UNIT-I
Introduction to Computer System

CHAPTER-1
COMPUTER SYSTEM

TOPIC-1
Basic Computer Organisation

Revision Notes

- **COMPUTER :** It is an electronic device which manipulates the data according to the list of instructions. It has the ability to store, retrieve and process data. A computer is used to type documents, send E-mails and browse the Internet. It is also used to handle accounting, database management, presentations, games and so on.
- **FUNCTIONING OF A COMPUTER**

  Computer performs four basic functions which are as follows :

  (i) **Input :** Information or data that is entered into a computer is called input. It sends data and instructions to the Central Processing Unit (CPU).

  (ii) **Processing :** It is the sequence of actions taken on data to convert it into information which is meaningful and useful to the user. It can be calculations, comparisons or decisions taken by the computer.

  (iii) **Output :** It makes processed data available to the user. It is mainly used to display the desired result to the user as per input instructions.

  (iv) **Storage :** It stores data and programs permanently. It is used to store information during the time of program execution and possible to get any type of information from it.
- **COMPONENTS OF COMPUTER :** A computer consists of following two main components :

  (i) Hardware (ii) Software

  (i) Hardware components are those components which we can touch, feel or see.

  (ii) Software is that part of computer which we cannot see or touch. It is a set of instructions given to computer to make the computer work.

  The hardware consists of the following four main components :

  1. Input Devices **2.** Output Devices **3.** CPU **4.** Memory

  1. **Input Devices :** The computer accepts coded information through input devices by the user. It is a device that is used to give required information to the computer. An input device/unit performs the following functions :

     (i) It accepts the instructions and data from the user.

     (ii) It converts these instructions and data in computer in acceptable format.

     (iii) It supplies the converted instructions and data to the computer system for further processing.

     Some of commonly used input devices are

     - **Keyboard :** The user can type text and command using this device. It is used to enter data or information in a computer system, which may be in numeric form or alphabetic form.
     - **Mouse :** It is a pointing device which provides a means to input data and commands in graphic form by selecting through moving an arrow called pointer on monitor. The mouse may be used to position the cursor on screen, move an object by dragging or select an object by clicking.
     - **Touch Screen :** It is an input device that accepts input when the user places a fingertip on the computer screen.

- **Scanner :** It is an optical input device that uses light as an input source to convert an image into an electronic form that can be stored on the computer.

2. **Output Devices :** This device sends the processed result to the user. It is mainly used to display the desired result to the user as per input instructions. The following functions are performed by output device/unit :
   - It accepts the results produced by the computer which are in coded form and hence cannot be easily understood by the user.
   - It converts these coded results to human acceptable form.
   - It supplies the converted results to the user.

   Some commonly used output devices are
   - **Monitor :** It is provided along with the computer to view the display result. It is known as Visual Display Unit (VDU). An image on monitor is created by a configuration of dots, also known as pixels.
   - **Printer :** It prints information and data from the computer into a paper. It can print documents in colour as well repeated twice as in black and white. Printers are divided into two basic categories – Impact printers and Non-impact printers.
   - **Speaker :** It is an output device that receives sound in the form of electric current. It needs a sound card connected to a CPU that generates sound via a card.
3. **Central Processing Unit (CPU) :** It consists a set of registers, arithmetic logic unit and control unit, which together interprets and executes instructions in assembly language.

   The primary functions of the CPU are as follows :
   - The CPU transfers instructions and input data from main memory to registers i.e. internal memory.
   - The CPU executes the instructions in the stored sequence.
   - When necessary, CPU transfers output data from registers to main memory.
   - Central Processing Unit is often called the brain of computer. The CPU is fabricated as single Integrated Circuit (IC) and is also known as microprocessor.
   - A CPU controls all the internal and external devices and performs arithmetic and logic operations.

   The CPU consists of following main sub-systems :

   (i) Arithmetic Logic Unit (ALU) : ALU contains the electronic circuitry that executes all arithmetic and logical operations on the available data. ALU uses registers to hold the data that is being processed.

   Most ALUs can perform the following operations :
   - Logical Operations (AND, NOT, OR, XOR)
   - Arithmetic operations (addition, subtraction, multiplication, division)
   - Shifting bit pattern to the left or right with or without sign extension.
   - Comparison operations (=, <, <=, >, >=)

   > **Registers :** These are used to quickly accept, store and transfer data and instructions that are being used immediately by the CPU. These registers are at the top of the memory hierarchy and are the fastest way for the system to manipulate data.

   (ii) Control Unit (CU) : CU coordinates with the input and output devices of a computer. It directs the computer to carry out stored program instructions by communicating with the ALU and the registers. It organises the processing of data and instructions.

   The basic functions of control unit is to fetch the instruction stored in the main memory, identify the operations and the devices involved in it and accordingly generate control signals.
4. **Memory :** The computer memory is one of the most important elements in a computer system. It stores data and instructions required during the processing of data and output the results.

   Storage may be required for a limited period of time, instantly or for an extended period of time. It also relates to many devices that are responsible for storing data on a temporary or a permanent basis.

   Type of Memory : In general, the memory is classified into two categories as follows

   (i) Primary Memory **(ii)** Secondary Memory

   (i) Primary Memory : The memory also called main memory unit that communicates directly with the CPU is called main memory or internal memory or primary memory.

The primary memory allows the computer to store data for immediate manipulation and to keep track of what is currently being processed. It has limited storage capacity.

Main memory is volatile in nature, it means that when the power is turned OFF, the contents of this memory are lost forever.

Primary memory can be further classified in two categories as :

(a) Random Access Memory (RAM) : It is used for temporary storage of input data, output data and intermediate results. It is also known as read/write memory that allows CPU to read as well as write data or instructions into it.

There are two types of RAM are

- **Dynamic RAM (DRAM) :** It must be refreshed continually to store information. DRAM is made up of memory cells where each cell is composed of one capacitor and one transistor.
- **Static RAM (SRAM) :** It need not be refreshed periodically. It uses multiple transistors for each memory cell. It does not use capacitor. SRAM is often used as a cache memory due to its high speed.

(b) Read Only Memory (ROM) : It does not lose its contents when the power is switched OFF. ROM is also known as non-volatile memory or permanent storage.

ROM can have data and instructions written to it only one time.

Once a ROM chip is programmed at the time of manufacturing, it cannot be reprogrammed or rewritten.

There are three categories of ROM as follows :

- **Programmable ROM(PROM) :** It is also non - volatile in nature. Once a PROM has been programmed, its contents can never be changed. It is one time programmable device.

  These types of memories are found in video game consoles, mobile phones, implantable medical devices and high definition multimedia interface.
- **Erasable Programmable ROM (EPROM) :** It is similar to PROM, but can be erased by exposure to strong, ultraviolet light, then rewritten. So it is also known as Ultraviolet Erasable Programmable ROM (UVEPROM).
- **Electrically Erasable Programmable ROM (EEPROM) :** It is similar to EPROM, but it can be erased electrically then rewritten electrically and the burning process is reversible by exposure to electric pulses. It is the most flexible type of ROM and is now commonly used for holding BIOS.

(ii) Secondary Memory/Storage : This memory stores much larger amounts of data and information for extended periods of time. Data in secondary memory cannot be processed directly by the CPU, it must first be copied into primary memory *i.e*, RAM. It is the slower and cheaper form of memory. Secondary storage is used to store data and program when they are not being processed. It is also non - volatile in nature. Due to this, the data remain in the secondary storage as long as it is not over written or deleted by the user. It is a permanent storage.

Secondary Memory devices are different types as follows :

(a) Hard Disk Drive (HDD) : It is a non-volatile and random access digital data storage device. HDD is a data storage device used for storing and retrieving digital data information using rotating disks (platters) coated with magnetic material. All programs of a computer are installed in hard disk. It is not required fixed *i.e.* cannot be removed from the drive. It consists of a spindle that holds non-magnetic flat circular disk; called platter, which requires two read/write heads, that are used to write and read information from a platter. All the read/write heads are attached to a single access arm so that they cannot move independently.

The information is recorded in bands; each band of information is called a track. Each platter has the same number of tracks and a track location that cuts across all platters is called a cylinder.

The track are divided repeated twice into pie-shaped sections known as sectors.

(b) Floppy disk (Diskette) : It is used to store data but it can store small amount of data and it is slower to access than hard disks. Floppy disks are round in shape and a thin plastic disk is coated with iron oxides. Data is retrieved or recorded on the surface of the disk through a slot on the envelope. Floppy disk can be removable from the drive. Floppy disk is available in three sizes as 8 inch, 5'.25" inch and 3'.50" inch.

(c) Magnetic Tapes : These tapes are made up of a plastic film-type material repeated coated with magnetic materials to store data permanently. Data can be read as well as recorded. It is usually 12.5 mm to 25

mm wide and 500 m to 1200 m long. Magnetic tapes hold not required the maximum data, which can be accessed sequentially. They are generally used to store backup data or that type of data, which is not recurrently used or to transfer data from one system to another.

(d) Compact Disc (CD) : It is the most popular and the least expensive type of optical disc. A CD is capable of being used as a data storage device along with storing of digital audio. The files are stored on this particular contiguous sector.

CD are categorised into three main types as follows :

(i) CD-ROM (Compact Disc-Read Only Memory)

(ii) CD-R (Compact Disc-Recordable)

(iii) CD-RW (Compact Disc-Rewritable)

(e) Digital Video Disc (DVD) : DVD is also known as Super Density Disc (SDD) or Digital Versatile Disc (DVD). It is an optical disc storage media. DVDs offer higher storage capacity than CD while having the same dimensions.

Depending upon the disc type, DVD can store several Gigabytes of data (4.7 GB - 17.0 GB). DVDs are primarily used to store music or 6 movies can be played back on your television or computer too. They are not re-writable media.

DVDs come in three varieties as follows:

(i) DVD-ROM (Digital Video Disc-Read Only Memory)

(ii) DVD-R (Digital Video Disc- Recordable)

(iii) DVD-RW(Digital Video Disc-Rewritable)

(f) Blue-ray Disc (BD) : It is an optical disc storage medium designed to recapture the data normally in DVD format. BD contains 25 GB per layer space. The name Blue-ray disc refers to the blue laser used to read the disc which allows information to be stored at a greater density than the longer wavelength red laser used in DVDs.

Blue-ray can hold almost 5 times more data than a single layer DVD.

The variation in the formats are as fallows :

(i) BD-ROM (Read Only)

(ii) BD-R (Recordable)

(iii) BD-RW (Re-writable)

(iv) BD-RE (Re-writable)

(g) Pen/Thumb Drive : Pen Drive is also known as flash drive. A flash drive is a data storage device that consists of flash memory (key memory) with a portable USB (Universal Serial Bus) interface.

USB flash drives are typically removable, re-writable and much smaller than a floppy disk.

Today, flash drives are available in various storage capacities as 256 MB, 512 MB, 1 GB, 4 GB, 16 GB upto 64 GB. They are widely used as an easy and a small medium to transfer and store the information from the computer.

(h) Memory Cards : These are the data storage devices in a chip shaped, which can store the data in it. They are commonly used in many electronic devices including digital cameras, mobile phones, laptop computers etc. They are small, re - recordable, easily portable and light weighted.

Memory Measurement

1 Bit - Binary digit (either 0 or 1)

4 Bits -1 Nibble

8 Bits - 1 Byte - 2 Nibble

1024 Bytes - 1 KB (Kilobyte)

1024 KB - 1 MB (Megabyte)

1024 MB - 1 GB (Gigabyte)

1024 GB - 1 TB (Terabyte)

1024 TB - 1 PB (Petabyte)

1024 PB - 1 EB (Exabyte)

1024 EB - 1 ZB (Zettabyte)

1024 ZB - 1 YB (Yottabyte)

1024 YB - 1 Bronto byte

1024 Brontobyte - 1 Geopbyte

➢ **DATA DELETION AND RECOVERY**

- One of the biggest threats associated with digital data is its deletion. The storage devices can malfunction or crash down resulting in the deletion of the stored data.
- Deletion digitally stored data means changing the details of data at bit level, which can be very time consuming. Therefore, when any data is simply deleted, its address entry is marked as free and that much space is shown as empty to the user, without actually deleting the data.
- In case data gets deleted accidentally or corrupted, there arises a need to recover the data. Recovery of the data is possible only if the contents/memory space marked as deleted have not been over written by some other data.
- Data recovery is a process of retrieving deleted corrupted and lost data from secondary storage devices.

There are usually two security concerns associated with data :

(i) One is its deletion by some unauthorized person or software.

(ii) The other concern is related to unwanted recovery of data by unauthorized person/software.

TOPIC-2
Software

Revision Notes

➢ Set of instructions are referred to as software. It is that component of a computer system, which we cannot touch or view physically.

➢ Software comprises of the instructions and data to be processed using the computer hardware. The computer software and hardware together to the given task.

➢ In other words, each software is written for some computational purpose.

