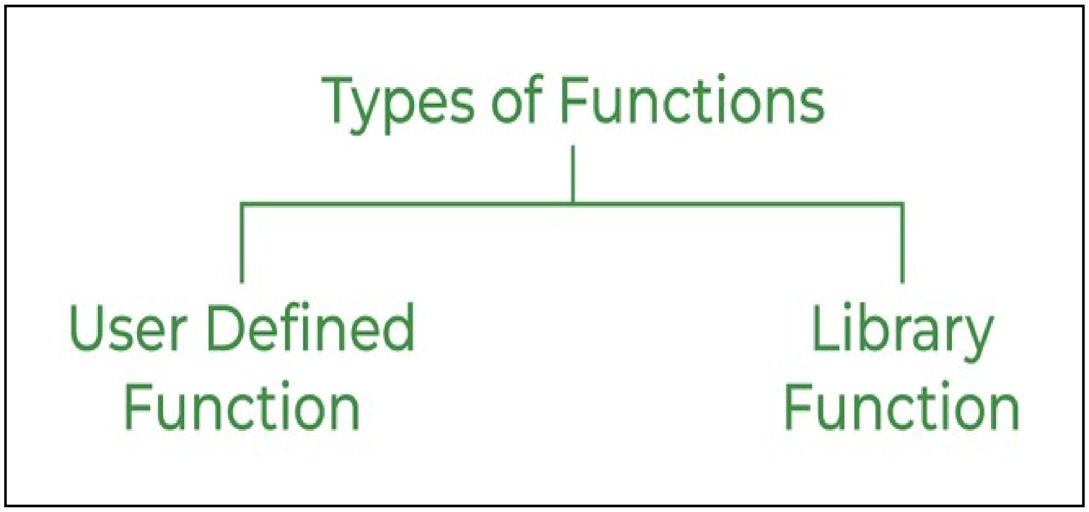


Figure 13

**1. Library Function**

A library function is also referred to as a **"built-in function"**. A compiler package already exists that contains these functions, each of which has a specific meaning and is included in the package. Built-in functions have the advantage of being directly usable without being defined, whereas user-defined functions must be declared and defined before being used.
**For Example: printf(), scanf(),**pow(), sqrt(), strcmp(), strcpy() etc.

**2. User Defined Function**

Functions that the programmer creates are known as User-Defined functions or **"tailor-made functions"**. User-defined functions can be improved and modified according to the need of the programmer. Whenever we write a function that is case-specific and is not defined in any header file, we need to declare and define our own functions according to the syntax.
**For Example: main(), sum()** etc.

## Exercises

1. What is an instruction?
2. Write down differences between constant and variable.
3. Find out error:

```
Int main()
{
        int 1n=8;
        printf("%d",1n);
        return (0);
}
```

4. Why do we need library function?
5. Write two examples of entry controlled loop.
6. What is parameter in function?
7. What do you mean by ternary operator? Give example.
8. What will be the value of the expression, a=7/22*(3.14+2)*3/5;
9. Can ‘void’ be used as a variable name? Why?
10. Which keyword is used to declare an identifier as constant?
11. Define operator and operand with suitable example.
12. Give the syntax of function declaration.
13. Write short note on main().
14. What is the difference between ‘a’ and “a”?
15. What will happen if there is no termination condition in loop?

# Session 7

## Programming with C language:

### C Quick start

A **computer program** is a list of "instructions" to be "executed" by a computer.In a programming language, these programming instructions are called **statements**.

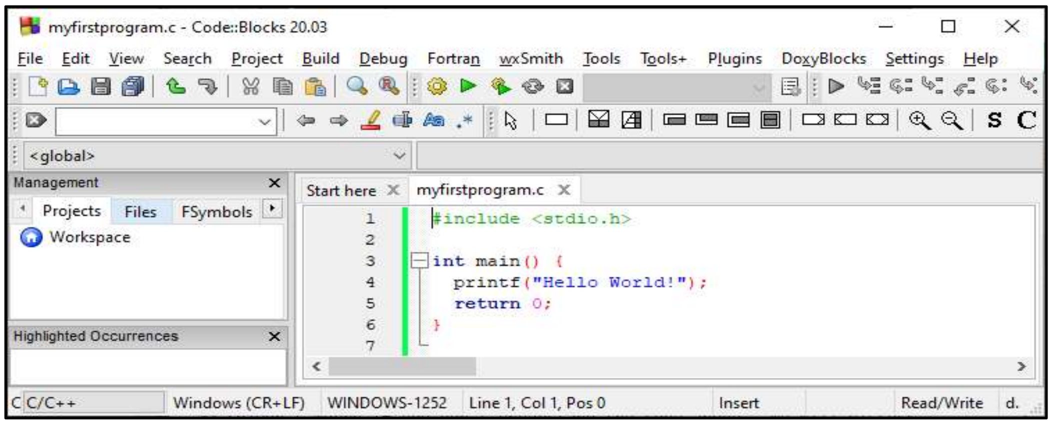
To create C file, Open Codeblocks and go to File > New > Empty File.

Write the following C code and save the file (File > Save File)

```
#include <stdio.h>
int main() {
  printf("Hello World!");
  return 0;
}
```

Before explaining the above code we will first focus on how to run the code.

In Codeblocks, it should look like this:


myfirstprogram.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSymbols
Workspace
Highlighted Occurrences
Start here
myfirstprogram.c
C C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write d.

```
#include <stdio.h>

int main() {
   printf("Hello World!");
   return 0;
}

```

After writing the code, then, go to Build > Build and Run to run (execute) the program. The result will look something to this:

```
Hello World!
Process returned 0 (0x0) execution time : 0.011 s
Press any key to continue.
```

Let's break the above code down to understand it better:

**Line 1:** `#include <stdio.h>` is a **header file library** that lets us work with input and output functions, such as `printf()` (used in line 4). Header files add functionality to C programs.

**A header file** is a file with extension . h which contains C function declarations and macro definitions to be shared between several source files.

**Line 2:** Another thing that always appear in a C program is `main()`. This is called a **function**. Any code inside its curly brackets `{}` will be executed.

**Line 3:** `printf()` is a **library function** used to output/print text to the screen. In our example, it will output "Hello World!".

In C programming language, a ***library function*** is a prewritten piece of code that performs a specific task. These functions are included in ***precompiled libraries***, which can be linked to a program to provide additional functionality.

**Note that:** Every C statement ends with a semicolon **`;`**

**Note:** The body of `int main()` could also been written as:
`int main(){printf("Hello World!");return 0;}`

**Remember:** The compiler ignores white spaces. However, multiple lines makes the code more readable.

**Line 4:** `return 0` ends the `main()` function.

**Line 5:** Do not forget to add the closing curly bracket `}` to actually end the main function.

## C Datatype

Example

```
// Create variables
int myNum = 5;             // Integer (whole number)
float myFloatNum = 5.99;   // Floating point number
char myLetter = 'D';       // Character

// Print variables
printf("%d\n", myNum);
```

**The data type** specifies the size and type of information the variable will store.

Some basic data types are as follows:

| Data Type | Size | Description | Example |
|---|---|---|---|
| int | 2 or 4 bytes | Stores whole numbers, without decimals | 1 |
| float | 4 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 6-7 decimal digits | 1.99 |
| double | 8 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits | 1.99 |
| char | 1 byte | Stores a single character/letter/number, or ASCII values | 'A' |

**Basic Format Specifiers**

There are different format specifiers for each data type. Here are some of them:

| Format Specifier | Data Type |
|---|---|
| `%d` or `%i` | `int` |

| | |
|---|---|
| %f | float |
| %lf | double |
| %c | char |
| %s | Used for strings **(text)** |

## User Input

You have already learned that `printf()` is used to output values in C.To get user input, you can use the `scanf()` function.

Example

```
// Create an integer variable that will store the number we get from the user
int myNum;

// Ask the user to type a number
printf("Type a number: \n");

// Get and save the number the user types
scanf("%d", &myNum);

// Output the number the user typed
printf("Your number is: %d", myNum);
```

T V v

T n
the following example):

Example

```
// Create an int and a char variable
int myNum;
char myChar;

// Ask the user to type a number AND a character
```

## Decision-Making Statements in C

In C, decision-making statements are technology structures enabling programmers to make decisions based on specific conditions or criteria. In C, there are three primary decision-making statements that you can use:

- If-else statements
- Switch statements
- Conditional operator statements

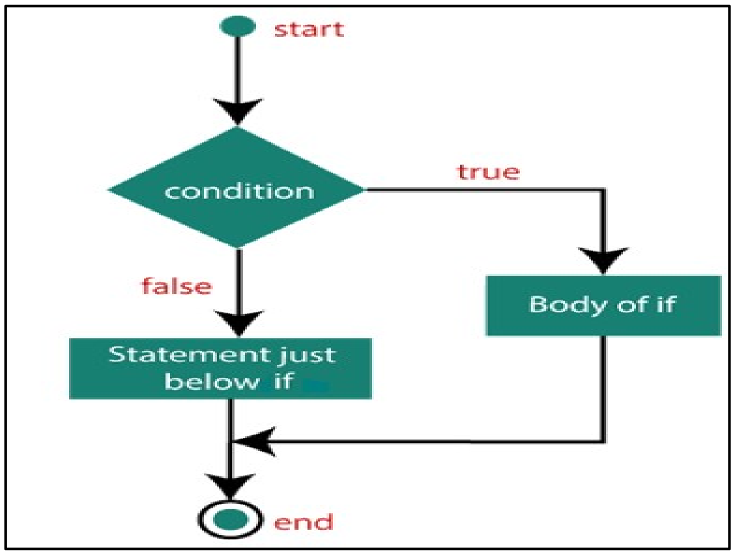
Each of these statements allows you to make decisions in different ways, depending on the complexity of your code and the specific conditions you need to evaluate.

**If-else statements:**

The if-else statement is C's most fundamental decision-making statement. It enables us to run one block of code if a given condition is met and another if it is not. An if-else statement in C has the following basic syntax:

Syntax:

```
 if (condition) {
// code to execute if the condition is true
 } else {
// code to execute if the condition is false
 }
```



For example, if you wanted to check if a number is even or odd, you could use an if-else statement like this:

Code:

```
nclude <stdio.h>

  int main() {
t num = 6;

 (num % 2 == 0) {
intf("The number is even");
  } else {
intf("The number is odd");
  }
turn 0;
```

Output

```
The number is even
```

This code will output "The number is even" because 6 is divisible by 2.

Syntax:

```
switch (expression) {
    case value1:
        // code to execute if expression == value1
        break;
```

For example, if you wanted to check the grade of a student based on their score, you could use a switch statement like this:

Code:

```
#include <stdio.h>

int main() {
int score = 85;
char grade;

switch (score / 10) {
   case 10: break;
   case 9:
      grade = 'A';
      break;
   case 8:
      grade = 'B';
      break;
   case 7:
      grade = 'C';
      break;
   case 6:
      grade = 'D';
      break;
   default:
      grade = 'F';
      break;
}

printf("The grade is %c", grade);

   return 0;
}
```

Output

```
The grade is B
```

This code will output "The grade is B" because 85 divided by 10 is 8.5, which falls between 80 and 89, so the switch statement matches the case for "8" and assigns the value 'B' to the variable "grade".

```
condition ? value_if_true : value_if_false
```

For example, if you wanted to check if a number is positive or negative, you could use a conditional operator statement like this:

**Code:**

```
#include <stdio.h>

int main() {
int num = -5;
( num >= 0 )? Printf("positive") : printf( "negative");

 return 0;
}
```

**Output**

```
The number is negative
```

This code will output "The number is negative" because -5 is less than 0, so the conditional operator statement chooses to print "negative" for the.

## C Loops

Loops in C is used to execute the block of code several times according to the condition given in the loop. It means it executes the same code multiple times so it saves code.

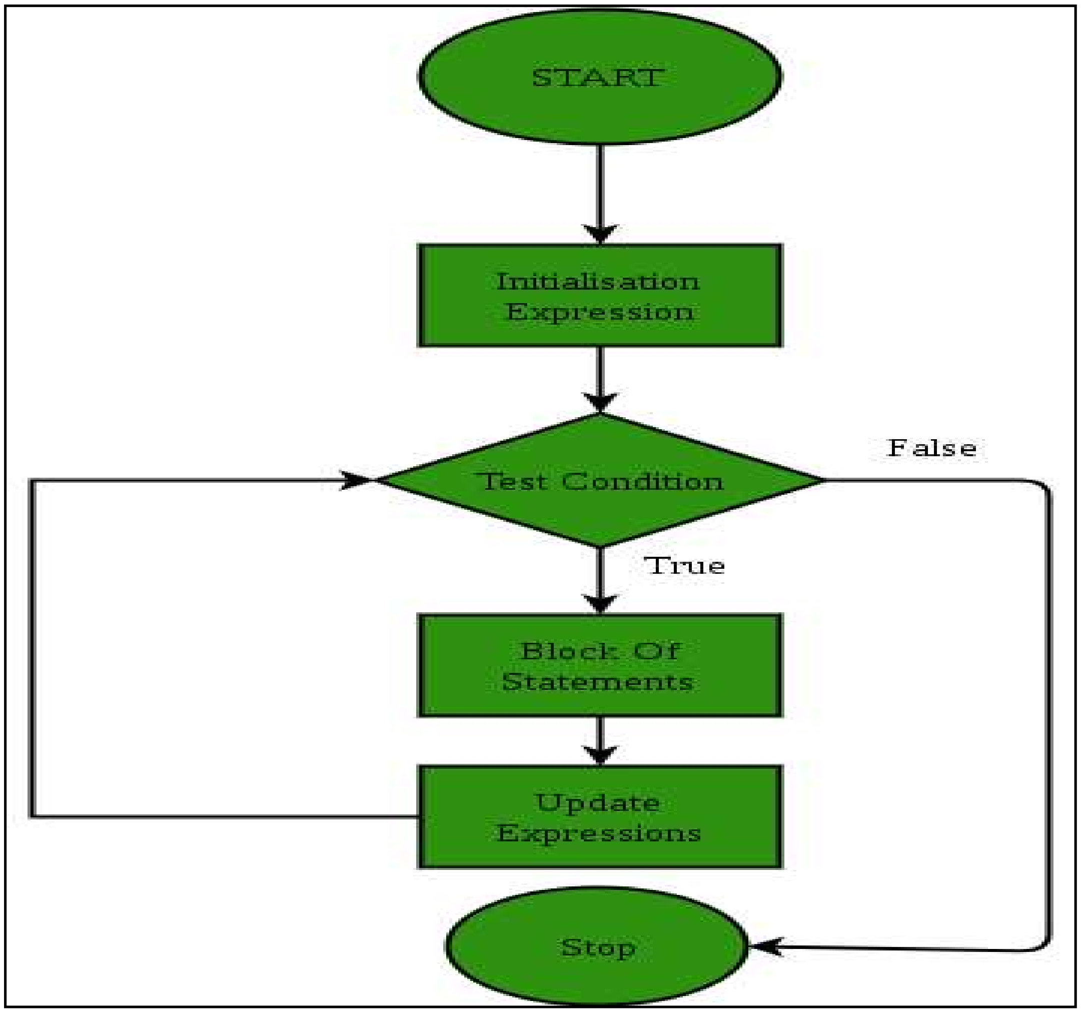
### for Loop

"for loop" in C programming is a repetition control structure that allows programmers to write a loop that will be executed a specific number of times.

**Syntax:**

```
for (initialize expression; test expression; update expression)
{
   // body of for loop
}
```

**for loop Equivalent Flow Diagram:**



Example

```
int i;

for (i = 0; i < 5; i++) {
  printf("%d\n", i);
}
```

**Example explained**

Expression 1 sets a variable before the loop starts (int i = 0).

Expression 2 defines the condition for the loop to run (i must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.

Expression 3 increases a value (i++) each time the code block in the loop has been executed.

## While Loop

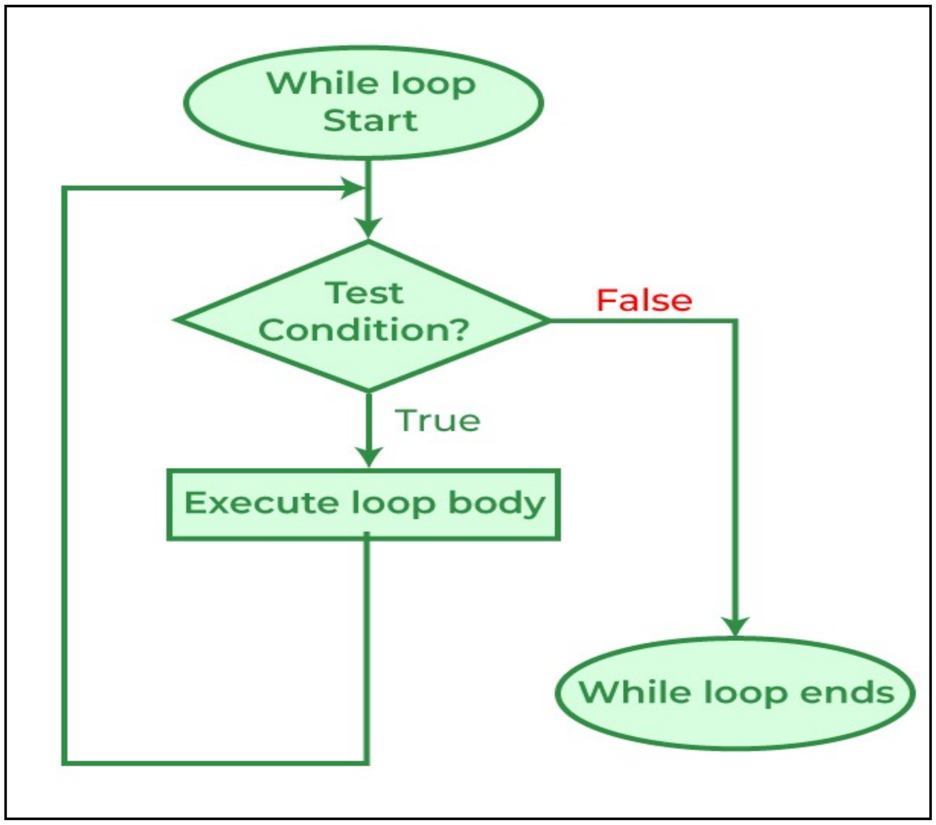
“While loop” does not depend upon the number of iterations. In for loop the number of iterations was previously known to us but in the While loop, the execution is terminated on the basis of the test condition. If the test condition will become false then it will break from the while loop else body will be executed.

**Syntax:**

```
initialization_expression;


while (test_expression)
{
   // body of the while loop
   update_expression;
}
```

**Flow Diagram for while loop:**



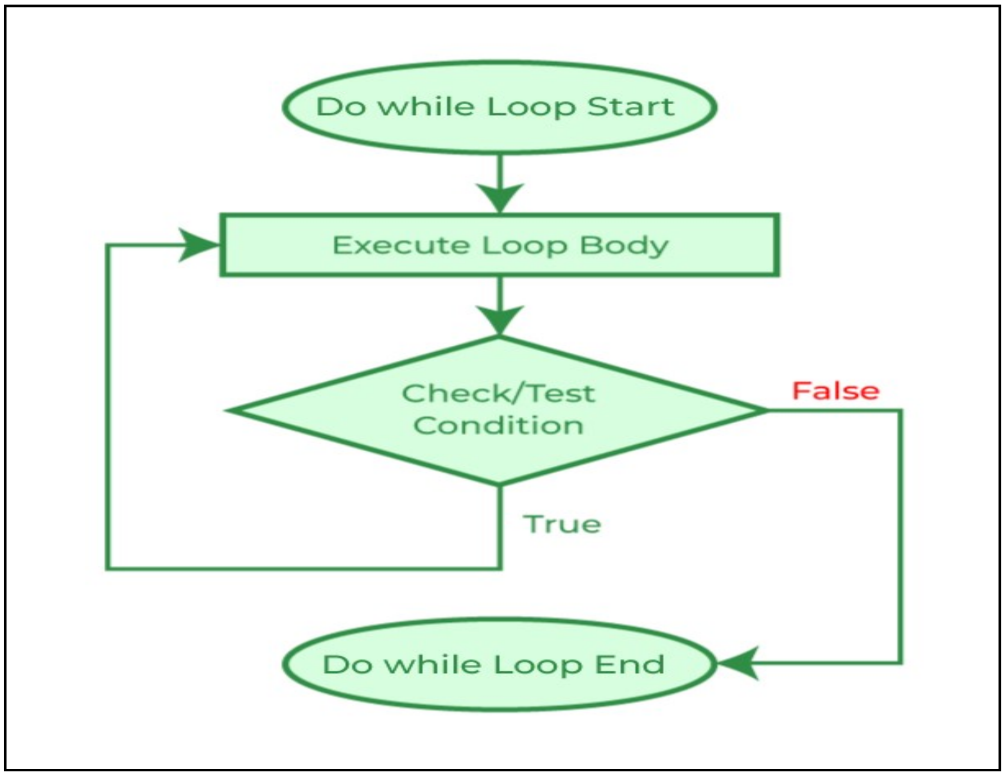
**do-while Loop**

The do-while loop is similar to a while loop but the only difference lies in the do-while loop test condition which is tested at the end of the body. In the do-while loop, the loop body will **execute at least once** irrespective of the test condition.