➢ A document or image stored on the hard disk or pen drive is referred to as a soft copy. Once printed, the document or an image is called a hard copy.

➢ Some examples of software include operating systems like Ubuntu or Windows 7/8, word processing tools like Libre Office Writer or Microsoft Word, Video player like VLC player, photo editors like Paint and Libre-Office Draw.

➢ **PURPOSE OF SOFTWARE**

- The purpose of software is to make computer hardware useful and operational.
- A software knows how to make different hardware components of a computer work and communicate with each other as well as with the end user. We cannot talk to or instruct the hardware of a computer directly. Hence, software acts as an interface between human users and the hardware.

➢ **TYPES OF COMPUTER SOFTWARE**

There are two types of software :

1. System software
2. Application software

1. SYSTEM SOFTWARE

- The software that provides the basic functionality to operate a computer by interacting directly with its constituent hardware is termed as system software. A system software knows how to operate and use different hardware components of a computer.
- It provides services directly to the end user or some other software.

Examples of system software includes.

(i) **Operating system :** It is the most basic system software, without which other software cannot work. The operating system manages other application programs and provides access and security to the users of the system. Some of the popular operating systems are Windows, Linux, Macintosh, Ubuntu, Android, iOS etc.

(ii) **Language Processor :** It is a system software which processes the program to make it understandable by the computer system, as computer system can only understand machine language or binary language, (also known as low level language). Assembler, interpreter and compiler language processors used to translate low level language into high level language and vice-versa.

(iii) **Device Drivers :** The purpose of a device is to ensure proper functioning of a particular device. The device driver acts as an interface between the device and the operating system. It provides required services by hiding the details of operations performed at the hardware level of the device. Just like a language translator, a device driver acts as a mediator between the operating system and the attached device.

2. APPLICATION SOFTWARE :

The system software provides the core functionality of the computer system. However, different users need the computer system for different purposes depending upon their requirements.

Hence, a new category of software is needed to cater to different requirements of the end users. This specific software that works on top of the system software is termed as application software.

There are two categories of application software :

(i) **General Purpose Software :** This software developed for generic applications, do cater to a bigger audience in general are called general purpose software. Such ready-made application software can be used by end users as per their requirements.

For example, spreadsheet tool Libre Office Cal can be used by any computer user to do calculation or to create an account sheet.

Adobe Photoshop, Mozilla web browser etc. fall under the category of general purpose software.

(ii) **Specific Purpose Software :** These are custom or tailor made application software, that are developed to meet the requirements of a specific organisation or an individual. They are better suited to the needs of an individual or an organisation. Considering that they are designed as per special requirements.

For example, school management software, accounting software etc.

➢ **FREE AND OPEN SOURCE SOFTWARE (FOSS)**

Developers of some software allow public to freely use their software along with source code with an aim to improve further with each other's help. Such software is known as Free and Open Source Software.

For example, Python, Libre-Office, Open-Office, Mozilla Firefox etc.

Software are freely available for use but source code may not be available. Such software is called **freeware**. For example, Skype, Adobe Reader etc.

When software to be used has to be purchased from the vendor who has the copyright of the software, then it is a proprietary software.

For example, Microsoft Windows, tally, QuickHeal etc.

❑❑

UNIT-II
Introduction to Python

CHAPTER-2
BRIEF OVERVIEW OF PYTHON

TOPIC-1
Getting Started with Python

Revision Notes

- Python is an interpreted and high-level language. Python programming language was developed by Guido Van Rossum in February 1991. It is a general purpose language.

- Python is based on two programming languages as:

  (i) ABC language

  (ii) Modula 3
- It supports multiple programming languages or paradigms including functional programming, procedural programming and object-oriented programming. Python is also dynamically typed and garbage collected.
- Python can run on many operating systems such as Windows, Mac OS, Unix etc. It is open source and freely distributed.

➢ **ADVANTAGES OF PYTHON**

- Python is easy to use and easy to read as compared to other languages like C/C++, Java etc. It is dynamically-typed and mandates indentation.
- Python is freely available and open source language that means its source code is available for public. You can download it directly from the Internet, use it, modify its source code and redistribute to others.
- It is a high level language so you do not need to know the architecture of the system and manage the memory.
- It has a standard library which contains the definitions of predefined functions, packages and modules.
- Python is platform independent that means it can run on any machine or server without any changes in its code. This makes Python a portable language.
- Python can be extended to other languages. You can write some of your code in languages such as C++ or C, this comes in handy especially in projects.
- It can support both-object oriented programming and procedure-oriented programming. Features of object oriented programming, like classes, objects & reusability help us to model the code easily.

➢ **DISADVANTAGES OF PYTHON**

- Python is interpreted language that means it reads the code line by line which makes the speed of it very slow.
- It serves a good server side language but it is rarely seen on the client side. It is very useful for system but seen as a weak language for mobile computing.
- Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprise.
- It is a dynamically typed language that means you do not need to declare variable before writing the code. This feature of python gives the run time error.

➢ **PYTHON DOWNLOAD AND INSTALLATION INSTRUCTIONS**

Python needs 25 Mb disk space to download. Before proceeding writing the code , you need to download and install Python.

➢ **DOWNLOADING**

- Write python.org in the URL and press Enter.
- A page will open. Click on Python Downloads.
- Now, click on download Python 3.x button. Here x is the name of version
- The file named Python-3.x.exe should start downloading into your default downloads folder.

➢ **INSTALLATION**

- After downloading the setup of Python, double-click on the setup. An **Open File-Security Warning** pop-up window will appear on screen.
- Click on **Run**. A **Python 3.x(32-bit) Setup** pop-up window will appear.
- Highlight the **Install Now** message and then click on it.
- Click the **Yes** button. A new **Python 3.x(32-bit) Setup** pop-up window will appear with a **Setup Progress** message and a progress bar
- Click the **Close** button.

➢ **WORKING IN PYTHON**

You can work in Python with two ways as follows:

(i) Interactive mode **(ii)** Script mode

(i) Interactive mode:

It is a quick way for running the Python code. It executes the code in Python shell. To access the interactive mode in Python follow the steps as:

Start button → All Programs → Python → IDLE (Python)

A window will appear with >>> sign.

You can write the code and press Enter key to execute that code.

```
>>> print ("Program")
Program
>>> a = 10
>>> b = 5
>>> c = a + b
>>> print ("The value of c :", c)
The value of c : 15
>>> 10 * 5
50
>>> 10 – 5 + 2
7
>>> print (10*2)
20
```

Advantages of Interactive mode

- Helpful when your script is extremely short and you want immediate results.
- Good for novice who need to understand Python basics.

Disadvantages of Interactive mode

- It is very tedious to run long a piece of code.
- Editing the code in interactive mode is hard as you have to move back to the previous commands.

(ii) Script Mode:

If you want to run long python code then interactive mode is not useful for that. Script mode is the way to go in such cases. In script mode, you need to write code then save it with a .py extension.

After writing the code, you can run it by clicking Run button then Run module or simply Press F5.

Example :

```
a = 10
b = 5
c = a + b + 2
print ("calculation")
print("The value of c : ", c)
print("Hello world")
print ("Python Program")
```

Output

```
calculation
The value of c : 17
Hello world
Python Program
```

➢ **Advantages of Script Mode**

- It is easy to run large piece of code.
- Editing your script is easier in script mode.

➢ **Disadvantages of Script Mode**

- Can be tedious when you need to run only a single or a few lines of code.
- You must create and save a file before executing your code.

TOPIC-2
Python Fundamental

Revision Notes

➢ ● A program is a set of instructions that governs the processing in any programming language.
- IPO is Input, Process and Output. In any language, you need to give input that processes and produces an appropriate output.
- To start the Python programming, you must know about basics of Python which are described in this chapter.

➢ **Character Set :** Every programming language contains some characters that can be recognised are called character set. These characters can be letter, digit or special symbol. Python includes some character sets such as:

- Letters A – Z, *a* – *z*
- Digits 0 – 9
- Special characters space + - * / ** \ ( )[ ] { } // = | = < > ' " , ; : % ! $ # @ underscore (_)
- White space tab (→), blank space, carriage return (↵), form feed, line terminator.
- Other characters like ASCII and unicode characters can be processed by Python

➢ **TOKEN**

It is the smallest unit in a program. It is meaningful to the compiler or interpreter. Tokens can be represented by a sequence of characters which acts as a logical unit

Python includes following tokens :

(i) Keywords (ii) Identifiers (iii) Literals or constants

(iv) Operators (v) Punctuators

(i) Keywords : Keywords are also known as reserved words that have some special meaning. These reserved words may not be used as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only except False, None and True.

The following list shows the reserved words in Python:

and	finally	pass
as	for	print
assert	from	raise
break	global	return
class	if	try
continue	import	while
def	in	with
del	is	yield
elif	lambda	False
else	not	None
except	or	True

(ii) Identifiers : An identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or *a* to *z* or an underscore (_) followed by zero or more letters, underscore and digits (0 to 9).

- Python does not allow punctuation characters such as @, $ and % within identifiers. Python is a case sensitive programming language. Thus, Value and value are two different identifiers in Python.

  Here are following identifiers naming convention for Python :

- Starting an identifier with a single leading underscore indicates by convention that the identifier is meant to be private.
- Starting an identifier with two leading underscores indicate a strongly private identifier.
- If the identifier also ends with two trailing underscores, the identifier is a language defined special name.
- Class name starts with an uppercase letter and all other identifiers with a lowercase letter.

Valid identifiers are :

SUM, For, _num, Num-2

Invalid identifiers are :

3 Num, Sum $, while

(iii) Literals : Literals or constants are those data items whose values can not be changed during program execution. They have fixed value.

Python allows several kinds of literals as :

- **String literals :** The literals which are enclosed with single quote or double quotes called string literals.

  Some valid string literals are :

  'A' "Rahul" "534" '543.75'

  Some invalid string literals are

  'A" Meenu 'Hi, "How are you ?",
- **Numeric literals :** These literals are all numbers. Python has following numeric literals.
  - **int (signed integers) :** These are whole numbers without any fractional part in it. *e.g.* 25, 43, –45, 02, etc.
  - **long (long integers) :** These are long integers with unlimited size followed by uppercase or lowercase L or l. *e.g.* 43474L, 5345l, etc.
  - **float :** These are real numbers with decimal point or fractional part *e.g.* 4347.32, –43.43, etc.
  - **complex :** These are in form of $a + bj$ where a is the real number and b is the imaginary number. *e.g.* $3 + 2j$, $5 – 4j$, etc.
- **Boolean Literals:** In Python, Boolean literals are represented by true and false.
- **Special Literal None :** Python has a special literal known as None which means nothing.

  e.g.

```
special-lit = None
print (special-lit)
```

  Output

```
None
```

- **Literal collections :** In Python, you have literal collections like list, tuple, dictionary and set.

  list - [1, 2, 3, 4]

  tuple - (1, 2, 3, 4)

  dictionary - {1 : 'One', 2 : 'Two', 3 : 'Three', 4 : 'Four'}

  set - { 'One', 'Two', 'Three', 'Four'}

(iv) Operators: Operators are special symbols which perform some computation. Operators and operands form an expression. Python operators can be classified as given below :

- **Unary Operators :** These operators required only one operand to operate upon. Some unary operators that are used in Python as

  \+ Unary Plus

  – Unary Minus

  ~ Bitwise complement

  not logical not
- **Binary Operators :** These operators require two operands to operate upon. In Python, below binary operators are used.
  - **Arithmetic Operators**

+ Addition
– Substraction
* Multiplication
/ Division
% Module
// Floor division
** exponent

- **Relational operators**
  < less than
  > Greater than
  <= less than or equal to
  >= Greater than or equal to
  == equal to
  != Not equal to
- **Logical operators**
  and–Logical AND
  or–Logical OR
- **Membership Operators**
  in–variable in sequence
  not in–variable not in sequence
- **Identity operators**
  is–the identity same
  is not–the identity not same
- **Bitwise Operators**
  & Bitwise AND
  ^ Bitwise XOR
  | Bitwise OR
  >> shift right
  << shift left
- **Assignment Operators**
  = Assignment
  /= Assign quotient
  += Assign sum
  *= Assign product
  -= Assign difference
  %= Assign remainder
  **= Assign Exponent
  //= Assign floor division

(v) Punctuators : It is a token that has syntactic and semantic meaning to the compiler, but the exact significance depends on the context.

- A punctuator can be a token that is used in the syntax of the preprocessor.
  Python has following punctuators:
  ' " # \ ( ) [] { } @ , : ; . =

STRUCTURE OF PYTHON PROGRAM

Before to start the programming in Python, you must know about the structure of Python program. Here is the simple code of python

```
# This is simple Python program
def sample( ) :                    <-- function
    print (''Here is the structure of Python program'')
# Main program
x = 15
y = x – 2 * 4                      <-- Expression
z = x + y – 10
print (''The value of x :'', x)
if (x > y) :        # if condition <-- Comment
    print (''Value of x is greater than that of y'')
else :
    print (''Value of x is less than that of y'')
sample( )
```

comments, Statements, Expression, Indentation

Various components of Python program are

(i) Comments : These are additional information which makes programming code to understand the logic and process of code. Comments are not displayed in the output and ignored by compiler or interpreter.