**Syntax:**

```
initialization_expression;
do
{
   // body of do-while loop
   update_expression;
} while (test_expression);
```

**Flow Diagram for do-while loop:**



## Loop Control Statements

Loop control statements in C programming are used to change execution from its normal sequence.

| Name | Description |
|---|---|
| break statement | the break statement is used to terminate the switch and loop statement. It transfers the execution to the statement immediately following the loop or switch. |
| continue statement | continue statement skips the remainder body and immediately resets its condition before reiterating it. |
| goto statement | goto statement transfers the control to the labelled statement. |

**Infinite Loop**

An infinite loop is executed when the test expression never becomes false and the body of the loop is executed repeatedly. A program is stuck in an Infinite loop when the condition is always true. Mostly this is an error that can be resolved by using Loop Control statements.

## Exercises

1. Why do we need format specifier?
2. Write down few differences between ‘while’ and ‘do-while’ loop.
3. What is the purpose of using loop in programs?
4. Write a C program to print even numbers using while loop.
5. Write down the limitations of “switch-case” over “if-else”.
6. Check whether an integer is even or odd using conditional operator.
7. What is the difference between ‘break” and “continue”?
8. How many bits are required to store a character variable?
9. What is data type? Give example.
10. What is the utilization of header files in a program?

# Session 8

## Some basic programs on C language:

### 1. Write a C program to swap two integers without using third variable.

```
#include<stdio.h>
int main()
{
int a=10, b=20;
printf("Before swap a=%d b=%d",a,b);
a=a+b;//a=30 (10+20)
b=a-b;//b=10 (30-20)
a=a-b;//a=20 (30-10)
printf("\nAfter swap a=%d b=%d",a,b);
return 0;
}
```

### 2. Ramesh's basic salary is input through the keyboard. His DA is 40% of basic salary and HRA is 20% of basic salary. Write a program to calculate his gross salary.

```
#include<stdio.h>
int main()
{
   float basic_salary, dallowance, house_rent, gross_salary;
   printf("Enter Basic Salary: ");
   scanf("%f",&basic_salary);

   dallowance = 0.4 * basic_salary;
   house_rent = 0.2 * basic_salary;
   gross_salary = basic_salary + dallowance + house_rent;

   printf("\n Basic Salary: %.2f",  basic_salary);
   printf("\n Dearness Allowance: %.2f", dallowance);
   printf("\n House Rent: %.2f", house_rent);
   printf("\n\n Gross Salary: %.2f", gross_salary);

   getch();
   return (0);
}
```

**3. Write a c program to take an integer ass input through keyboard and check whether it is even or odd.**

```
#include <stdio.h>

int main() {

   int num;

   printf("Enter an integer: ");

   scanf("%d", &num);

   // true if num is perfectly divisible by 2

   if(num % 2 == 0)

      printf("%d is even.", num);

   else

    printf("%d is odd.", num);

   return 0;

}
```

**4. Enter a year through the keyboard. Write a C program to check whether it is leap year or not.**

```
#include <stdio.h>

int main() {
   int year;
   printf("Enter a year: ");
   scanf("%d", &year);
   // leap year if perfectly divisible by 400
   if (year % 400 == 0) {
      printf("%d is a leap year.", year);
   }
   // not a leap year if divisible by 100
   // but not divisible by 400
   else if (year % 100 == 0) {
      printf("%d is not a leap year.", year);
   }
   // leap year if not divisible by 100
   // but divisible by 4
   else if (year % 4 == 0) {
      printf("%d is a leap year.", year);
   }
   // all other years are not leap years
   else {
      printf("%d is not a leap year.", year);
   }
  return 0;
}
```

### 5. Write a C program to calculate the factorial value of an integer number.

```
#include <stdio.h>
int main()
 {
   int n, i;
   unsigned long long fact = 1;
   printf("Enter an integer: ");
   scanf("%d", &n);
   // shows error if the user enters a negative integer
   if (n < 0)
      printf("Error! Factorial of a negative number doesn't exist.");
   else
 {
      for (i = 1; i <= n; ++i)
{
         fact *= i;
      }
  printf("Factorial of %d = %llu", n, fact);
   }
   return 0;
}
```

**6. Write a C program to check whether an integer is prime or not.**

```
#include <stdio.h>
int main()
{
 int n, i, flag = 0;
 printf("Enter a positive integer: ");
 scanf("%d", &n);
 // 0 and 1 are not prime numbers
 // change flag to 1 for non-prime number
 if (n == 0 || n == 1)
  flag = 1;
 for (i = 2; i <= n / 2; ++i)
{
  // if n is divisible by i, then n is not prime
  // change flag to 1 for non-prime number
  if (n % i == 0) {
   flag = 1;
   break;
  }
 }
 // flag is 0 for prime numbers
 if (flag == 0)
  printf("%d is a prime number.", n);
 else
  printf("%d is not a prime number.", n);
 return 0;
}
```

## 7. Write a C program to determine whether a number is Armstrong or not.

```
#include <stdio.h>
int main() {
    int num, originalNum, remainder, result = 0;
    printf("Enter an integer: ");
    scanf("%d", &num);
    originalNum = num;
    while (originalNum != 0) {
       // remainder contains the last digit
        remainder = originalNum % 10;
       result += remainder * remainder * remainder;
       // removing last digit from the orignal number
       originalNum /= 10;
    }
    if (result == num)
        printf("%d is an Armstrong number.", num);
    else
        printf("%d is not an Armstrong number.", num);
    return 0;
}
```

## 8. Take an integer as input through keyboard and calculate the sum of its digits.

```
#include<stdio.h>
 int main()
{
int n,sum=0,m;
printf("Enter a number:");
scanf("%d",&n);
while(n>0)
{
m=n%10;
sum=sum+m;
n=n/10;
}
printf("Sum is=%d",sum);
return 0;
}
```

## 9. Write a C program to determine GCD of two integers using do-while loop.

```
#include <stdio.h>

int main() {
   int n1,n2,rem;
   printf("\nEnter the bigger number: ");
   scanf("%d",&n1);

   printf("\nEnter the smaller number: ");
   scanf("%d",&n2);
   if(n2>0)
   {
      do
      {
      rem=n1%n2;
      if(rem==0)
         break;
      else
      {
         n1=n2;
         n2=rem;
      }
      }while(rem!=0);
      printf("\nGCD= %d",n2);
   }
   else
   printf("\nEnter valid number!!!");
   return 0;
}
```

## 10. Write a C program to calculate the power value of an integer using function call.

```
#include <stdio.h>
#include <math.h>
int main ()
{
   // declare local variable
   long int base, exp;
   printf (" Enter the base value: ");
   scanf (" %d", &base); // take a number from user

   printf (" Enter the power value: ");
   scanf (" %d", &exp); // take a number from user

   // use pow() function to pass the base and exp variable
   printf (" %d to the power %d is = %d ", base, exp, pow_num (base, exp));
}

// definition of the function
int pow_num (int x, int y)
{
   int power = 1, i; // declare variables
   for (i = 1; i <= y; ++i)
   {
      power = power * x;

   }
   return (power);
}
```

# Assignments:

1. The distance between two cities (in km.) is input through the keyboard. Write a program to convert this distance in meters, feet, inches and centimetres.

2. Write a C program to interchange the value of two variables using third one.

3. Take the radius of a circle as input and calculate its area and perimeter accordingly.

4. If the ages of Ram, Shyam and Ajay are input through the keyboard, write a program to determine the youngest of the three.

5. Write a C program to check whether a triangle is valid or not, when the three angles of the triangle are entered through keyboard. A triangle is valid if the sum of all the three angles is equal to 180 degrees.

6. Write a C program to print the multiplication table of an integer taking as input through keyboard.

7. Write a C program to print all the perfect numbers within a given range.

8. Take an integer and write a C program to check whether it is palindrome or not.

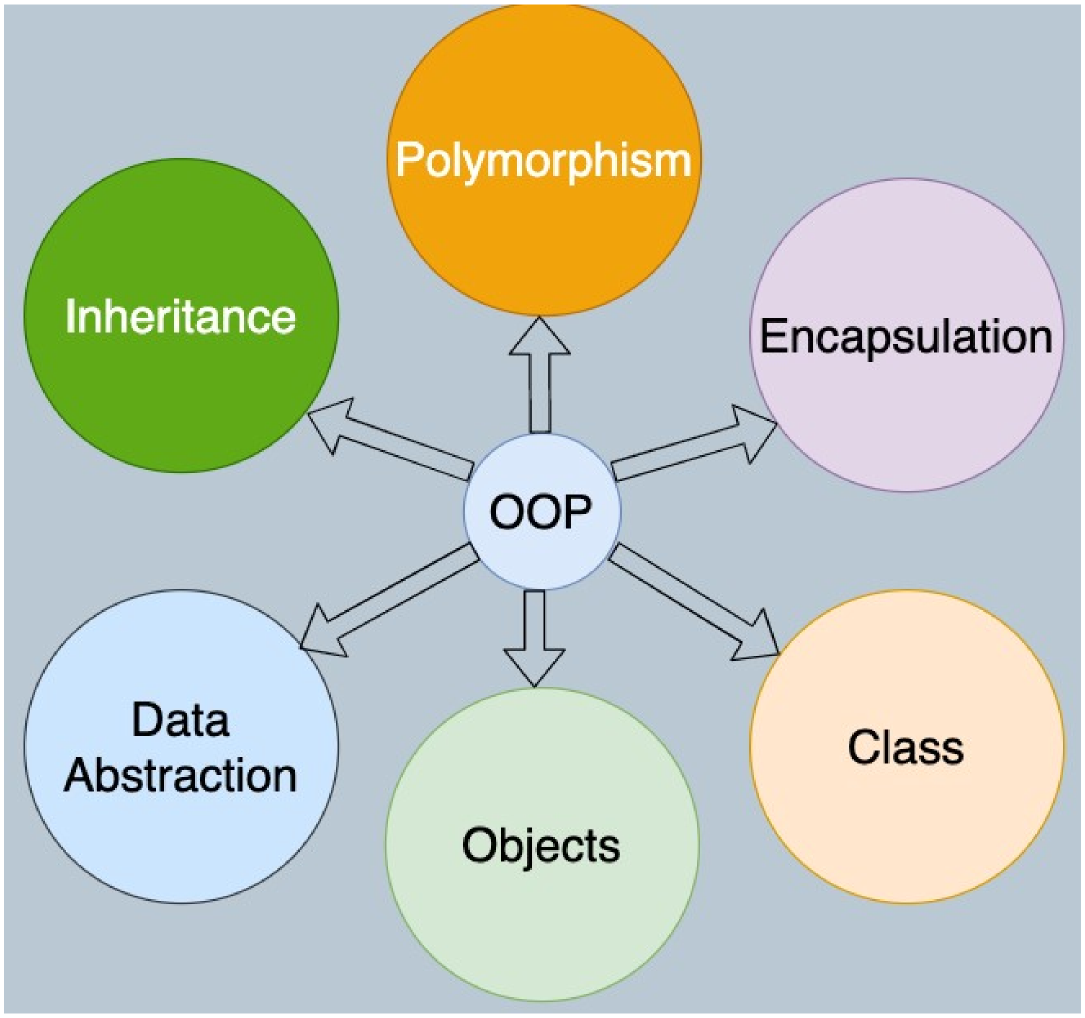
9. Write a C program to determine the LCM of two integers.