- In Python, comments are represent by # (hash sign.)
- There are two types of comments : full line comments and in line comments.
- The lines that start with # sign represent full line comments. In above code, two full line comments are presented as:

  # This is simple Python program

  # Main program
- The comments that start with mid of line represent inline comment. In above code, one in line comment is presented as:

  # if condition

(ii) Expression : It is a combination of variable, constant and operators.

An expression provides something that is evaluated by Python and represent appropriate result.

In above code, four expression are there as

$x = 15$

$y = x - 2 * 4$

$z = x + z - 10$

if $(x > y)$:

(iii) Statements : Simple statements are comprised within a single logical line.

Instructions that a Python interpreter can execute are called statements.

e.g. $a = 2$ is an assignment statement.

In above code, three statements are there as

$x = 15$

$y = x - 2 * 4$

$z = x + y - 10$

(iv) Function : It is a code or block of codes that can be reused in the program. Function is an important feature of any programming language that helps to reduce time for writing the code again.

In above code, one function is present as:

def sample ( ):

This function will be called at the end of the code by following statement

sample ()

(v) Indentation

- Blocks of code are denoted by line indentation, which is rigidly enforced.
- A code block which represents the body of a function or a loop begins with the indentation and end with the first unindented line.

➢ **VARIABLE AND ASSIGNMENT**

- Variable is a memory location that is used to store the value of any element. The value of variable can be changed during the program execution.
- When you create any variable that means you reserve some space in memory for that variable. Different types of value can be assigned to the variable, such as integers, strings, decimal, etc.
- Python variables do not need explicit declaration to reserve memory space. The variable's declaration is automatically done when you assign a value to the variable.
- To assign a value to the variable, equal to (=) sign is used.

e.g.

Name = "Riya"

Marks = 95.5

RollNo. = 25

Here Name, Marks and RollNo. are variable that hold value Riya, 95.5 and 25 respectively.

Multiple Assignment : Python has feature of multiple assignment. You can use different ways to implement multiple assignment in Python. These are:

- **Assigning Same Values to Multiple Variables :** In Python, single statement is used to assign same value to the multiple variables

  e.g. $a = b = c = 25$

  Here, *a, b* and *c* are variables which hold same value is 25.

- **Assigning Multiple Values to Multiple Variables :** In Python, you can assign multiple values to multiple variables in single statement.

  e.g. *a, b, c* = 8, 25, "Aayan"

  Here, *a, b* and *c* are variables which hold the value 8, 25 and "Aayan" respectively.

  This is also useful when you want to print multiple values in single line.

  e.g. *a, b, c* = 10, 20, "Computer"
  print (*a, b, c*)

  Output
  1020Computer

- **Variable Definition :** In Python, you do not need to declare variable before its use. Variable is automatically created when you assign a value to it.

  If you use some variables without assigning values to them, then it will give error as variable is not defined

  e.g. print (*a*)

  Output
  NameError : name a is not defined

  a = 25
  print (*a*)

  Output
  25

➢ **L-VALUE and R-VALUE**

Python uses the concept of L-value and R-value that is derived from the typical mode of evaluation on the left and right side of an assignment statement.

L-Value : This concept refers to the requirement that the operand on the left side of the assignment operator is modifiable, usually a variable.

R-Value : This concept pulls or fetches the value of the expression or operand on the right side of the assignment operator

e.g. name = 'Riya'

The value 'Riya' is pulled or fetched (R-value) and stored into the variable named name (L-value).

➢ **DYNAMIC TYPING**

- Python is a dynamic-typed language. In Python, while the value that a variable points to has a type, the variable itself has no strict type in its definition.
- It does not know about the type of the variable until the code is run.
- In Python, there is no need to declare data type with variable name but if you want to know the type of the value of variable, type ( ) method is used.

  Syntax type (variable_name)

  e.g.
  >>> *x* =25
  >>> print (*x*)
  25
  >>> type (*x*)
  <class 'int' >
  >>> name = 'Riya'
  >>> print (name)

```
Riya
>>> type (name)
<class 'str'>
>>> a = 25.5
>>> type (a)
<class 'float'>
```

➢ **MUTABLE AND IMMUTABLE TYPES**

- Mutable objects are those that allow and support changes in their contents. The mutable type includes list and dictionaries. It means that no new value object is created rather changes are made in the same value object.
- Immutable objects are those that can never change their value. In Python, integers, floating point numbers, Booleans, strings and tuples are immutable.

➢ **DELIMITER**

- A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data streams.

  Delimiters represent one of various means to specify boundaries in a data stream.

  Following tokens serve as delimiters in the Python grammar.

  ( ) [ ] { }, : . ' = ; += – = * = /= % **= &= |= ^ = >>= <<=

➢ **DATA TYPES**

Python data types are used to define the types of a variable. Any values that are represented in quotes (double or single) are string data type. Number without decimal point is integer data type.

Python has 5 data types as

(i) Numbers **(ii)** String **(iii)** List **(iv)** Tuple **(v)** Dictionary

(i) Numbers : This data type is used to hold numeric value. Like other languages C++, Java etc, there is no need to declare data types with variable. You can simply assign values to the variables. But if you want to know the type of variable, type ( ) method is used.

Number data type includes the following :

- **int :** Considers signed integers without fractional part. It can be positive or negative. *e.g.* 245, –75, etc.
- **long :** Considers long integers of non-limited size.

  e.g. 437 L, 53l. This data type exists in Python 2.x, deprecated in Python 3.x.
- **float :** Considers floating precision numbers, *e.g.* 453.3, 43.0, 34.34 etc.
- **Complex :** Considers complex numbers which are in the form of $a + bj$. Here a is the real number and b is the imaginary number.

```
>>> a = complex (4, 5)
>>> a
4 + 5j
>>> a. real
4
>>> a. imag
5
```

(ii) String data type : String is a sequence of characters. In Python, strings are represented by double quotes or single quotes. *e.g.* "Rahul", 'age', '34.50' etc.

```
>>> print ("Hello word")
Hello word
>>> name = "Rihaan"
>>> print (name)
Rihaan
```

```
>>> type (name)
<class 'str' >
```

A string in Python can contain as many characters as you want. The only limit is it depends on computer's memory resources.

(iii) List data type : List is a container of elements that can contain different types of values as string, float, integer, etc in it. Lists are mutable which means they can be changed after creation. The elements represent the list are written in square brackets [ ] and separated by comma (,)

e.g.

```
>>> list1 = [25, 'Hello', 75.3, 26]
>>> print (list1)
[25, 'Hello', 75.3, 26]
```

(iv) Tuple data type : It is also a container of objects that can be of any type as string, float, integer, etc. It is similar as list but it is immutable that means they can not be changed after creation. Elements in tuple is written using parentheses and separated by comma.

e.g.

```
>>> tuple1 = (25, 'Hello', 75.3, 26)
>>> print (tuple1)
(25, 'Hello', 75.3, 26)
```

(v) Dictionary data type : It is an unordered sequence of data that stores a key and value pair. Dictionaries are written within curly braces separated by comma. Key and value are separated by colon (:). Dictionaries are mutable that means they can be changed after creation.

e.g.

```
>>> Teacher = {"Name" : "Riya" , "Subject" : "Math"}
>>> print (Teacher)
{"Name" : "Riya" , "Subject" : "Math"}
```

➢ INPUT AND OUTPUT IN PYTHON

To give the input and get the output, we use some input/output functions. Python includes various functions to input/output as follows.

- **raw_input () :** This function is used to give input through keyboard by user. raw_input () function is used in Python 2.x version but later version of Python like 3.x does not support this function.

Syntax : Variable_name = raw_input (Message to be displayed for inputting)

e.g.

```
>>> subject = raw_input ("Enter the subject's name :")
Enter the subject's name : Computer
>>> Marks = raw_input ("Enter the marks :")
Enter the marks : 95
```

When you want to display the subject and marks on screen you should use following command

```
>>> subject
'Computer'
>>> Marks
'95'
```

You will notice that raw_input() function considers integer as string and gave the value in single quotes. That means, this function considers all values as string.

So, if you add or subtract some value in Marks, it will give error.

```
>>> Mark-5
Traceback (most recent call last) :
File "<pyshell#8>", line 1, in < module>
Marks-5
TypeError : cannot concatenate 'str' and 'int' objects
```

For this type of condition, Python provides two addition features int and float which are used with raw_input function

To input the integer value, int is used with raw_input()

e.g.

```
>>> marks = int (raw_input("Enter marks :"))
Enter marks : 95
>>> marks
95
>>> marks-5
90
```

To input the float value, float is used with raw_input

e.g.

```
>>> percentage = float (raw_input("Enter percentage :"))
Enter percentage  : 75.5
>>> percentage
75.5
>>> percentage + 10
85.5
```

- **input () :** This function is used to provide input entered by user, there is no need to give int and float. Input ( ) function automatically considers the entered value as string, integer or float.

  Syntax : variable_name = input ("Message to be displayed")

  e.g.

```
>>> marks = input ("Enter marks :")
Enter marks : 95
>>> marks -10
85
```

  This function is not used in Python version 2.x.

- **print ( ) :** This function is used to display output on the screen.

  Syntax : print ("Expression")

  e.g.

```
>>> print ("Hello world")
Hello world
>>> sal = input ("Enter Salary :")
Enter Salary : 20000
>>> print ("Salary :", sal)
Salary : 20000
>>> x = 2
>>> print (x + 2 *6)
14
>>> print (4 + 3*2 – 4)
6
```

If you assign multiple values to multiple variables and print their values separately.

```
x,y, z = 10, 20, "Hello"
print ("x =", x)
print ("y = ", y)
print ("z =", z)
```

Output

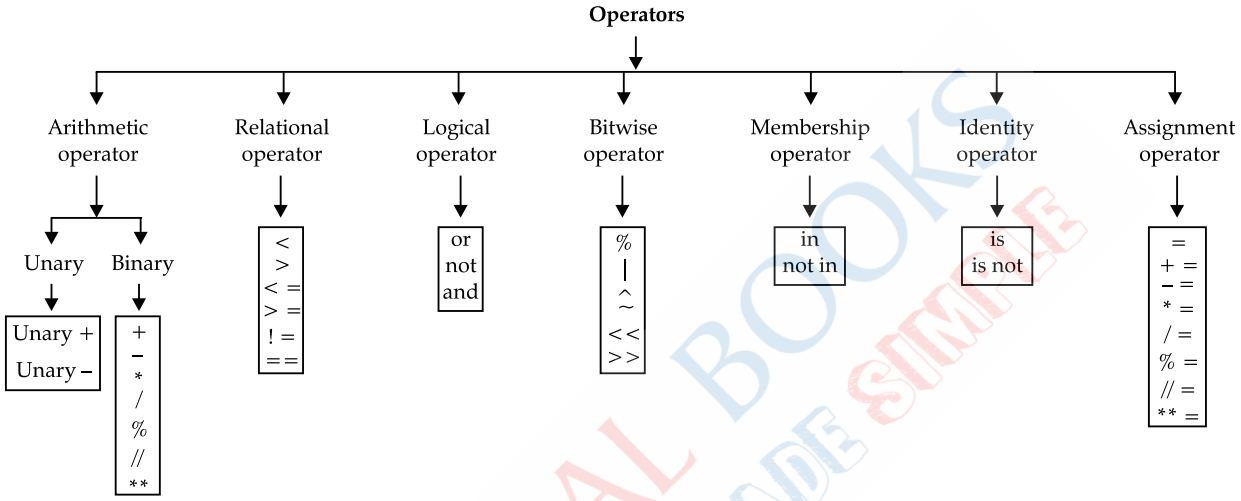
```
x = 10
y = 20
z = 'Hello'
```

TOPIC-3
Operators and Expression in Python

Revision Notes

➢ **OPERATORS :**

- Operators are special symbols which perform some computation. Operators and operands form an expression. Python operators can be classified as given below :



1. Arithmetic Operator

In Python, arithmetic operators are used to do arithmetic operations. Arithmetic operators are of two types as

(i) Unary Operators

Those operators work on only one operand are known as unary operators. In Python, these are further classified into two types

- **Unary + :** The result of an operation on a numeric type is the value of the operand itself. This operator has been predefined for all numeric types.

  e.g. if $x = 7$ then $+x = 7$

  if $x = -7$ then $+x = -7$

- **Unary – :** This operator can be used to negate numbers of the integer, floating – point and decimal type.

  e.g. if $x = 7$ then $-x = -7$

  if $x = -7$ then $-x = 7$

(ii) Binary Operators

The operators work on two operands are known as binary operators. In Python, these are further classified into following.

- **Addition (+) operator :** This operator is used to add the value of two or more operands.

  e.g.
  ```
  >>> x = 4
  >>> y = 6
  >>> print (x + y)
  10
  ```

- **Subtraction (–) operator :** This operator is used to subtract the value of second operand from first operand.

  e.g.
  ```
  >>> x = 10
  >>> y = 6
  ```

```
>>> print (x – y)
4
```

- **Multiplication(*) operator :** This operator is used to find the product of values of two or more operands.

e.g.
```
>>> a = 6
>>> b = 3
>>> c = a * b
>>> print (c)
18
```

- **Division (/) operator :** This operator is used to divide first operand by second operand.

e.g.
```
>>>16/4
4
>>>11/2
5.5
>>>36.6/6
6.1
```

- **Floor Division (//) operator :** This operator is also used to divide first operand by second operand but it shows only whole part of the result and fractional part is truncated.

e.g.
```
>>> 36.6//6
6.0
>>> 9//2
4
```

- **Modulus operator (%) :** This operator is used to find the modulus of two operands. It gives the remainder of dividing first operand by second operand.

e.g.
```
>>> 9 % 2
1
>>> 20 % 3
2
```

- **Exponentiation operator (**) :** This operator is used to find the result of a number raised to a power.

e.g.
```
>>> 2 ** 3
8
>>> 5**4
625
```

2. Relational Operators

These operators are used to determine the relation among given operands. It compares the value of operands and give the result either True or False.

Relational operators in Python are classified as :

Operators	Description	Example

< (less than)	It gives True if the value of left operand is less than the value of right operand.	>>> $x = 4$
		>>> $y = 5$
		>>> $x < y$
		True

> (greater than)	It gives True if the value of left operand is greater than the value of right operand.	>>> $x = 4$
		>>> $y = 4$
		>>> $x > y$
		False

<= (less than or equal to)	It gives True if the value of left operand is less than or equal to the value of right operand	>>> $x = 4$
		>>> $y = 4$
		>>> $x <= y$
		True

>= (greater than or equal to)	It returns True if the value of left operand is greater than or equal to that of right operand.	>>> $x = 5$
		>>> $y = 6$
		>>> $x >= y$
		False

== (Equal to)	It returns True if the value of left operand is equal to the value of right operand.	>>> $x = 6$
		>>> $y = 6$
		>>> $x == y$
		True

!= (Not equal to)	It returns True if the value of left operand is not equal to the value of right operand.	>>> $x = 6$
		>>> $y = 5$
		>>> $x != y$
		True

3. Logical Operators

In Python, logical operators are used for conditional statements that returns True or False. Logical operators are further classified as follows.

Operator	Description	Example
and	It returns True if both the operands left side and right side are true.	>>> (3 == 3) and (3 < 4) True
or	It returns True if either of the operand left side or right side is true.	>>> (3 == 4) or (4 > 3) True
not	It returns True if operand is false.	>>> not (5 > 7) True

4. Bitwise Operator

In Python, bitwise operators are used to perform bit operation. All the decimal value will be converted into binary values. Next, Python bitwise operators will work on these bits such as shifting them left to right, etc.

e.g. $x = 12$

$y = 10$

and their binary values

$x = 1100$

$y = 1010$

A table is given to understand bitwise operators more precisely.

Operator	Name of Operator	Description	Example
&	Bitwise AND	It copies 1 to the result if it exists in both operands otherwise copies 0.	$x \& y = 8$ which means 1000
|	Bitwise OR	It copies 1 to the result if it exists in either operand otherwise copies 0.	$x \| y = 14$ which means 1110
^	Bitwise exclusive OR	both operands digits are different. Otherwise copies 0	$x \wedge y = 6$ which means 0110
~	Bitwise complement	This operator is unary and has the effect of flipping bits.	$\sim x = 3$ which means 0011
<<	Shift left	value of left operand is moved left by the number of bits specified by the right operand.	$x << 2 = 48$ which means 110000
>>	Shift Right	value of left operand is moved right by the number of bits specified by the right operand.	$x >> 2 = 3$ which means 0011

5. Membership Operators

These operators are used to validate the membership of a value. These are used in lists, tuples, strings, etc. Membership operators are of two types.

- **in Operator :** This operator is used to check whether a value exists in a sequence or not. If a value exists in a sequence it gives True otherwise False.

  e.g.
  ```
  >>> 4 in [1, 2, 3, 4, 5]
  True
  >>> 5 in [1, 2, 3, 4]
  False
  ```

- **not in Operator :** This operator is opposite of in operator. It gives True if a value does not exist in a sequence otherwise False.

  e.g.
  ```
  >>> 4 not  in [1, 2, 3, 4, 5]
  False
  >>> 3 not in [1, 2, 4, 5]
  True
  ```

6. Identity Operators

In Python, these operators are used to determine whether a value is of a certain class or type. Identity operators are two types in Python

- **is operator :** It returns True if the type of the value in the right operand points to the same type in the left operand.

  e.g.
  ```
  >>> a = 8
  >>> type (a) is int
  True
  >>> type (a) is float
  False
  ```

- **is not operator :** It returns True if the type of the value in the right operand point to a different type.

  e.g.
  ```
  >>> x = 4.5
  >>> type (x) is not float
  False
  >>> type (x) is not int
  True
  ```

➢ **ASSIGNMENT OPERATORS**

These operators are used to assign some values to the variable.

e.g. $a = b + 5$

Here value of $b + 5$ assigned to variable a.

Given below is the table that will describe the assignment operators :

Operator	Operation	Equivalent to	Description
=	$a = 4$	$a = 4$	It assigns value of right side operand to left side
+=	$a += b$	$a = a + b$	It adds values of both operands (left and right) and assign the result to the left operand.
-=	$a -= b$	$a = a - b$	It subtracts the value of right operand from left operand and assigns the result to the left operand.
*=	$a *= b$	$a = a * b$	It multiplies the values of both operands (left and right) and assigns the result to the left operand.
/=	$a /= b$	$a = a/b$	It divides the value of left operand by right operand and assigns the result to left operand.
%=	$a\% = b$	$a = a \% b$	It divides the value of left operand by right operand and assigns the remainder to left operand.

//=	a //= b	$a = a // b$	It divides the value of left operand by right operand and assigns the quotient to the left operand.
**=	a ** = b	$a = a$ ** b	It computes the a raised to power of b and assigns the result to left operand.

➢ **PRECEDENCE OF OPERATORS IN PYTHON**

Precedence of operators are shown in below table.

It is in descending order means higher precedence than the lower ones.

Operators	Meaning
()	Parentheses
**	Exponent
+, —, ~	Unary plus, unary minus, Bitwise not
*, /, // %	Multiplication, division, floor division, modulus
+, –	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=	Comparison Operators
=, +=, -=, *=, /=, //=, **=	Assignment Operators
is, is not	Identity Operators
in, not in	Membership Operators
not	logical NOT
and	logical AND
or	logical OR

Expression

An expression is a combination of variables, literals and operators. In Python, some atoms and operators form an expression. An atom is something that has some value.

In Python, an expression can be of any type ; arithmetic expression, string expression, relational expression, logical expression, compound expression, etc.

(i) Arithmetic Expression This expression includes variables, digits and arithmetic operators.

e.g.

$a + b * c$

$a // b + c + 2$

$a\% b * c - 3$

$a ** b + c//2 + d*4$

(ii) Relational Expression This expression includes variables, literals and some relational operators.

e.g.

$a >= b$

$a <= b + 2$

$a == b$

$a == b < c$

(iii) Logical Expression This expression includes variables, literals and logical operators.

e.g.

a and b

a or not b

not a and not b

(iv) String expression

In Python, there are two string operators that combined with integer and operands.

Concatenation Operator (+) is used to concatenate two or more strings.

e.g.

"Hello" + "World"

" 4 " + " 7 "

Replication Operator * is used to replicate the string.

e.g. "Hello" * 3

"7" *4

TOPIC-4
Conditional Statements and Iteration

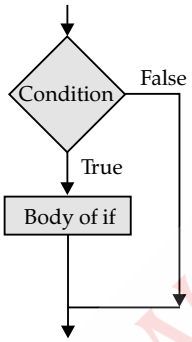
Revision Notes

- A conditional statement is a statement which is executed on the basis of result of a condition.
  - **if statement :** Checks a particular condition, if the condition is true, statements followed by if are executed.

    Syntax

    if (condition) :

    statement (s)



    e.g. $a = 5$

    if ($a == 5$) :

    print ("a is equal to 5")

    Output

    a is equal to 5

  - **if - else statement :** Checks the condition if the condition is true, statements followed by if are executed but if the condition is false, statement followed by else are executed.

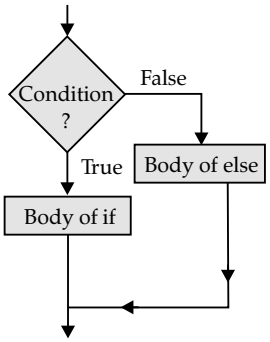
    Syntax

    if (condition) :

    statement (s)

    else :

    statement (s)



    e.g. $a = 5$

```
if (a = = 5) :
        print ("a is equal to 5" )
else :
        print ("a is not equal to 5" )
```

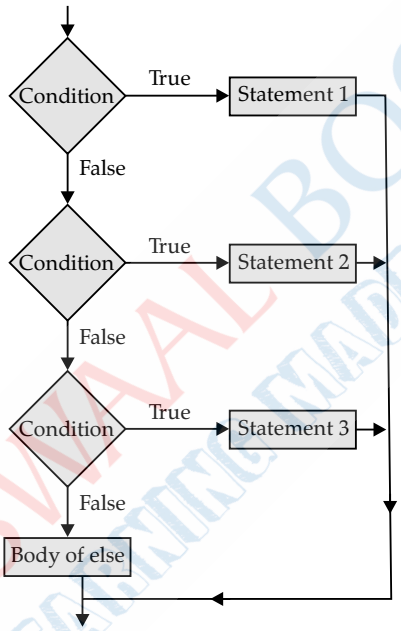
Output

a is equal to 5

- **if - elif statement :** allows you to check multiple expressions for true and execute a block of code as soon as one of the conditions evaluates to TRUE.

Syntax

```
if (condition ) :
        statement (s)
elif (condition ) :
        statement (s)
```



OR

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

e.g.

```
a = 10
b = 15
c = 20
if (b > a) :
        print ("b is greater than a" )
elif (b > c) :
        print ("b is greater than c" )
```

Output

b is greater than a

or

```
if (b > a) :
        print ("b is greater than a" )
elif (b > c) :
        print ("b is greater than c" )
else :
        print ("b is less than both a and c" )
```

Output

b is greater than a.

- **Nested if statement :** It means an if statement inside another if statement and else statement inside another if .... else statement

Syntax (form 1)

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

Form 2

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

Form 3

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

Form 4

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

➢ ITERATION CONSTRUCTS

- The iteration statements or repetition statements allow a set of instructions to be performed repeatedly.
- Python provides two types of loops

  (a) Counting loop repeat a certain number of times *e.g.* for.

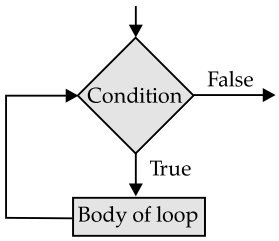
  (b) Conditional loop repeat until a certain thing happens *e.g.* while.
- In Python, we have different types of loops like for loop, while loop and nested for loop with their subtypes, syntax and examples.

➢ WHILE LOOP

A while loop in Python iterates till its condition becomes False. In other words, it executes the statement under itself while the condition it takes is true.

Syntax

```
while ( logical-Expression) :
        loop body
```



When the program control reaches the while loop, the condition is checked. If the condition is true, the block of code under it is executed. Remember to indent all statements under the loop equally. After that, the condition is checked again. This continues until the condition becomes false. Then the first statement, if any after the loop is executed.

e.g.

```
a = 4
while (a > 0) :
        print (a)
        a - = 1
```

Output

```
4
3
2
1
```

(a) An Infinite loop : Be careful while using a while loop. Because if you forget to increment the counter variable in Python or write flawed logic, the condition may never become false. In such a case, the loop will run infinitely and the statements after the loop will starve. To stop execution, press Ctrl + C. However, an infinite loop may actually be useful.

(b) The else statement for while loop : A while loop may have an else statement after it. When the condition becomes false, the block under the else statement is executed. However, it does not execute if you break out of the loop or if an exception is raised.

e.g.

```
a = 4
while (a > 0) :
        print (a)
        a - = 1
else :
    print ("Reached 0")
```

Output

```
4
3
2
1
Reached 0
```

In the following code, we put a break statement in the body of the while loop for a ==1. So, when that happens, the statement in the else block is not executed.

e.g.

```
a = 4
while (a > 0) :
```

```
        print (a)
        a - = 1
        if (a == 1):
          break
    else :
        print ("Reached 0")
```

Output

4
3
2

(c) Single Statement While : Like an if statement, if we have only one statement in while loop's body, we can write it all in one line.

e.g.

```
a = 4
while (a > 0) : print (a); a - = 1
```

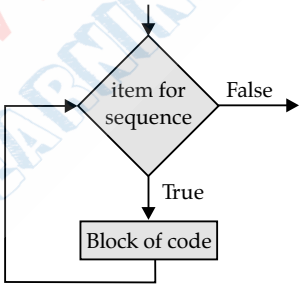
Output

4
3
2
1

You can see that there were two statements in while loop's body, but we used semicolons to separate them without the second statement, it would form an infinite loop.