10. Add two numbers using function call in C program.

# Session 9

## What is Object-oriented Programming?

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic.



## Looking for the right definitions?

**Definition:** Object-oriented programming (OOP) is a way of writing computer programs that's based on the idea of "objects." Think of objects like things in the real world, such as cars, animals, or people.

## Characteristics of OOPs

**Objects:**Objects are like things in the real world. They have properties (like color, size, or type) and actions (like moving, eating, or talking).
**Classes:** Classes are like blueprints for creating objects. They define what properties and actions objects of that type can have.
**Encapsulation:** This is like putting things in a box. It means keeping the details of how an object works hidden, so you can use it without worrying about how it's made.
**Inheritance:** Inheritance is like passing traits from parents to children. It lets one class inherit properties and actions from another class. This helps avoid repeating code.
**Polymorphism:** This is like being able to use different tools in the same way. It lets you treat objects of different classes in a similar way, even if they do things differently.
Abstraction: Abstraction is like simplifying things. It's about focusing on what's important and ignoring the details that don't matter right now.

So, in simple terms, OOP is about creating objects that have properties and actions, organizing them into classes, and using concepts like encapsulation, inheritance, polymorphism, and abstraction to build flexible and understandable code.

## What is C?

C is a general-purpose programming language created by Dennis Ritchie at the Bell Laboratories in 1972.

It is a very popular language, despite being old. The main reason for its popularity is because it is a fundamental language in the field of computer science.

C is strongly associated with UNIX, as it was developed to write the UNIX operating system.

---

## Why Learn C?

- It is one of the most popular programming languages in the world
- If you know C, you will have no problem learning other popular programming languages such as Java, Python, C++, C#, etc, as the syntax is similar
- C is very fast, compared to other programming languages, like Java and Python
- C is very versatile; it can be used in both applications and technologies

---

## Difference between C and C++

- C++ was developed as an extension of C, and both languages have almost the same syntax
- The main difference between C and C++ is that C++ support classes and objects, while C does not

## What is C++?

C++ is a cross-platform language that can be used to create high-performance applications.

C++ was developed by Bjarne Stroustrup, as an extension to the C language.

C++ gives programmers a high level of control over system resources and memory.

The language was updated 4 major times in 2011, 2014, 2017, and 2020 to C++11, C++14, C++17, C++20.

---

## Why Use C++

C++ is one of the world's most popular programming languages.

C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.

C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

C++ is portable and can be used to develop applications that can be adapted to multiple platforms.

C++ is fun and easy to learn!

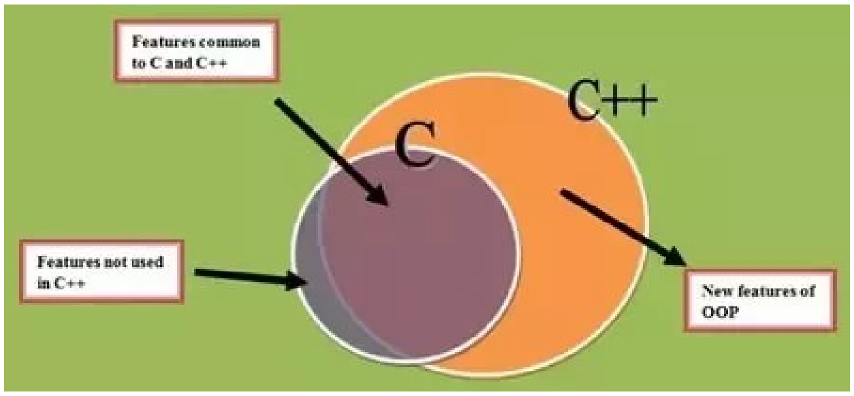
As C++ is close to C, C# and Java, it makes it easy for programmers to switch to C++ or vice versa.

---

## Difference between C and C++

C++ was developed as an extension of C, and both languages have almost the same syntax.

The main difference between C and C++ is that C++ support classes and objects, while C does not.



## What is Java?

Java is a popular programming language, created in 1995.

It is owned by Oracle, and more than 3 billion devices run Java.

It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection
- And much, much more!

---

## Why Use Java?

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)

- It is one of the most popular programming languages in the world
- It has a large demand in the current job market
- It is easy to learn and simple to use
- It is open-source and free
- It is secure, fast and powerful
- It has huge community support (tens of millions of developers)
- Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa

## What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

### What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.

- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

## Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

# C vs. C++ vs. Python vs. Java

Programs are a big part of our daily lives now. Almost everything is digital and connected through the Internet. People still really like using C, C++, Java, and Python for programming. C is called a middle-level language because it has features from both low-level and high-level languages. C++ is popular because it's fast and gets turned into code really quickly. It's often the first language new programmers learn. Java is used a lot because it works on lots of different types of computers. Python is another popular language. It's easy to read and write quickly.

## C

**Composition**: C comprises statements, functions, and variables.
**Expandability**: It's easily expandable, allowing for the addition of new functionalities.
**Procedural:** C is a procedural programming language.
**Pointers:** It supports pointers, which are used to manipulate memory and optimize performance.
**Efficiency**: C is known for being quick and efficient.
**Keywords:** It has 32 keywords that are reserved for specific purposes.
**Procedural Programming:** Supports procedural programming paradigms.

**Built-in Operators:** C has built-in operators that simplify the process of building complex programs.

## C++

**Object-Oriented:** C++ is an object-oriented programming language that uses classes and objects.
**Versatility:** It's versatile and can be used for a wide range of applications including operating systems, browsers, and games.
**Programming Approaches:** Supports various programming approaches like procedural, functional, and object-oriented.
**Strength and Adaptability**: C++ is both strong and adaptable.
Utilization: Widely used in developing professional gaming software and high-performance applications.
**Features:** Provides power over computer memory and resources, used in machine learning, graphical user interfaces, and embedded systems.
**Similarities:** Similar to C# and Java, making it easy for programmers to switch between languages.
**Evolution:** Developed as an extension of C, introducing object-oriented programming paradigm to C.
**Overloading:** Allows for function overloading, enabling multiple functions with the same name but different parameters.
**Built from the Ground Up:** C++ is built from the ground up, offering control and efficiency.
**Keywords:** It has a total of 52 keywords reserved for specific purposes.
**Input and Output:** Uses channels `cin` and `cout` for input and output operations.
Memory Allocation: Supports new memory allocation operator for dynamic memory management.

## Python

**Powerful:** Python is a powerful programming language.
**Dynamic Binding and Typing:** It dynamically binds various operations and has auto dynamic typing capabilities.
**Simple Syntax:** Known for its simple syntax, making it popular among beginner programmers.
**Organized Packages and Plug-ins:** Python has organized packages and plug-ins, enhancing its usability.
**Code Readability:** Python's code readability is facilitated by extensive whitespace in its design philosophy.
**Object-Oriented:** It follows an object-oriented approach, aiding in writing logical and unambiguous code for applications.
**Database Connectivity:** Python can connect to database systems.

**Web Development:** Used for developing server-side web applications.
**Math and Prototyping:** Capable of handling difficult math and suitable for quick prototyping.
**High-Level Language:** Python is a high-level language.
**GUI Programming:** It supports GUI programming capabilities.

## Java

**Widely-Used:** Java is a popular programming language released in 1995.
**Safe and Secure:** It's known for its safety, object-oriented nature, and security.
**Ownership:** Oracle owns Java, and it's used on over 3 billion devices worldwide.
**Versatile Applications**: Java is used for various applications like web apps (with Spring Boot), desktop and mobile apps, data processing, and embedded systems.
**Automatic Garbage Collection:** Java automatically manages memory by deleting unreferenced objects, thanks to its Automatic Garbage Collection feature.
**APIs:** Provides APIs for almost every task, making development easier.
**Support for OOP Principles:** Supports key Object-Oriented Programming (OOP) principles like encapsulation, abstraction, and inheritance.
**Exception Handling:** Has a robust exception handling system, which helps in writing reliable code.
**Type-Checking System:** Java has a powerful type-checking system, ensuring type safety.
**Simple Syntax:** Java's syntax is straightforward and easy to understand.