➢ **FOR LOOP**

Python for loop can iterate over a sequence of items. The structure of a for loop in Python is different than that in C++ or Java. That is, for (int $i = 0$; $i < n$; i++) would not work here. In Python, we use the 'in' keyword.



e.g. for a in range (4):

```
    print(a)
```

Output

0
1
2
3

If we want to print 1 to 4, we could write the following code.

```
for a in range (4) :
    print (a + 1)
```

Output

1
2

3

4

➢ **range() FUNCTION**

This function generates the integer numbers between the given start integer and the stop integer, which is generally used to iterate over with for loop.

Syntax

range (start, stop, step)

e.g.

```
for i in range (2, 12, 2) :
        print (i)
```

Output

2

4

6

8

10

➢ **JUMP STATEMENTS**

Python offers two jump statements as break and continue; to be used within loops to jump out of loop iterations.

- **break statement :** A break statement terminates the loop it lies within. It skips the rest of the loop and jumps over to the statement following the loop.

e.g.

```
a = 4
while (a > 0) :
    if (a = = 1) :
        break
    print (a)
    a – = 1
```

Output

4

3

2

- **continue statement:**

Unlike break statement, the continue statement forces the next iteration of the loop to take place, skipping any code in between.

➢ **NESTED LOOP**

A loop may contain another loop in its body. This form of a loop is called nested loop. But in a nested loop, the inner loop must terminate before the outer loop.

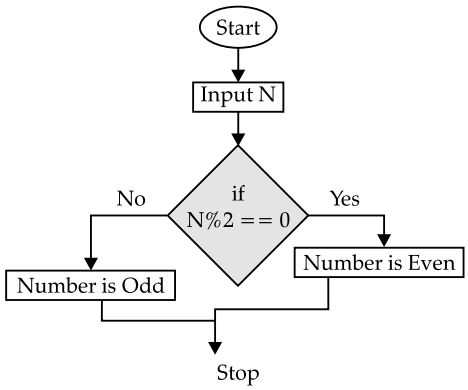
e.g.

```
for i in range (1, 6) :
    for i in range (1, i) :
        print ("*"),
    print
```

➢ **FLOW CHART**

It is simply a graphical representation of steps. It shows steps in a sequential order, and is widely used in presenting flow of algorithms, workflow or processes.

e.g.



TOPIC-5
Debugging

Revision Notes

- An error or a bug is anything in the code that prevents a program from compiling and running correctly.
- Due to errors , a program may not execute or may generate wrong output.

  There are various program errors, in which some of them are as fellows :

  1. **Compile time errors :** These errors are displayed by the interpreter and compiler. If compiler displays this error then program will not run. Before to run the program, it must fix all compile time errors. Compile time error further classified into two types as :

  (i) **Syntax Error :** Like any other languages, Python language has some rules that specify how a program is to be written. This is called syntax. Any statement will be interpret by interpreter if its syntax is correct.

  Some commonly syntax errors in Python are :

  - Missing parenthesis
  - Incorrect indentation
  - Wrong spelling of keywords
  - Missing colon.

  (ii) Semantic Errors : This type of error occurs when wrong or improper statements are used. Those statements are meaningless.

  2. Runtime Errors : While executing the program, runtime error occurs due to abnormal termination of program. It is occurred when statements are correct syntactically but interpreter cannot execute it correctly.

  Some commonly runtime errors in Python are :

  - Division by zero
  - Access the file which does not exist
  - Using identifier which has not been defined.

  3. **Logical Errors :** A logical error is a mistake in a program's source code that results in incorrect or unexpected behaviour. It may simply produce the wrong output or may cause a program to rash while running.

  Some commonly logical errors in Python are :

  - Wrong variable name
  - Wrong operator precedence
  - Mistake in a boolean expression.

- **Debugging**

  It is the process of detecting and removing of existing and potential errors in a program code that can cause it to behave unexpectedly or crash.

CHAPTER-3
LIST AND DICTIONARY

TOPIC-1
List

Revision Notes

- Lists are the container which store the heterogeneous elements of any type as integer, character, floating that store in square brackets [].
- Lists are mutable which means they can be modified after creation. You can easily add and remove element from a list.
- Lists are similar as tuples and strings but lists are mutable while strings and tuples are immutable.
- **CREATING A LIST**

  In Python, you can create different types of list. Some of them are described below:

  (i) Empty list : This type of list does not contain any element. This list is equivalent to 0 or ' '. Empty list can be created using

  ```
  >>> l = list ( )
  >>> print (l)
  [ ]
  ```

  (ii) Mixed data types list : This type of list contains different data types. Elements in mixed data types list are type of integer, string etc.

  e.g.
  ```
  l = [12, 'arc','xyz', 45, – 67]
  print (l)
  Output
  [12, 'arc','xyz', 45, – 67]
  ```

  (iii) Long lists : In Python, you can create long list which contain many elements that may be different or same.

  e.g. list1 = [25, 42, 43, 73, – 44, 'Neha', 'Rahul', '*abc*', 2.5, 4.4, 79, 63, 44, '*xyz*', 'Hello']
- **CREATING LIST FROM EXISTING SEQUENCE**

  List can be created from an existing sequence such as tuples, strings, etc. Elements of list will be represented individually separated by comma.

  Syntax listName = list (sequence)

  e.g. 1.
  ```
  >>> l = list ('PROGRAM')
  >>> l
  ['P', 'R', 'O', 'G', 'R', 'A', 'M']
  ```

  e.g. 2.
  ```
  >>> tuple1 = ('P', 'R', 'O', 'G', 'R', 'A', 'M')
  >>> l1 = list (tuple1)
  >>> l1
  ['P', 'R', 'O', 'G', 'R', 'A', 'M']
  ```

  Python also allows to create the list of digit or string which are entered through keyboard.

  e.g.
  ```
  >>> list1 = list (input ("Enter list elements:"))
  Enter list elements : 7433456
  >>> list1
  [ 7, 4, 3, 3, 4, 5, 6]
  ```

➢ ACCESSING LISTS

- Like strings, lists are accessed by index number,
- With the help of index number, you can access the individual elements from the list. Two types of indexing are used to access the elements : positive indexing and negative indexing.
- Index number 0 represents the first element of the list, 1 represents the second element and so on. It is called positive indexing.
- Index number –1 represents the last element of the list, –2 represents the second last element and so on. It is called negative indexing.

	0	1	2	3	4	5	6	7	8
l =	45	23	–44	'abc'	32	'*xyz*'	'the'	34	–25
	–9	–8	–7	–6	–5	–4	–3	–2	–1

```
>>> l [2]
– 44
>>> l [–3]
'the'
>>> l [2]+ l [–2]
–10
>>> l [–5]
32
>>> l [–6]
'abc'
>>> l [1] * 3
69
```

If you give decimal number or string as an index number then it will give error.

```
>>> l [2.5]
Trackback (most recent call last) :
    l [2.5]
Type Error : list indices must be integers, not float
>>> l ['g']
Trackback (most recent call last ) :
    File < pyshell # 1>, line 4, in < module >
        l ['g']
Type Error : list indices must be integers, not str
```

➢ TRAVERSING A LIST

Traversing a list is a process to iterate the element of the list. With the help of traversing you can iterate all or particular elements of a list.

There are two ways to traverse a list as follows :

- **Using for loop:** for loop allows to iterate or traverse all the elements of the list.

Syntax

```
for variable in listName :
    process each element
```

e.g.

```
list = ['W', 'O', 'R', 'L', 'D', '8', '3']
for i in list :
    print (i)
```

Output

```
W
O
```

R
L
D
8
3

OR

```
list = ['W', 'O','R','L','D', 8.3]
for i in list :
    print (i, end = "  ")
```

Output

W O R L D 8 3

- **Using for loop with range function :** If you want to traverse specified elements of list, you must use for loop with range function.

Syntax

```
for index in range (len (list_name)) :
    process List [index]
```

e.g.

```
l = ['W', 'O', 'R', 'L', 'D', 8, 3]
for i in range (len (l)) :
    print (l [i])
```

Output

W
O
R
L
D
8
3

➢ LIST OPERATIONS

In Python, some basic operations can be performed on lists as follows :

- **Concatenate Lists**

Lists can be concatenated in Python with the help of concatenate operator (+).

e.g.

```
>>> l1 = [45, 11, 7]
>>> l2 = [4, 79, 3]
>>> l = l1 + l2
[45, 11, 7, 4, 79, 3]
```

Concatenate operator is used when two lists are to be joined. If you add one list to an integer or a string it will give error.

```
>>> l1 = [45, 11, 7]
>>> l = l1 + 8
Trackback (most recent call last ) :
    File "<pyshell # 32>", line 1, in <module>
    l = l1 + 8
Type Error : can only concatenate list (not 'int') to list
>>> l1 = [45, 11, 7]
>>> l = l1 + "Hello"
Trackback (most recent call last ) :
    File "<pyshell # 33>", line 1, in <module>
    l = l1 + "Hello"
Type Error : Can only concatenate list (not 'str') to list
```

- **Replicating lists :** Lists can replicated in Python with the help of replicate operator (*).

e.g.
```
>>> l1 = [45, 11, 7]
>>> l = l1 * 2
[45, 11, 7, 45, 11, 7]
```

Replicating operator is used when one list is to be replicated. If you use two lists as operands with replicate operator, it will give error.

```
>>> l1 = [45, 11, 7]
>>> l2 = [2, 3, 4]
>>> l = l1 * l2
Trackback (most recent call last ) :
    File "<pyshell # 44>", line 1, in <module>
    l = l1 * l2
Type Error : two lists can't replicate
```

- **Slicing :** It is the process to extract the sub part of the list. In Python, indexing is used for slicing the list.

Syntax listname [start : stop]

e.g.
```
>>> lis1 = [45, 50, 55, 'Neha', 'Riya', 20, 25, 'abc']
>>> list1 [2 : 4]
[55, 'Neha']
>>> list1 [0 : 4]
[45, 50, 55, 'Neha']
>>> list1 [4 :]
['Riya', 20, 25, 'abc']
>>> list1 [: 4]
[45, 50, 55, 'Neha']
>>> list1 [4] + list1 [–2]
['Riya' ', 25]
```

In Python, lists also support slice with steps.

Syntax listName [start : stop : step]

```
>>> list1 = [10, 15, 16, 20, 21, 28, 30, 35, 40, 44, 7, 17]
>>> list1 [1 : 10 : 2]
[15, 20, 28, 35, 44]
>>> list1 [0 : 13 : 3]
[10, 20, 30, 44]
>>> list1 [ : : 4]
[10, 21, 40]
```

➢ **LIST MODIFICATION USING SLICING**

Lists are mutable means they can be modified after creation. Using slicing, modification can be done in the list.

e.g.
```
>>> l = [15, 15, 34, 7, 8]
>>> l [1] = ['Riya']
>>> l
[15, 'Riya', 34, 7, 8]
>>> l [2 : 4] = [44, 79]
>>> l
[15, 'Riya', 44, 79, 8]
>>> l [0 : 3] = ["Hello"]
>>> l
["Hello", 79, 8]
```

➢ WORKING WITH LISTS

Various operations can be performed on lists as : appending, updating, deleting, etc.

- **Appending elements to a list :** In Python, you can add individual elements to an existing list. For this, append ( ) method is used.

  Syntax listName.append (element)

  e.g.
  ```
  >>> list = [45, 33, 12, 45]
  >>> list. append (66)
  >>> list
  [45, 33, 12, 45, 66]
  ```

- **Updating elements to a list :** Lists are mutable which can be modified. You can update or change an element in list using index number.

  Syntax list_name [index] = value

  e.g.
  ```
  >>> list = [45, 33, 12, 45]
  >>> list [1] = 49
  >>> list
  [45, 49, 12, 45]
  ```

- **Deleting elements from a list :** Elements can be deleted from a list. For this, del keyword is used in Python.

  Syntax del list_name [indexNumber]
  del list_name [start : stop]

  e.g.
  ```
  >>> list = [45, 33, 12, 45, 62, 73, 93]
  >>> del list [2]
  >>> list
  [45, 33, 45, 62, 73, 93]
  >>> del list [1 : 4]
  >>> list
  [45, 73, 93]
  ```

  Other method, pop ( ) is used to delete an individual element from a list. This method also returns deleted element.