## Differences Between C, C++ Python and Java

| C | C++ | JAVA | PYTHON |
|---|---|---|---|
| Compiled Language | Compiled Programming language | Compiled Programming Language | Interpreted Programming Language |

| | | | |
|---|---|---|---|
| Operator overloading is not supported. | Supports Overloading the operator | Overloading of the operator is not supported. | Overloading of the operator is supported |
| Multiple inheritance is not supported in C. | Allow for both single as well as multiple inheritance options. | Java provides partial multiple inheritance | Provides both single as well as multiple inheritance |
| Platform-specific | Platform dependent | Platform-unaffected | Platform independent |
| Threads are supported. | Threads are not supported on this platform. | Multithreading capability is built-in. | Multithreading is supported. |
| A small number of libraries are available. | Has a small number of library patrons | Many concepts, such as UI, are supported by the library. | It comes with a large library set that allows it to be used for AI, data science, and other applications. |
| Outside of the class, variables and functions are utilized. | Outside of the class, variables and functions are utilized. | Every line of code is contained within a class. | Functions and variables can be declared global. |

| | | | |
|---|---|---|---|
| Have a similar speed as C++ | C++ is a computer language that compiles quickly. | The Java Program Compiler is a little slower than the C++ Compiler. | Execution is delayed due to the employment of an interpreter. |
| Syntax rules are strictly followed. | Syntax rules are strictly followed. | Syntax rules are strictly followed. | It isn't necessary to use the semicolon ' ;'. |

## Exercises

1. Is C a compiled language?
2. Which language supports operator overloading?
3. Does Java support multiple inheritance?
4. Is Python platform-independent?
5. Are threads supported in Python?
6. How many libraries does C have?
7. Does C++ support outside class variables and functions?
8. Is Java platform-unaffected?
9. Are threads supported in C++?
10. How many libraries does Java have?
11. Are variables and functions declared outside the class in Java?
12. Is Python's execution delayed compared to C++?
13. Are syntax rules strictly followed in C?
14. Does C++ support multiple inheritance?
15. Are variables and functions declared outside the class in Python?
16. How does Python handle syntax rules regarding semicolons?
17. Does C++ compile quickly?
18. Is the Java program compiler slower than the C++ compiler?
19. Are syntax rules strictly followed in Python?
20. Is operator overloading supported in C?

## Assignments :

1. C Programming Assignment: Write a program to add two numbers using variables and print the result.

2. C++ Programming Assignment: Create a program to calculate the area of a rectangle using user input.
3. Java Programming Assignment: Develop a program to print "Hello, World!" on the console.
4. Python Programming Assignment: Write a script to ask the user for their name and greet them.
5. C Programming Assignment: Create a program to check if a number is even or odd.
6. C++ Programming Assignment: Write a program to find the maximum of two numbers entered by the user.
7. Java Programming Assignment: Develop a program to calculate the sum of digits of a number entered by the user.
8. Python Programming Assignment: Create a script to generate and print a list of even numbers up to a given limit.
9. C Programming Assignment: Write a program to print the Fibonacci series up to a certain number of terms.
10. C++ Programming Assignment: Create a program to check if a given number is a prime number.
11. Java Programming Assignment: Develop a program to find the factorial of a number entered by the user.
12. Python Programming Assignment: Write a script to reverse a given string.
13. C Programming Assignment: Create a program to convert temperature from Celsius to Fahrenheit.
14. C++ Programming Assignment: Write a program to swap two numbers without using a temporary variable.
15. Java Programming Assignment: Develop a program to calculate the area of a circle given its radius.
16. Python Programming Assignment: Create a script to calculate the factorial of a number using recursion.
17. C Programming Assignment: Write a program to find the largest element in an array.
18. C++ Programming Assignment: Create a program to calculate the area of a triangle given its base and height.
19. Java Programming Assignment: Develop a program to print the multiplication table of a given number.
20. Python Programming Assignment: Write a script to find the sum of all elements in a list.

# Session 10

# Python Introduction

Python is a versatile programming language created by Guido van Rossum in 1991. It's widely used for various purposes including web development, software development, mathematics, and system scripting.

## Here's what Python can do:

- Create web applications on servers.
- Facilitate workflows alongside other software.
- Interact with databases and handle file operations.
- Process big data and perform complex mathematical operations.
- Enable rapid prototyping and develop production-ready software.

Python's appeal lies in its platform compatibility (Windows, Mac, Linux, Raspberry Pi, etc.), simple syntax akin to English, and its ability to write concise code. Its interpreter system allows immediate code execution, speeding up prototyping. Python supports procedural, object-oriented, and functional programming paradigms.

## Key points to remember:

- Python 3 is the latest major version, but Python 2 remains popular despite receiving only security updates.
- Python can be written in a text editor or IDEs like Thonny, PyCharm, Netbeans, or Eclipse, especially useful for managing large codebases.

## Python's syntax sets it apart:

- Designed for readability, it resembles English with influences from mathematics.
- Uses new lines to complete commands instead of semicolons or parentheses.
- Relies on indentation for defining scope (e.g., loops, functions, and classes), unlike curly brackets used in other languages.

In Python, you can quickly test code without writing it in a file using the command line. Here's how you can do it on Windows, Mac, or Linux:

1. Open your command line interface.
2. Type "python" or "py" if "python" doesn't work.
3. You'll enter the Python command line interface, where you can write Python code directly.

For example, to print "Hello, World!", you'd type:

```
>>> print("Hello, World!")
```

When you're done, just type "exit()" to leave the Python command line.

Python's indentation is crucial. Unlike other languages where indentation is for readability only, Python uses it to indicate code blocks. For instance:

```
>>> if 5 > 2:
        print("Five is greater than two!")
```

Leaving out indentation will cause an error. You can use any number of spaces, but they must be consistent within the same block.

Comments start with "#" in Python:

```
# This is a comment

print("Hello, World!")
```

You can also place comments at the end of a line:

```
>>> print("Hello, World!") # This is a comment
```

Comments are useful for explanations or preventing code execution:

```
# print("Hello, World!")

print("Cheers, Mate!")
```

## Variable Declaration in Python

Variables are created dynamically in Python

**x = 5**

**y = "Hello"**

No need to declare variables; assigning a value creates them.

In Python, variable names follow certain rules:

1. Variable names can be short (like x and y) or descriptive (like age, car_name, total_volume).

2. Rules for Python variables:

   - They must start with a letter or underscore.
   - They cannot start with a number.
   - They can only contain alphanumeric characters and underscores.
   - They are case-sensitive (e.g., age, Age, and AGE are different variables).
   - They cannot be Python keywords.

**Example of legal variable names:**

my_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

Example of illegal variable names:

2myvar = "John"

my-var = "John"

my var = "John"

Python allows you to assign values to multiple variables in one line:

**>>> x, y, z = 10, 20, 30**

You can also assign the same value to multiple variables in one line:

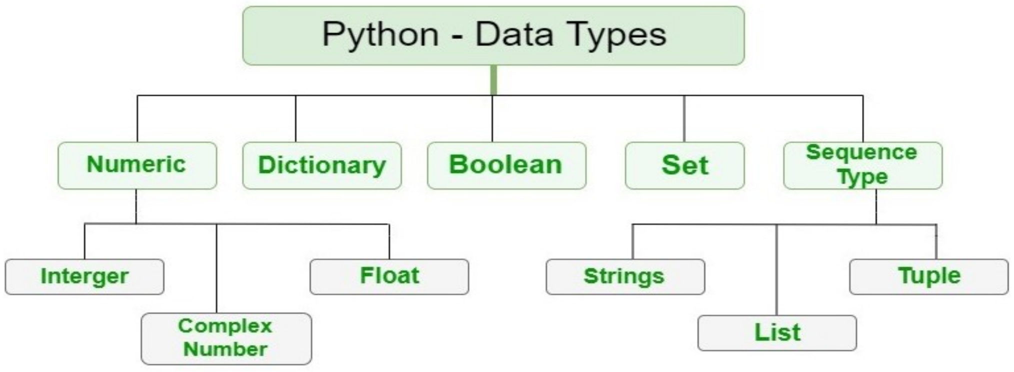
**>>> x = y = z = 10**

This makes it convenient to initialize multiple variables with the same value.

### Python Data types

Python Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, Python data types are classes and variables are instances (objects) of these classes. The following are the standard or built-in data types in Python:

- **Numeric**
- **Sequence Type**
- **Boolean**
- **Set**
- **Dictionary**
- **Binary Types( memoryview, bytearray, bytes)**



## What are Python Data Types?

To define the values of various data types of Python and check their data types we **use the type() function.** Consider the following examples.

This code assigns variable **'x'** different values of various Python data types. It covers **string**, **integer**, **float**, **complex**, **list**, **tuple**, **range**, **dictionary**, **set**, **frozenset**, **boolean**, **bytes**, **bytearray**, **memoryview**, and the special value **'None'** successively. Each assignment replaces the previous value, making **'x'** take on the data type and value of the most recent assignment.

```
x = "Hello World"

x = 50

x = 60.5

x = 3j

x = ["geeks", "for", "geeks"]

x = ("geeks", "for", "geeks")

x = range(10)

x = {"name": "Suraj", "age": 24}

x = {"geeks", "for", "geeks"}

x = frozenset({"geeks", "for", "geeks"})

x = True

x = b"Geeks"

x = bytearray(4)

x = memoryview(bytes(6))

x = None
```

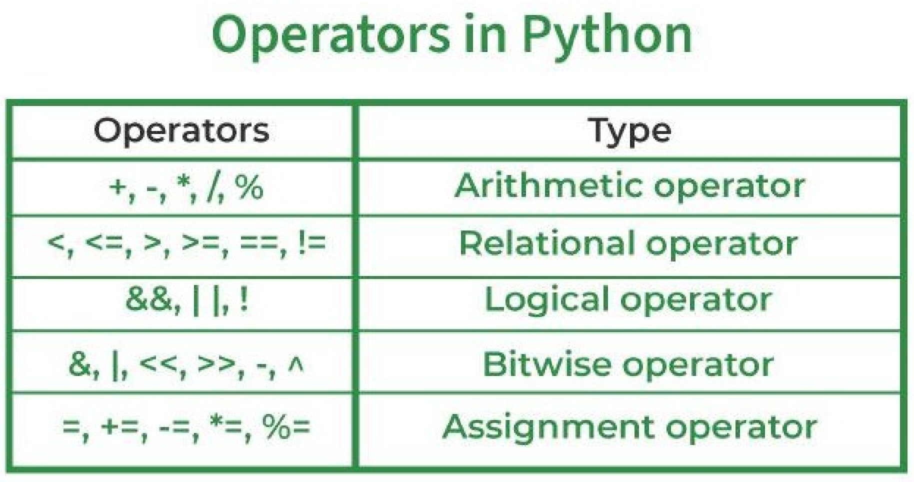
# Operators in Python

In Python programming, Operators in general are used to perform operations on values and variables. These are standard symbols used for logical and arithmetic operations. In this article, we will look into different types of **Python operators.**

- OPERATORS: These are the special symbols. Eg- + , * , /, etc.
- OPERAND: It is the value on which the operator is applied.