  Syntax list_name.pop (index No)

  e.g.
  ```
  >>> list = [45, 33, 25, 63, 79, 29, 64]
  >>> list. pop (2)
  25
  >>> list
  [45, 33, 63, 79, 29, 64]
  ```

➢ LIST METHOD

There are many methods to manipulate a list in Python. Some of them are described below :

- **extend ( )** This method is used to add multiple elements to a list. It is different from append ( ) method. append ( ) method adds individual element while extend ( ) method adds multiple elements to a list.

  Syntax list1.extend (list2)

  e.g.
  ```
  >>> l1 = [45, 22, 13]
  >>> l2 = [79, 83]
  >>>l1.extend (l2)
  >>> l1
  [45, 22, 13, 79, 83]
  >>> l2
  [79, 83]
  ```

- **insert ( )** Like append ( ) and extend ( ) methods, insert ( ) method is used to add element to a list but at specified index number.

  Syntax listName.insert (indexNumber, element)

  e.g.
  ```
  >>> list1 = [45, 33, 45, 76, 82, 12]
  >>> list1.insert (3, 99)
  >>> list1
  [45, 33, 45, 99, 76, 82, 12]
  ```

- **remove ( )** This method is used to remove the element which is passed as an argument. pop ( ) method is used to remove the element whose index number is given as an argument.

  Syntax <listName>.remove (value)

  e.g.
  ```
  >>> list1 = [45, 33, 45, 76, 82, 12]
  >>> list1.remove (76)
  >>> list1
  [45, 33, 45, 82, 12]
  >>> list1.remove (90)
  Trackback (most recent call last ) :
          File "<pyshell # 44>", line 1, in <module>
                  list1.remove (90)
  Value Error : list.remove (x) : x not in list.
  ```

- **reverse ( )** This method is used to reverse the elements of a list.

  Syntax <listName>.reverse ( )

  e.g.
  ```
  >>> list1 = [45, 33, 45, 82, 12]
  >>> list1.reverse ( )
  >>> list1
  [12, 82, 45, 33, 45]
  ```

- **sort ( )** This method is used to arrange the elements in ascending order in a list.

  Syntax listname.sort ( )

  e.g.
  ```
  >>> list1 = [12, 82, 45, 33, 45]
  >>> list1 . sort ()
  >>> list1
  [12, 33, 45, 45, 82]
  ```

TOPIC-2
Dictionary

Revision Note

- Dictionary is mutable which means they can be changed after creation.
- Dictionary is a data type in Python which stores the data in key value pair. Each key value pair is separated by comma. While each key with its corresponding value is given by colon (:), key value pairs are shown in curly brackets in dictionaries. Keys of a dictionary are immutable.

  Syntax dictionaryName = {key 1 : value 1, key 2 : value 2, ...}

  e.g. dic 1 = {"Color" : "Blue", "Fruit" : "Mango", "Price" : 100}

  Below is the relationship between key value pair of above dictionary

Key - Value pair	Key	Value
"Color" : "Blue"	"Color"	"Blue"

"Fruit" : "Mango" "Price" : 100	"Fruit" "Price"	"Mango" 100

- Dictionaries are also called "associative arrays" or " mapping" or "hashes".
- A dictionary operation that takes a key and finds the corresponding value is called "lookup".
- **ACCESSING ELEMENTS OF A DICTIONARY**

In lists and strings, indexing is used to access the elements. While in dictionary, key is used to access the elements.

Syntax dictionaryName [Key]

e.g. >>> state = {"Uttar Pradesh" : "Lucknow", "Maharashtra" : "Mumbai", "Madhya Pradesh" : "Bhopal", "Rajasthan" : "Jaipur"}

>>> state ["Madhya Pradesh"]

Bhopal

>>> print ("The capital of Rajasthan is", state ["Rajasthan"])

The capital of Rajasthan is Jaipur

If you give the key that does not exist in the dictionary, it will show error.

>>> state ["Delhi"]

Trackback (most recent call last) :

File "<pyshell # 10>", line 1, in <module>

state KeyError : "Delhi"

["Delhi"]

- **TRAVERSING A DICTIONARY**

Traversing is a process to iterate or access each element of a dictionary.

for loop is used to iterate all the elements of a dictionary.

Syntax for variable in dictionaryName :

process each item here

e.g. student = {'RollNo' : 35, 'Name' : 'Vihaan', 'Subject' : 'Computer'}

for key in student :

print (key , ':', student [key])

Output

'Name' : 'Vihaan'

'RollNo' : 35

'Subject' : 'Computer'

- **ACCESSING KEYS OR VALUES SIMULTANEOUSLY**

You can access all keys or all values simultaneously in dictionary.

To access keys use this, dictionaryName.keys ( )

To access values use this, dictionaryName.values ( )

e.g. >>> student = {"Name" : "Vihaan", "RollNo" : 35, "Subject" : "Computer", "Address" : "Delhi"}

>>> student.keys ( )

['Name', 'RollNo', 'Subject', 'Address']

>>>student.values ()

['Vihaan', 35, 'Computer', 'Delhi']

- **CHARACTERISTICS OF DICTIONARY**

Some characteristics of dictionary are as follows :

- **Not in Sequence :** List, string and tuple all are in sequence but dictionary is not in a particular sequence. Dictionary is an unordered collection.

- **Keys must be unique :** Keys should not be duplicate in dictionary. They must be unique which used to identify the values of dictionary. Values can be duplicate but keys can't.
- **Mutable :** Dictionaries are mutable which means they can be modified after creation. To change value of respective key use following

  dictionaryName [Key] = Value
- **Indexed by keys :** Keys are used to access the elements of the dictionary. While lists, tuples and strings use index number to access the elements.

➢ **WORKING WITH DICTIONARIES**

Various operations are performed in dictionaries. Some of them are:

- **Creating a Dictionary :** Dictionary can be created using key value pair.

```
>>> Student = {'Name' : 'Vihaan', 'RollNo' : 35, 'Subject' : 'Computer'}
```

Key - Value can be added into an empty dictionary.

```
>>> Employee  = { }
>>> Employee
{}
>>> Employee ['Name'] = 'Shekhar'
>>> Employee ['Salary'] = 30000
>>> Employee ['Post'] = 'Accountant'
>>> Employee
{'Name' : 'Shekhar', 'Salary' : 30000, 'Post' : 'Accountant'}
```

- **Adding Elements to a Dictionary :** After creating a dictionary, element (s) can be added to it.

Syntax dictionaryName [Key] = value

e.g.

```
>>> Employee  = { 'Name' : 'Shekhar', 'Salary' : 30000, 'Post' : 'Accountant'}
>>> Employee ['Age'] =32
>>> Employee
{'Name' : 'Shekhar', 'Salary' : 30000, 'Post' : 'Accountant', 'Age' : 32}
```

- **Updating Existing Elements in a dictionary :** You can also update an existing element in a dictionary.

Syntax dictionaryName [Key] = value

e.g.

```
>>> Employee  = { 'Name' : 'Shekhar', 'Salary' : 30000, 'Post' : 'Accountant', 'Age' : 32}
>>> Employee ['Salary'] =35000
>>> Employee
{'Name' : 'Shekhar', 'Salary' : 35000, 'Age' : 32, 'Post' : 'Accountant'}
```

- **Checking for Existence of a Key :** Two operators are used to check for existence of a key. These are **in** and not **in** operators.

Syntax Key in dictionaryName

Key not in dictionaryName

e.g.

```
>>> Employee = {'Name' : 'Shekhar', 'Salary' : 30000, 'Post' : 'Accountant'}
>>> 'Age' in Employee
False
>>> 'Salary' in Employee
True
>>> 'Age' not in Employee
True
>>> 'Name' not in Employee
False
```

➢ DICTIONARY FUNCTIONS or METHODS

There are various built-in functions or methods in Python to manipulate the dictionaries. Some of them are as follows:

1. **len ( )** This method is used to return the length of the dictionary that is counts the number of elements or key - value pairs present in the dictionary.

Syntax len (dictionaryName)

e.g.
```
>>> state = {"Uttar Pradesh" : "Lucknow", "Madhya Pradesh" : "Bhopal", "Maharashtra" : "Mumbai", "Rajasthan" : "Jaipur", "Gujarat" : "Gandhi Nagar"}
>>> len (state)
5
>>> state ["Punjab"] = "Chandigarh"
>>> len (state)
6
```

2. **clear ( )** This method is used to remove all the elements from a dictionary

Syntax dictionaryName.clear ( )

e.g.
```
>>> state = {"Uttar Pradesh" : "Lucknow", "Madhya  Pradesh" : "Bhopal", "Maharashtra" : "Mumbai", "Rajasthan" : "Jaipur", "Gujarat" : "Gandhi Nagar"}
>>> state.clear ()
>>> state
{}
```

3. **get ()** This method is used to return the value of respective key. If key is not present in the dictionary, it will give error message.

Syntax dictionaryName.get (key, [default])

e.g.
```
>>> state = {"Uttar Pradesh" : "Lucknow", "Madhya Pradesh" : "Bhopal", "Maharashtra" : "Mumbai", "Rajasthan" : "Jaipur", "Gujarat" : "Gandhi Nagar"}
>>> state. get ("Maharashtra")
'Mumbai'
>>> state.get ("Punjab")
'Key not found'
```

4. **items ()** This method is used to return all the elements *i.e.* all key value pairs. But, these key value pairs are not in order.

Syntax dictionaryName.item ( )

e.g.
```
>>> state = {"Uttar Pradesh" : "Lucknow", "Madhya Pradesh" : "Bhopal", "Maharashtra" : "Mumbai", "Rajasthan" : "Jaipur", "Gujarat" : "Gandhi Nagar"}
>>> state.items ( )
dict _ items ([('Uttar Pradesh', 'Lucknow'),
('Madhya Pradesh', 'Bhopal'),
('Maharashtra', 'Mumbai'),
('Rajasthan', 'Jaipur').
('Gujarat', 'Gandhi Nagar')])
```

5. **keys ()** This method returns only keys of the dictionary.

Syntax dictionaryName.keys ()

e.g.
```
>>> state = {"Uttar Pradesh" : "Lucknow", "Madhya Pradesh" : "Bhopal", "Maharashtra" : "Mumbai", "Rajasthan" : "Jaipur", "Gujarat" : "Gandhi Nagar"}
>>> state.keys ( )
dict _ keys (['Uttar Pradesh', 'Madhya Pradesh', 'Maharashtra', 'Rajasthan', 'Gujarat'])
```

6. **values ( )** This method returns only values of the dictionary.

Syntax dictionaryName.values ()

e.g. >>> state = {"Uttar Pradesh" : "Lucknow", "Madhya Pradesh" : "Bhopal", "Maharashtra" : "Mumbai", "Rajasthan" : "Jaipur", "Gujarat" : "Gandhi Nagar"}

>>> state.values ()

dict _ values (['Lucknow', 'Bhopal', 'Mumbai', 'Jaipur', 'Gandhi Nagar'])

7. **update ()** This method updates the existing dictionary with new dictionary if needed.

Syntax dictionary1.update (dictionary2)

e.g. >>> student = {"Name" : "Rakhi", "Class" : 11, "Subject" : "Mathematics", "RollNo" : 27}

>>> student1 = {"Address" : "Delhi", "Subject" : "Science"}

>>> student.update (student1)

>>> student

{'Name' : 'Rakhi',

'Class' : 11,

'Subject' : 'Science'

'RollNo' : 27,

'Address' : 'Delhi'}

>>>student1

{'Address' : 'Delhi', 'Subject' : 'Science'}

❑❑

UNIT-IV
Database Concepts and The Structured Query Language

CHAPTER-4
DATABASE CONCEPTS

Revision Notes

- Data, in the context of databases, refers to all the single items that are stored in a database, either individually or as a set.
- Database is a collection of interrelated data which is stored together to serve multiple applications.

  A database has the following properties :
  - It is a coherent collection of data with some inherent meaning. A random assortment of data can't correctly be referred as database.
  - A database is a design, build properly with data for a specific purpose.
  - A database can be of any size and complexity.
  - A database may be generated and maintained manually or it can be computerised.

Database Management System (DBMS)

- It is a collection of programs that enables users to create and maintain a database.
- The DBMS is a general purpose system that facilitates the process of defining, constructing, manipulating and sharing database among various users and applications.

  e.g. : ORACLE, MS-ACCESS, FOXPRO, SQL, etc.
- Defining a database involves specifying a data type, structure and constraint of the data to be stored in the database.
- Sharing a database allows multiple users and programs to access the database simultaneously.

- The goal of a DBMS is to provide an environment that is both convenient and efficient to use in
  - retrieving information from the database.
  - storing information into the database.

  Databases are usually designed to manage large bodies of information. This involves
  - definition of structures for information storage (data modelling).
  - provision of mechanisms for the manipulation of information (file and systems structure, query processing)
  - providing the safety of information in the database (crash recovery and security)
  - concurrency control if the system is shared by users.

- **Components of Database systems**

  The major components of a database system are described below :

  (i) User : The users are the people who manage the database and perform different operations on it.

  (ii) Hardware : The hardware consists of various secondary storage devices as CD, DVD, floppy disks etc. on which data is stored and input/output devices as mouse, keyboard, printer, monitor etc, which are used for providing commands and retrieving the result.

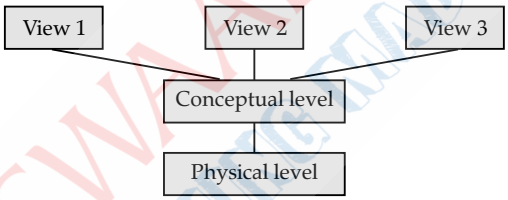
  (iii) Software : It acts as an interface between the user and the database.

  (iv) Data : It is an important component of database. It acts as a bridge between the hardware, software and the user.

- **Data Abstraction (View of Data)**

  Main purpose of a database system is to provide users with an abstract view of the system. The system hides certain details of how data is stored, created and maintained. All complexity are hidden from database users.

  Levels of Data abstraction : There are several levels of abstraction



1. **Physical Level**
   - How the data is stored :

     e.g. index, B-tree, hashing
   - Lowest level of abstraction :

     e.g. Data compression and encryption techniques
   - Complex low level structures
2. **Conceptual Level**
   - Next highest level of abstraction
   - Describes what data are stored
   - Describes the relationships among data
   - Database administrator level
3. **View Level**
   - Highest level
   - Describes part of the database for a particular group of users
   - Can be many different views of a database

     e.g., tellers in a bank get a view of customer accounts, but not of payroll data.

- **Data Model**

  Data model is a collection of conceptual tools for describing data, data relationship, data semantics and consistency constraints.

  There are three different data models as:

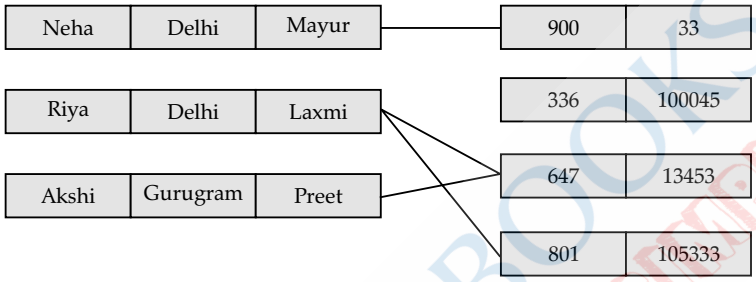
(i) Relational model (ii) Network model (iii) Hierarchical model

(i) Relational model : Data and relationship are represented by a collection of tables. Each table has multiple columns with unique names. *e.g.*, customer, account.

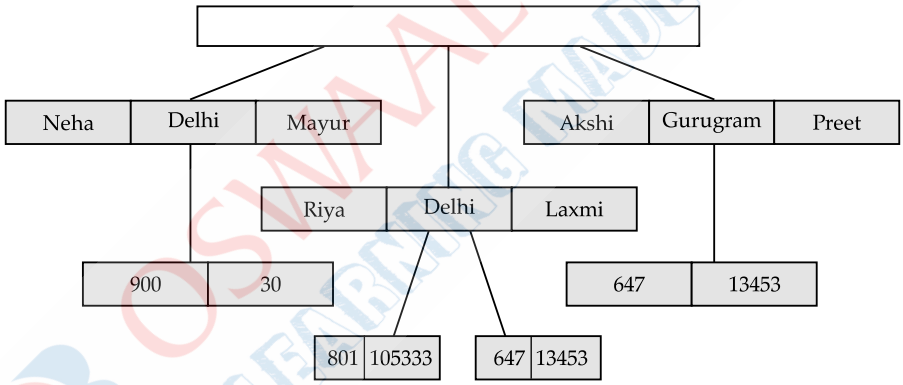
Name	City	Street	Number
Neha	Delhi	Mayur	900
Riya	Delhi	Laxmi	647
Riya	Delhi	Laxmi	801
Akshi	Gurugram	Preet	647

Number	Balance
900	33
336	100045
647	13453
801	105333

(ii) Network Model : Data are represented by collection of records. Relationships among data are represented by links. Organization is an arbitrary graph.



(iii) Hierarchical Model : It is similar to the network model. Organisation of the records is as a collection of trees rather than arbitrary graphs.



Relational Model

The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column name are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as a table. However, the physical storage of the data is interdependent of the way the data are logically organized.

Relational Model Terminology :

- **Attribute :** In a database management system (DBMS), an attribute refers to a data base component such as table columns. It also may refer to a database field. Attribute describes the instance in the row of database. *E.g.* Student, RollNo., Name etc.
- **Relation :** It is sometimes used to refer to a table in a relational database but is more commonly used to describe the relationships that can be created between those tables in a relational database. Relations have three important properties as name, cardinality and degree. These are described as :

  Name : The first property of a relation is its name which is represented by the title or entity identifier.

  Cardinality : It refers to the number of rows (tuples) in relation that defines the uniqueness of data value contained in a column.

  Degree : It refers to the number of columns (attributes) in a relation.

- **Domain :** It is defined as the set of all unique values permitted for an attribute. For example; a domain of dates is the set of all possible valid dates, a domain of integer is a set of all possible whole number, a domain of day of week is Monday, Tuesday ..... Sunday.
- **Tuples** : Rows of relations are generally termed as tuples.

Keys

- It is a data item that allows to uniquely identify individual occurrences of an entity type.
- An entity type usually has an attribute whose values are distinct for each individual entity in the entity set such a attribute is called key attribute.
- A key is normally correlated with one column in table and it might be associated with multiple tables.

There are different types of keys as follows :

(i) Primary Key

- The primary key of a relation can be said to be a minimal super key.
- The field or group of fields which forms the unique identifier for a table is called the table's primary key.
- The primary key uniquely identifies each record in the table and must never be the same for two records.

e.g. EmpCode can be primary key for entity set Employees. The primary key should be chosen such that its attributes are never or very rarely changed. For instance, the address field of a person should not be part of the primary key, since it is likely to change.

EmpCode, on the other hand, is guaranteed to never change, till the employee is in the organisation.

(ii) Candidate Key

- There is only one primary key in a table. But there can be multiple candidate keys. A candidate key is an attribute or set of attributes that uniquely identifies a record.
- These attributes or combinations of attributes are called candidate keys.
- When there are more than one candidate keys, one of the candidate key is chosen to be a primary key. The remaining candidate keys are called alternate keys.

(iii) Alternate key

If any one of the candidate key among the different candidate keys available, is selected as primary key then remaining keys are called alternate keys.

(iv) Foreign Key

- In a relation, column whose data values correspond to the values of a primary key column in another relation is called foreign key.
- In a relational database, the foreign key of a relation would be the primary key of another relation.

(v) Super key

For an entity, it is a set of one or more attributes whose combined value uniquely identifies the entities in the entity set.

e.g. For an entity set Employees, the set of attributes (EmpName, Address) can be considered to be a super key, if we assume that there are no two employees with the same name EmpName and same address.

❑❑

CHAPTER-5
STRUCTURED QUERY LANGUAGE (SQL)

Revision Notes

- **SQL (Structured Query Language)** consists of commands, each command has an operational part and a condition part. Operation is executed through a search in all relations defined in relational database.
- Required solution (list of data) is returned by the command. The data listed in solution satisfies all

conditions given in that command.

- SQL is a comprehensive database language, it has statements for data definition, query and update. Hence, it is both DDL and DML.
- In addition, it has facilities for defining views on the database for specifying security and authorization, for defining integrity constraints and for specifying transaction controls. It also has rules for embedding SQL statements into a general purpose programming language such as C or Pascal.
- SQL provides commands for a variety of tasks including
  - **(i)** querying data
  - **(ii)** inserting, updating and deleting rows in a table
  - **(iii)** creating, altering, and dropping objects
  - **(iv)** controlling access into the database and its objects.
- Installing MySQL

  MySQL is an open source relational DBMS software which can be downloaded from internet directly. It has officoal website where you can download it.

  http://dev.mysql.com/downloads

  After installing MySQL, you can work on it.
- **Types of SQL statements**

  SQL statements can be divided into four major categories

  (1) **Data Manipulation Language (DML) -** These statements consist of queries that insert data into tables in a database and statements that change the data in the database. SQL statements under this category are INSERT, UPDATE, DELETE etc.

  (2) **Data Definition Language (DDL) -** These statements define the structure of the database. DDL consists of these statements that create, alter and drop database objects. Statements under this category are CREATE, ALTER, DROP, etc.

  (3) **Transaction Control Language (TCL) -** These commands manage changes made by data manipu lation language commands. These commands are COMMIT, ROLLBACK, etc.

  (4) **Data Query Language (DQL) -** The commands of SQL that are used to retrieve data from the database. SQL statement under this category is SELECT.
- **Introduction to MySQL**
  - MySQL is an open source Relational Database Management System (RDBMS) based on SQL. In MySQL database, information is stored in tables and runs virtually on all platforms including Linux, Unix and Windows.
  - It provides many features such as storing, maintaining and controlling data in a secured environment. It is a fast, reliable, scalable substitute to many of the commercial RDBMS available today.
- **Data Types**

  Data type indicates the type of data value that an attribute can have. Commonly used data types are :
  - **VARCHAR 2 (SIZE) :** It is variable length characters string having maximum length size bytes. It can hold 4000 bytes of characters.
  - **CHAR (size) :** It specifies a fixed length characters string. Maximum size is 255.
  - **VARCHAR (size) :** This data type is currently synonymous with VARCHAR 2 datatype.
  - **NUMERIC (P,S)**: It is used to store fixed or floating point numbers.

    where

    P-Precision or total number of digits in range 1 to 38.

    S-Scale or numbers of digits to right of decimal point.
  - **LONG:** It stores variable length character strings containing up to 2 gigabytes.

- **DATE:** It is used to store date information. Default format is DD-MM-YYYY.

➢ **Create Database**

To create database, use CREATE DATABASE statement as following :

CREATE DATABASE database_name;

e.g. create a database named company

mysql > CREATE DATABASE company;

➢ **Create Table**

CREATE TABLE statement allows you to create a new table in a database.

Create command used to create database objects.

Syntax

CREATE TABLE tableName

(Column_name 1 datatype [column constraint],
Column_name 2 datatype [column constraint],
⋮
Column_name n datatype [column constraint]);

The following statement creates an EMPLOYEE table :

CREATE TABLE EMPLOYEE

(Emp_No INTEGER PRIMARY KEY,
Emp_Name char (30) NOT NULL,
DEPT CHAR (20)
Salary Numeric (10,4)
);

➢ **DESCRIBE Table**

We can view the structure of an already created table using the DESCRIBE statement.

Syntax

DESCRIBE tableName;

➢ **Altering definition of table : Alter Command**

ALTER TABLE command is used to alter the definition of a table in the database.

Syntax

ALTER TABLE Table_name ADD/MODIFY/DROP column_name datatype size;

e.g ALTER TABLE Employee ADD age INTEGER;

Some times we need to change the datatype of a column. To do this, we use ALTER TABLE with MODIFY column command syntax

Syntax

ALTER TABLE Table_name MODIFY column_name datatype **(size);**

e.g ALTER TABLE Employee MODIFY Numeric (10, 4);

➢ **INSERT Command**

INSERT command is used to add rows to a table.

Syntax

INSERT INTO Table_name VALUES (value1, value2, ...);

The number and sequence of values should match that of columns in the table. If the number of values is less then specify the column names into which data is being entered as illustrated.

INSERT INTO Table_name (column1, column2) VALUES (data_value1, data_value2);

➢ To insert null values. NULL may be used,

INSERT INTO EMP VALUES (1001,'Sharma',NULL, 3000, NULL);

➢ **SELECT Command**

SELECT command is used to retrieve the sub part of rows or columns from one or more tables.

Syntax

SELECT column_name FROM Table_name;

Table:Employee

Emp_Code	Emp_Name	Designation	Salary	Joining_Date
1001	Rahul	Accountant	25000	2011-5-25
1002	Krishna	Clerk	20000	2010-6-19
1003	Akshat	Accountant	22000	2012-7-22
1004	Apoorvi	Clerk	18000	2013-3-17
1005	Nishant	Supervisor	24000	2016-4-23
1006	Sonam	Accountant	22000	2010-5-24
1007	Pihu	Manager	38000	2012-6-18

e.g SELECT Emp_Name FROM Employee;

Output

Emp_Name
Rahul
Krishna
Akshat
Apoorvi
Nishant
Sonam
Pihu

If you want to display the details of all employee, asterisk * is used.

Syntax SELECT*FROM Table_name ;

e.g. SELECT* FROM Employee;

➢ **Use Distinct keyword**

DISTINCT keyword is used to eliminate the duplicate rows from the result of SELECT statement.

Syntax SELECT DISTINCT column_name FROM Table_name

e g SELECT DISTINCT Designation FROM Employee;

Output

Designation
Clerk
Accountant
Supervisor
Manager

➢ **Use ALL Keyword**

If you use ALL keyword instead of DISTINCT, it will give all rows with duplicate from the result of a SELECT statement.