## Types of Operators in Python

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Identity Operators and Membership Operators



Operators in Python

| Operators | Type |
|---|---|
| +, -, *, /, % | Arithmetic operator |
| <, <=, >, >=, ==, != | Relational operator |
| &&, \| \|, ! | Logical operator |
| &, \|, <<, >>, -, ^ | Bitwise operator |
| =, +=, -=, *=, %= | Assignment operator |

### Precedence of Arithmetic Operators in Python

The precedence of Arithmetic Operators in Python is as follows:

1. P – Parentheses
2. E – Exponentiation
3. M – Multiplication (Multiplication and division have the same precedence)
4. D – Division
5. A – Addition (Addition and subtraction have the same precedence)
6. S – Subtraction

# Control Statements in Python

Loops are employed in Python to iterate over a section of code continually. Control statements are designed to serve the purpose of modifying a loop's execution from its default behaviour. Based on a condition, control statements are applied to alter how the loop executes. In this tutorial, we are covering every type of control statement that exists in Python.

## Control Statements in Python

### if Statements

The if statement is arguably the most used statement to control loops. For instance:

```
# Python program to show how if statements control loops

n = 5
```

```
for i in range(n):
if i < 2:
        i += 1
if i > 2:
        i -= 2
print(i)
```

**Output:**

```
1

2

2

1

2
```

## Break Statements

In Python, the break statement is employed to end or remove the control from the loop that contains the statement. It is used to end nested loops (a loop inside another loop), which are shared with both types of Python loops. The inner loop is completed, and control is transferred to the following statement of the outside loop.

**Code**

```
# Python program to show how to control the flow of loops with the break statement

Details = [[19, 'Itika', 'Jaipur'], [16, 'Aman', 'Bihar']]
for candidate in Details:
    age = candidate[0]
if age <= 18:
break
print (f"{candidate[1]} of state {candidate[2]} is eligible to vote")
```

## Continue Statements

When a Python interpreter sees a continue statement, it skips the present iteration's execution if the condition is satisfied. If the condition is not satisfied, it allows the implementation of the current iteration. It is employed to keep the program running even when it meets a break while being executed.

**Code**

```
# Printing only the letters of the string
for l in 'I am a coder':
if l == ' ':
continue
print ('Letter: ', l)
```

```
Letter:  I

Letter:  a

Letter:  m

Letter:  a

Letter:  c

Letter:  o

Letter:  d

Letter:  e

Letter:  r
```

## Pass Statements

If the condition is met, the pass statement, or a null operator, is used by the coder to leave it as it is. Python's pass control statement changes nothing and moves on to the following iteration without stopping the execution or skipping any steps after completing the current iteration.

A coder can put the pass statement to prevent the interpreter from throwing an error when a loop or a code block is left empty.

**Code**

```
# Python program to show how to create empty code blocks using a pass statement
for l in 'Python':
if l == 't':
pass
print('Letter: ', l)
```

```
Letter:  P

Letter:  y

Letter:  t

Letter:  h

Letter:  o

Letter:  n
```

## Python If Else Statements – Conditional Statements

In both real life and programming, decision-making is crucial. We often face situations where we need to make choices, and based on those choices, we determine our next actions. Similarly, in programming, we encounter scenarios where we must make decisions to control the flow of our code.

Conditional statements in Python play a key role in determining the direction of program execution. Among these, If-Else statements are fundamental, providing a way to execute different blocks of code based on specific conditions. As the name suggests, If-Else statements offer two paths, allowing for different outcomes depending on the condition evaluated.

### Types of Control Flow in Python

- Python If Statement
- Python If Else Statement
- Python Nested If Statement
- Python Elif
- Ternary Statement | ShortHand If Else Statement

## Python 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.

```
#if syntax Python

if condition:

  # Statements to execute if

  # condition is true
```

As we know, Python uses indentation to identify a block.

```
if condition:

  statement1

statement2

# Here if the condition is true, if block

# will consider only statement1 to be inside

# its block.
```

## Python 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 Python to execute a block of code when the Python if condition is false.

**Syntax:**

```
if (condition):

   # Executes this block if

   # condition is true

else:

   # Executes this block if

   # condition is false
```

## Python 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.

```
if (condition1):

  # Executes when condition1 is true

  if (condition2):

    # Executes when condition2 is true

  # if Block is end here

# if Block is end here
```

## Python Elif

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.

```
if (condition):
   statement
elif (condition):
   statement
else:
   Statement
```

# Loops in Python

## For, While and Nested Loops

Python programming language provides two types of Python loop checking time. In this article, we will look at Python loops and understand their working with the help of example – For loop and While loop to handle looping requirements. Loops in Python provides three ways for executing the loops.

While all the ways provide similar basic functionality, they differ in their syntax and condition-checking time. In this article, we will look at Python loops and understand their working with the help of examples.

## While Loop in Python

In Python, a while loop is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.

**Python While Loop Syntax:**

```
while expression:
    statement(s)
```

**Using else statement with While Loop in Python**

The else clause is only executed when your while condition becomes false. If you break out of the loop, or if an exception is raised, it won't be executed.

**Syntax of While Loop with else statement:**

```
while condition:
     # execute these statements
else:
     # execute these statements
```

## For Loop in Python

For loops are used for sequential traversal. For example: traversing a list or string or array etc. In Python, there is "for in" loop which is similar to for each loop in other languages. Let us learn how to use for loops in Python for sequential traversals with examples.

**For Loop Syntax:**

```
for iterator_var in sequence:
    statements(s)
```

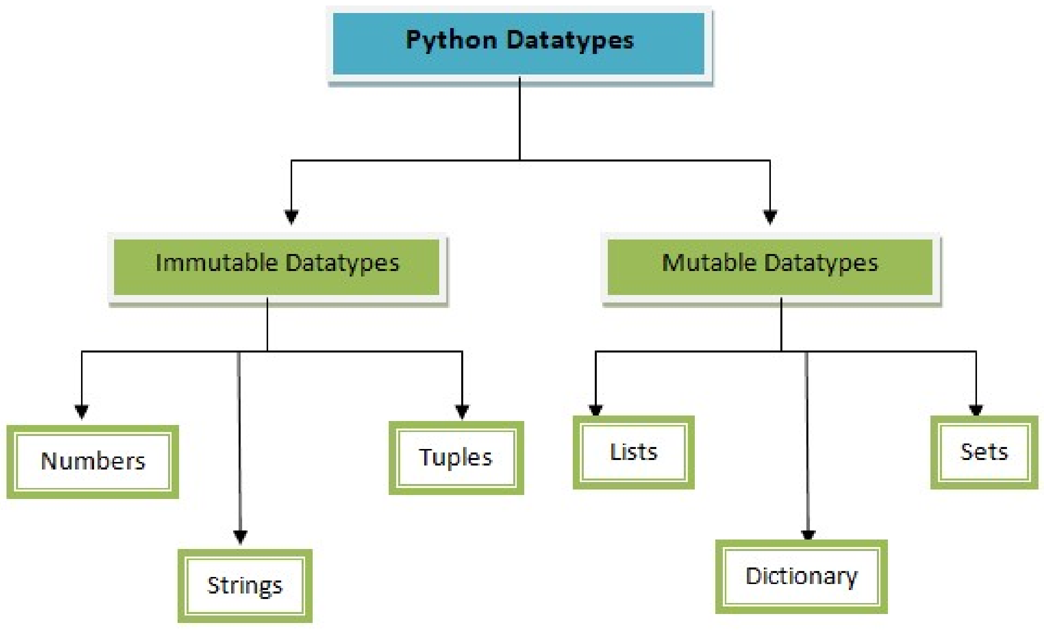
**Break Statement**

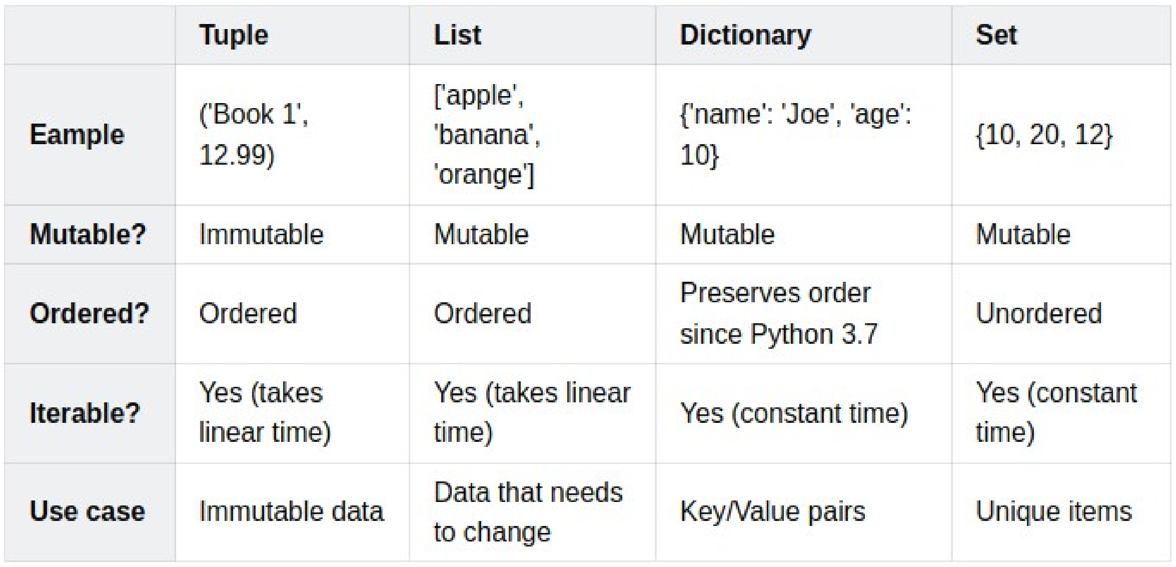
The break statement in Python brings control out of the loop.

**Pass Statement**

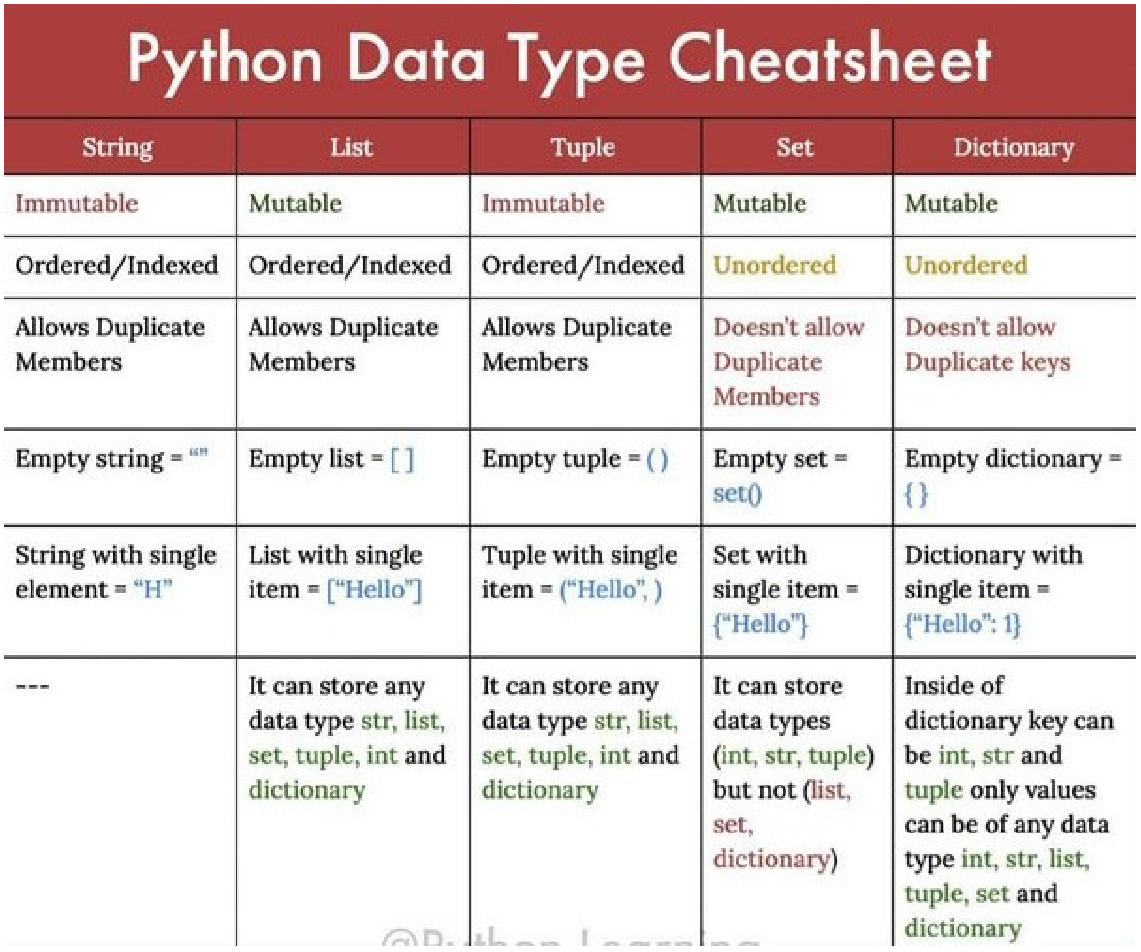
We use pass statements in Python to write empty loops. Pass is also used for empty control statements, functions and classes.

## Collection in Python





| | Tuple | List | Dictionary | Set |
|---|---|---|---|---|
| **Eample** | ('Book 1', 12.99) | ['apple', 'banana', 'orange'] | {'name': 'Joe', 'age': 10} | {10, 20, 12} |
| **Mutable?** | Immutable | Mutable | Mutable | Mutable |
| **Ordered?** | Ordered | Ordered | Preserves order since Python 3.7 | Unordered |
| **Iterable?** | Yes (takes linear time) | Yes (takes linear time) | Yes (constant time) | Yes (constant time) |
| **Use case** | Immutable data | Data that needs to change | Key/Value pairs | Unique items |



Python Data Type Cheatsheet

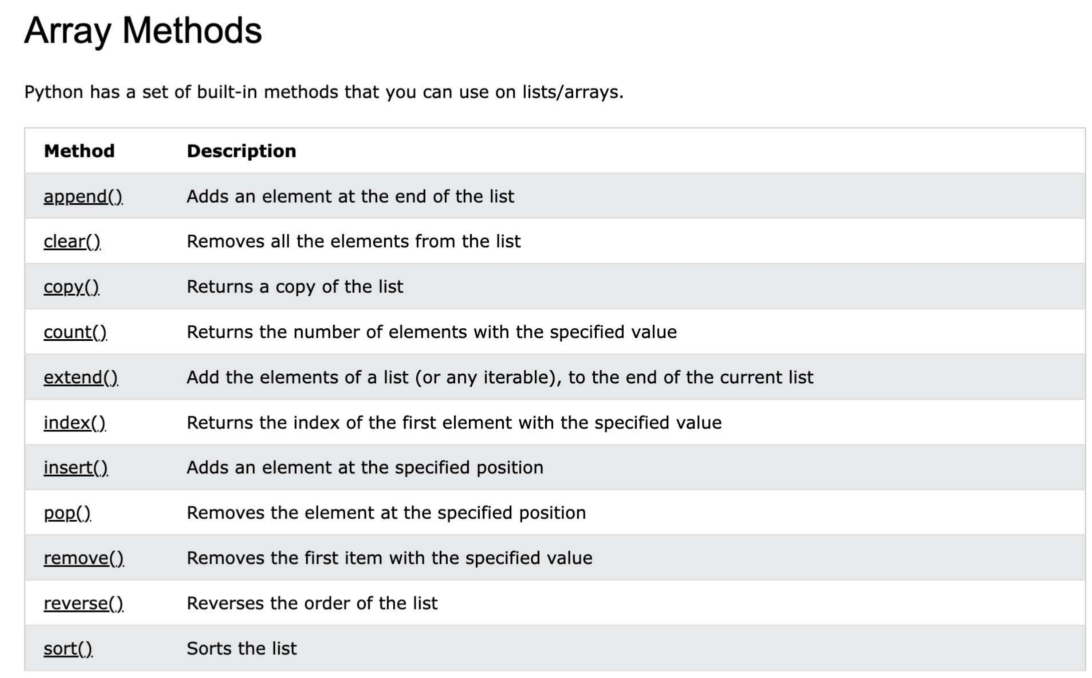
| String | List | Tuple | Set | Dictionary |
|---|---|---|---|---|
| Immutable | Mutable | Immutable | Mutable | Mutable |
| Ordered/Indexed | Ordered/Indexed | Ordered/Indexed | Unordered | Unordered |
| Allows Duplicate Members | Allows Duplicate Members | Allows Duplicate Members | Doesn't allow Duplicate Members | Doesn't allow Duplicate keys |
| Empty string = "" | Empty list = [ ] | Empty tuple = ( ) | Empty set = set() | Empty dictionary = { } |
| String with single element = "H" | List with single item = ["Hello"] | Tuple with single item = ("Hello", ) | Set with single item = {"Hello"} | Dictionary with single item = {"Hello": 1} |
| --- | It can store any data type str, list, set, tuple, int and dictionary | It can store any data type str, list, set, tuple, int and dictionary | It can store data types (int, str, tuple) but not (list, set, dictionary) | Inside of dictionary key can be int, str and tuple only values can be of any data type int, str, list, tuple, set and dictionary |

## What is an Array?

An array is a special variable, which can hold more than one value at a time.

## Array Methods

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).



### Array Methods

Python has a set of built-in methods that you can use on lists/arrays.

| Method | Description |
|---|---|
| append() | Adds an element at the end of the list |
| clear() | Removes all the elements from the list |
| copy() | Returns a copy of the list |
| count() | Returns the number of elements with the specified value |
| extend() | Add the elements of a list (or any iterable), to the end of the current list |
| index() | Returns the index of the first element with the specified value |
| insert() | Adds an element at the specified position |
| pop() | Removes the element at the specified position |
| remove() | Removes the first item with the specified value |
| reverse() | Reverses the order of the list |
| sort() | Sorts the list |

Python has a set of built-in methods that you can use on lists/arrays.

## 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 characters 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.

| Method | Description |
|---|---|
| capitalize() | Converts the first character to uppercase |

| | |
|---|---|
| casefold() | Converts string into lower case |
| center() | Returns a centered string |
| count() | Returns the number of times a specified value occurs in a string |
| encode() | Returns an encoded version of the string |
| endswith() | Returns true if the string ends with the specified value |
| expandtabs() | Sets the tab size of the string |
| find() | Searches the string for a specified value and returns the position of where it was found |
| format() | Formats specified values in a string |
| format_map() | Formats specified values in a string |
| index() | Searches the string for a specified value and returns the position of where it was found |
| isalnum() | Returns True if all characters in the string are alphanumeric |
| isalpha() | Returns True if all characters in the string are in the alphabet |
| isascii() | Returns True if all characters in the string are ascii characters |
| isdecimal() | Returns True if all characters in the string are decimals |
| isdigit() | Returns True if all characters in the string are digits |
| isidentifier() | Returns True if the string is an identifier |
| islower() | Returns True if all characters in the string are lower case |

| | |
|---|---|
| isnumeric() | Returns True if all characters in the string are numeric |
| isprintable() | Returns True if all characters in the string are printable |
| isspace() | Returns True if all characters in the string are whitespaces |
| istitle() | Returns True if the string follows the rules of a title |
| isupper() | Returns True if all characters in the string are upper case |
| join() | Converts the elements of an iterable into a string |
| ljust() | Returns a left justified version of the string |
| lower() | Converts a string into lower case |
| lstrip() | Returns a left trim version of the string |
| maketrans() | Returns a translation table to be used in translations |
| partition() | Returns a tuple where the string is parted into three parts |
| replace() | Returns a string where a specified value is replaced with a specified value |
| rfind() | Searches the string for a specified value and returns the last position of where it was found |
| rindex() | Searches the string for a specified value and returns the last position of where it was found |
| rjust() | Returns a right justified version of the string |
| rpartition() | Returns a tuple where the string is parted into three parts |
| rsplit() | Splits the string at the specified separator, and returns a list |

| | |
|---|---|
| rstrip() | Returns a right trim version of the string |
| split() | Splits the string at the specified separator, and returns a list |
| splitlines() | Splits the string at line breaks and returns a list |
| startswith() | Returns true if the string starts with the specified value |
| strip() | Returns a trimmed version of the string |
| swapcase() | Swaps cases, lower case becomes upper case and vice versa |
| title() | Converts the first character of each word to upper case |
| translate() | Returns a translated string |
| upper() | Converts a string into upper case |
| zfill() | Fills the string with a specified number of 0 values at the beginning |

## List in Python

Python Lists are just like dynamically sized arrays, declared in other languages (vector in C++ and ArrayList in Java). In simple language, a Python 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.*

| Method | Description |
|---|---|

| | |
|---|---|
| append() | Used for adding elements to the end of the List. |
| copy() | It returns a shallow copy of a list |
| clear() | This method is used for removing all items from the list. |
| count() | These methods count the elements. |
| extend() | Adds each element of an iterable to the end of the List |
| index() | Returns the lowest index where the element appears. |
| insert() | Inserts a given element at a given index in a list. |
| pop() | Removes and returns the last value from the List or the given index value. |

| | |
|---|---|
| remove() | Removes a given object from the List. |
| reverse() | Reverses objects of the List in place. |
| sort() | Sort a List in ascending, descending, or user-defined order |
| min() | Calculates the minimum of all the elements of the List |
| max() | Calculates the maximum of all the elements of the List |

## Tuples in Python

Python Tuple is a collection of objects separated by commas. In some ways, a tuple is similar to a Python list in terms of indexing, nested objects, and repetition but the main difference between both is that the Python tuple is immutable, unlike the Python list which is mutable.

| Method | Description |
|---|---|
| count() | Returns the number of times a specified value occurs in a tuple |
| index() | Searches the tuple for a specified value and returns the position of where it was found |

## Differences between List and Tuple in Python

| Sno | LIST | TUPLE |
|---|---|---|
| 1 | Lists are mutable | Tuples are immutable |
| 2 | The implication of iterations is Time-consuming | The implication of iterations is comparatively Faster |
| 3 | The list is better for performing operations, such as insertion and deletion. | A Tuple data type is appropriate for accessing the elements |
| 4 | Lists consume more memory | Tuple consumes less memory as compared to the list |
| 5 | Lists have several built-in methods | Tuple does not have many built-in methods. |
| 6 | Unexpected changes and errors are more likely to occur | Because tuples don’t change they are far less error-prone. |

## Set

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*.

| Method | Shortcut | Description |
|---|---|---|
| add() | | Adds an element to the set |
| clear() | | Removes all the elements from the set |
| copy() | | Returns a copy of the set |
| difference() | - | Returns a set containing the difference between two or more sets |
| difference_update() | -= | Removes the items in this set that are also included in another, specified set |
| discard() | | Remove the specified item |
| intersection() | & | Returns a set, that is the intersection of two other sets |
| intersection_update() | &= | Removes the items in this set that are not present in other, specified set(s) |
| isdisjoint() | | Returns whether two sets have a intersection or not |
| issubset() | <= | Returns whether another set contains this set or not |
| | < | Returns whether all items in this set is present in other, specified set(s) |

| | | |
|---|---|---|
| issuperset() | >= | Returns whether this set contains another set or not |
| | > | Returns whether all items in other, specified set(s) is present in this set |
| pop() | | Removes an element from the set |
| remove() | | Removes the specified element |
| symmetric_difference() | ^ | Returns a set with the symmetric differences of two sets |
| symmetric_difference_update() | ^= | Inserts the symmetric differences from this set and another |
| union() | \| | Return a set containing the union of sets |
| update() | \|= | Update the set with the union of this set and others |

## Dictionary

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.

Python has a set of built-in methods that you can use on dictionaries.

| Method | Description |
|---|---|
| clear() | Removes all the elements from the dictionary |
| copy() | Returns a copy of the dictionary |

| | |
|---|---|
| fromkeys() | Returns a dictionary with the specified keys and value |
| get() | Returns the value of the specified key |
| items() | Returns a list containing a tuple for each key value pair |
| keys() | Returns a list containing the dictionary's keys |
| pop() | Removes the element with the specified key |
| popitem() | Removes the last inserted key-value pair |
| setdefault() | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| update() | Updates the dictionary with the specified key-value pairs |
| values() | Returns a list of all the values in the dictionary |

# Session 11

1. Python Command Line Interface:

- Write a Python script to accept user input for their name and greet them accordingly.
- Develop a Python program to perform basic arithmetic operations (addition, subtraction, multiplication, division) based on user input.
- Create a Python script to convert temperature units (e.g., Celsius to Fahrenheit) based on user input.
- Build a Python application to list files in a directory specified by the user.
- Write a Python script to calculate the factorial of a number entered by the user.

2. Variable Declaration in Python:

- Write a Python script to swap the values of two variables without using a temporary variable.
- Develop a Python program to calculate the area of a circle given its radius using a variable for π.
- Create a Python script to concatenate two strings and store the result in a variable.
- Build a Python application to convert miles to kilometers using variables for conversion factors.
- Write a Python script to calculate the volume of a cylinder given its radius and height using variables.
- Develop a Python program to store student names and their corresponding scores in variables and perform calculations like average score.
- Create a Python script to format and print a message using variables for placeholders.
- Build a Python application to calculate compound interest based on user input for principal amount, interest rate, and time.
- Write a Python script to generate a random number within a specified range and store it in a variable.
- Develop a Python program to calculate the perimeter of a rectangle using variables for length and width.
-

3. Arithmetic Operators:

- Write a Python script to add two numbers and print the result.
- Develop a Python program to subtract two numbers and display the result.
- Create a Python script to multiply two numbers and output the product.
- Build a Python application to divide two numbers and print the quotient.
- Write a Python script to calculate the remainder when dividing two numbers.

## 4. Comparison Operators:

- Create a Python program to check if two numbers are equal.
- Develop a Python script to verify if a number is greater than another number.
- Write a Python program to check if a number is less than or equal to another number.
- Build a Python script to compare two strings for equality.
- Create a Python program to test if a string is not equal to another string.

## 5. Logical Operators:

- Write a Python script to check if a number is greater than 5 and less than 10.
- Develop a Python program to verify if a number is divisible by both 2 and 3.
- Create a Python script to test if a number is either positive or negative.
- Build a Python program to check if a number is not equal to 0 and less than 100.
- Write a Python script to verify if a string contains both 'a' and 'b'.

## 6. Bitwise Operators:

- Create a Python program to perform bitwise AND operation on two numbers.
- Develop a Python script to perform bitwise OR operation on two numbers.
- Write a Python program to perform bitwise XOR operation on two numbers.
- Build a Python script to perform bitwise NOT operation on a number.
- Create a Python program to left shift a number by 2 bits.

## 7. Assignment Operators:

- Write a Python script to increment a variable by 1 using the assignment operator.
- Develop a Python program to add 5 to a variable using the assignment operator.
- Create a Python script to multiply a variable by 2 and assign the result back to the variable.

- Build a Python program to divide a variable by 3 and update the variable with the quotient.
- Write a Python script to perform bitwise AND operation between a variable and 2, updating the variable with the result.

## 8.if Statements:

- Write a Python script to check if a number is positive.
- Develop a Python program to determine if a given number is even.
- Create a Python script to check if a string is empty.
- Build a Python program to verify if a user's age is above 18.
- Write a Python script to check if a character is a vowel or consonant.

## 9. Break Statements:

- Create a Python program to find the first occurrence of a specific element in a list and stop the search once found.
- Write a Python script to search for a keyword in a text file and stop reading the file once the keyword is found.
- Develop a Python program to iterate through a list of numbers and break the loop if a negative number is encountered.
- Build a Python script to find the first prime number greater than 100 and stop searching once found.

## 10. Continue Statements:

- Write a Python program to print all numbers between 1 and 10 except 5 using a continue statement.
- Create a Python script to iterate through a list of names and skip printing any name starting with the letter 'A'.
- Develop a Python program to print all even numbers between 1 and 20 using a continue statement.
- Build a Python script to calculate the sum of all numbers between 1 and 50, excluding multiples of 5, using a continue statement.

## 11. Pass Statements:

- Create a Python program to define a function without any implementation using the pass statement.
- Write a Python script to create a class with placeholders for methods using pass statements.
- Develop a Python program to create an empty loop to be filled later using pass statements.
- Build a Python script to define conditional blocks with no actions using pass statements.

## 12. For Loops:

- Write a Python script to print numbers from 1 to 10 using a for loop.

- Develop a Python program to print the elements of a list using a for loop.
- Create a Python script to print each character of a string using a for loop.
- Build a Python program to calculate the sum of numbers from 1 to 100 using a for loop.
- Write a Python script to iterate over a tuple and print each element using a for loop.

## 13. While Loops:

- Create a Python program to print numbers from 1 to 10 using a while loop.
- Develop a Python script to print even numbers between 1 and 20 using a while loop.
- Write a Python program to find the factorial of a number entered by the user using a while loop.
- Build a Python script to keep asking the user for input until they enter 'quit' using a while loop.
- Create a Python program to generate Fibonacci numbers up to a specified limit using a while loop.

## 14.Nested Loops:

- Write a Python script to print a multiplication table (1 to 10) using nested loops.
- Develop a Python program to print a pattern of stars in the shape of a triangle using nested loops.
- Create a Python script to print a pattern of numbers in the shape of a square using nested loops.
- Build a Python program to print a pattern of alphabets in the shape of a diamond using nested loops.
- Write a Python script to print a pattern of numbers in the shape of a pyramid using nested loops.

## 15.List:

- Write a Python script to create a list of numbers and print it.
- Develop a Python program to create a list of strings and print each string.
- Create a Python script to append an element to an existing list and print the updated list.
- Build a Python program to remove an element from a list and print the updated list.
- Write a Python script to access elements of a list using indexing and print them.

## 16.Tuple:

- Create a Python program to create a tuple of fruits and print it.

- Develop a Python script to access elements of a tuple using indexing and print them.
- Write a Python program to unpack a tuple into individual variables and print them.
- Build a Python script to concatenate two tuples and print the result.
- Develop a Python program to convert a tuple into a list and print the list.

## 17. Set:

- Write a Python script to create a set of colors and print it.
- Create a Python program to add an element to an existing set and print the updated set.
- Develop a Python script to remove an element from a set and print the updated set.
- Build a Python program to perform set intersection between two sets and print the result.
- Write a Python script to check if a specific element exists in a set and print the result.

## 18. Dictionary:

- Create a Python program to create a dictionary of students and their ages and print it.
- Write a Python script to access values in a dictionary using keys and print them.
- Develop a Python program to add a new key-value pair to an existing dictionary and print the updated dictionary.
- Build a Python script to remove a key-value pair from a dictionary and print the updated dictionary.
- Write a Python program to check if a specific key exists in a dictionary and print the result.