Syntax SELECT ALL column_name FROM Table_name ;

e.g. SELECT ALL Designation FROM Employee;

Output

Designation
Accountant
Clerk
Accountant
Clerk
Supervisor
Accountant
Manager

➢ **Use WHERE clause**

WHERE clause in SELECT statement specifies the criteria for selection of rules to be returned. It gives the particular result based on some conditions.

Syntax SELECT column_name 1, column_name 2

FROM Table_name WHERE condition;

e.g SELECT Emp_Name, Designation FROM Employee WHERE Salary >22000;

Output

Emp_Name	Designation
Rahul	Accountant
Nishant	Supervisor
Pihu	Manager

➢ **Logical Operator**

Logical operators are used with WHERE clause. These operators are:

- **OR operator** ***e.g.*** To list the employee's details having Emp_Code as 1004 or 1006 from table Employee.

  SELECT Emp_Code, Emp_name, Designation, Salary FROM Employee WHERE (Emp_Code =1004 or Emp_Code 1006);

- **AND Operator** ***e.g.*** To list the employee's details having Emp_Code as 1005 but with Salary less than 24000.

  SELECT Emp_Code, Emp_Name , Designation, Salary FROM Employee WHERE (Emp_Code 1005 AND Salary < 24000);

- **NOT operator** ***e.g.*** To list all employee's details whose Emp_Code are other than 1004.

  SELECT Emp_Code, Emp_Name, Designation, Salary FROM Employee Where (NOT Emp_Code =1004);

➢ **BETWEEN operator** This operator defines the specified range of values that come to make condition true. In this, lower limit and upper limit are given as a range.

e.g. SELECT Emp_Code, Emp_Name , Designation FROM Employee

WHERE Salary BETWEEN 21000 AND 25000;

Output

Emp_Code	Emp_Name	Designation
1001	Rahul	Accountant
1003	Akshat	Accountant
1005	Nishant	Supervisor
1006	Sonam	Accountant

➢ **IN Operator**

This operator allows you to easily test if the expression matches any value in the list of values. You can also use NOT IN to exclude the rows in your list.

e.g. To list all the employee's with designation Clerk and Manager

SELECT Emp_Code, Emp_Name, Designation, Salary FROM Employee.

WHERE Designation IN ('Clerk ','Manager);

But if you want to display the details of those employees whose designation not in Accountant.

e.g. SELECT* FROM Employee

WHERE Designation NOT IN ('Accountant');

➢ **LIKE Operator**

This operator is used in a WHERE clause to search for specified pattern in a column. Pattern are defined using two special wildcard characters.

- percent (%) matches any substring
- underscore (_) matches any one character

To illustrate pattern matching

- "abc%" matches any string that start with 'abc'
- "%abc%" matches any string in which abc comes in mid
- "---" matches any string of exactly four characters
- "---%" matches any string of at least 3 characters

e.g. SELECT Emp_Code, Emp_Name FROM Employee WHERE Emp_Name LIKE "%oo%";

Output

Emp_Code	Emp _Name
1004	Apoorvi

➢ **DELETE Command**

DELETE Command is used to remove the one or more record(s) from a table. It does not delete individual field from the table.

Syntax DELETE FROM Table_Name WHERE condition;

e.g. DELETE FROM Employee WHERE Emp_Code = 1004;

SELECT* FROM Employee;

Output

Emp_Code	Emp_Name	Designation	Salary	Joining_Date
1001	Rahul	Accountant	25000	2011-5-25
1002	Krishna	Clerk	20000	2010-6-19
1003	Akshat	Accountant	22000	2012-7-22
1005	Nishant	Supervisor	24000	2016-4-23
1006	Sonam	Accountant	22000	2010-5-24
1007	Pihu	Manager	38000	2012-6-18

To remove the all rows from table use

DELETE FROM Table_name;

It will delete all rows but not a table structure.

➢ **DROP Command**

DROP command is used to delete a whole database or just a table. It destroys the objects like an existing database, table, index, etc.

Syntax DROP TABLE Table_Name;

Or

DROP DATABASE database_name ;

➢ **UPDATE Command**

UPDATE command is used to modify the existing records in a table. You can use WHERE clause with the UPDATE command to update the selected rows, otherwise all rows would be affected.

Syntax

```
UPDATE Table_name
SET Column1 = value1, Column2 = value2 .....
WHERE condition;
```

e.g. To change the salary of all employees to 35000, use this

```
UPDATE Employee
Set Salary =35000;
```

But if you want to change the salary to 35000 only for those Employees that have salary less then or equal to 20000, use this

```
UPDATE Employee
SET Salary = 35000
WHERE Salary <=20000;
```

To update multiple columns, change the salary and designation of those Employees who have Emp_Code 1002

```
UPDATE Employee
SET Salary = 40000, Designation = 'Supervisor' WHERE Emp_Code = 1002;
```

❑❑

CHAPTER-6
INTRODUCTION TO THE EMERGING TRENDS

Revision Notes

- Emerging trends are the state of the art technologies, which gain popularity and set a new trend among users.
- Knowledge of emerging trends is particularly important to individuals and companies who are charged with monitoring a particular field or company.
- Some emerging trends that will make a huge impact on digital economy and interaction in digital societies.

➢ **Artificial Intelligence (AI)**

- AI is wide ranging branch of computer science that refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions.
- The term artificial intelligence was coined in 1956, but AI has become more popular today thanks to increased data volumes, advanced algorithms and improvement in computing power and storage.
- The intelligent digital personal assistants like Google now, Cortana, Alexa are all powered by AI. Artificial Intelligence endeavours to simulate the natural intelligence of human beings into machines, thus making them behave intelligently.
- The goals of artificial intelligence include learning, reasoning and perception.
- AI is being tested and used in the healthcare industry for dosing drugs and different treatment in patients and for surgical procedures in the operating room.
- Artificial intelligence also has applications in the financial industry, where it is used to detect and flag activity in banking and finance such as debit card usage and large account deposits all of which help a bank's fraud department.

- AI provides virtual shopping capabilities that offer personalized recommendations and discuss purchase options with the consumer. Stock management will also be improved with AI.

➢ **Machine Learning**

- Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. It focuses on the development of computer programs that can access data and use it to learn for themselves.
- The term machine learning was coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence.
- Banks and other business in the financial industry uses machine learning technology for two key purposes : to identify important insights in data and prevent fraud.
- Machine learning is a fast growing trend in the health care industry, thanks to the advent of wearable devices and sensors that can use data to assess a patient's health in real time.
- The data analysis and modeling aspects of machine learning are important tools for delivery companies, public transportations and other transportation organizations.

➢ **Natural Language Processing (NLP)**

- NLP is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language.
- Natural language processing represents the automatic handling of natural human language like speech or text and although the concept itself is fascinating, the real value behind this technology comes from the use cases.
- NLP is important because it helps resolving ambiguity in language and adds useful numeric structure to the data for many downstream applications such as speech recognition or text analytics.
- It includes many different techniques for interpreting human language, ranging from statistical and machine learning methods to rules and algorithmic based approaches. We need a broad array of approaches because the text and voice based data varies widely as do the practical applications.
- Benefits of NLP are :

  (i) Improved accuracy and efficiency of documentation.

  (ii) Useful for personal assistants such as Alexa.

  (iii) Allows an organization to use chatbets for customer support.

  (iv) Easier to perform sentiment analysis.

➢ **Immersive Experience**

- Immersive experience allow us to visualize, feel and react by stimulating our senses. It enhances our interaction and involvement making them more realistic and engaging.
- An immersive experience is the perception of being in one place when you are actually in another. It is essentially the suspension of reality, even if just for a few moments.

➢ **Augmented Reality (AR)**

- The superimposition of computer generated perceptual information over the existing physical surroundings is called as Augmented Reality (AR).
- The most important benefit of AR is that to the user it feels like a natural extension. AR only adds or hides data from the environment. With the help of location AR App, travellers can access real time information of historical places just by pointing their camera viewfinder to the subjects.

➢ **Virtual Reality (VR)**

- Virtual reality is a three dimensional, computer generated situation that simulates the real world. VR tries to stimulate as many senses as possible to make the user feel as if they are really there.
- Google's Expeditions is a great example of mobile VR in action, offering users the opportunity to explore imaginary world.

➢ **Robotics**

- A robot is basically a machine capable of carrying out one or more tasks automatically with accuracy and precision. Unlike other machines, a robot is programmable which means it can follow the instructions given through computer programs.

- Robots were initially conceptualized for doing repetitive industrial tasks that are boring or stressful for humans or were labour intensive.
- Robot can be of many types, such as wheeled robots, legged robots, manipulators and humanoids. Robots that resemble humans are known as humanoids.
- Robotics is the intersection of science, engineering and technology that produces machines called robots, that substitute for human actions. Many aspects of robotics involve artificial intelligence; robots may be equipped with the equivalent of human senses such as vision, touch and the ability to sense temperature.
- Robots are being used in industries, medical science, bionics, scientific research, military etc.
- Some example of robotics are NASA's Mars Exploration Rover (MER), Sophia, Drone etc.

➢ **Big Data**

- Big data is a combination of structured, semi structured and unstructured data collected by organizations that can be ruined for information and used in machine learning projects, predictive modeling and other advanced analytics applications.
- Big data not only represents voluminous data, it also involves various challenges like integration, storage, analysis, searching, processing, transfer, querying and visualization of such data.
- Big data sometimes hold rich information and knowledge which is of high business value and therefore there is a keen effort in developing software and methods to process and analyse big data.

➢ **Characteristics of Big Data**

- **Volume :** It refers to the unimaginable amount of information generated every second from social media, cell phones, images, videos etc.
- **Variety :** Big data is generated in multiple varieties, such as structured, semi-structured and unstructured.
- **Veracity :** Veracity basically means the degree of reliability that the data has to offer.
- **Value :** It is the major issue that we need to concentrate on. It is actually the amount of valuable and reliable data that need to be stored.
- **Velocity :** It plays a major role compared to the others, there is no point in investing so much to end up waiting for the data.

➢ **Internet of Things (IoT)**

- IoT refers to the billions of physical devices around the world that are now connected to the Internet.
- It is a giant network of connected things and people all of which collect and share data about the way they are used and about the environment around them.
- IoT exploits recent advances in software, falling hardware prices and modern attitudes towards technology. It's new and advanced elements bring major changes in the delivery of products, goods and services; and the social, economic and political impact of those changes.
- IoT makes areas of improvement clear. Current analytics gives us superficial insight but IoT provides real world information leading to more effective management of resources.

➢ **Sensors**

- Sensors are very commonly used for monitoring and observing elements in real world applications. The evaluation of smart electronic sensors is contributing in a large way to the evolution of IoT. It will lead to creation of new sensor based intelligent systems.
- A smart sensor is a device that takes input from the physical environment and uses built in computing resources to perform predefined functions upon detection of specific input and then process data before passing it on.
- Sensors are separate entities that need to be connected separately to the interfacing circuit of the device. This causes difficulties related to embedding sensors into the device as well as their maintenance.

➢ **Smart Cities**

- Smart city is a framework, predominantly composed of Information and Communication Technologies to develop, deploy and promote sustainable development practices to address growing urbanization challenges.

- Smart cities improve the quality of the lives of citizens. They often employ a mobile app to give fast access to traffic information, road conditions and more.
- Smart cities affect everyone whether directly or indirectly. People who live in smart cities or who are vising smart cities have the immediate benefit of being connected to the governing body for information and services.

➢ **Cloud Computing**

- It is an emerging trend in the field of information technology, where computer based services are delivered over the Internet or the cloud, for the case of their accessibility from any where using any smart device.
- One benefit of using cloud computing services is that firms can avoid the upfront cost and complexity of owning and maintaining their own IT infrastructure and instead simply pay for what they use, when they use it.
- Through, cloud computing, a user can run a bigger application or process a large amount of data without having the required storage or processing power on their personal computer as long as they are connected to the Internet.

➢ **Cloud Services**

- Cloud computing can be separated into three general services as :
- **(i) Infrastructure as a Service (IaaS) :** It contains the basic building blocks for cloud IT. It provides access to networking features, computers and data storage space.
- **(ii) Platform as a Service (PaaS) :** It removes the need for you to manage underlying infrastructure and allows you to focus on the deployment and management of your applications.
- **(iii) Software as a Service (SaaS) :** It provides you with a complete product that is run and managed by the service provider. In most cases, people referring to SaaS are referring to end user application.

➢ **Grid Computing**

- It is a computer network in which each computer's resources are shared with every other computer in the system. The grid computing concept is not a new one. It is a special kind of distributed computing.
- Grids have a variety of resources based on diverse software and hardware structures, computer languages and frameworks, either in a network or by using open standards with specific guidelines to achieve a common goal.

  Grid can be of two types :

  (i) Data grid, used to manage large and distributed data having the required multi-user access.

  (ii) Processor grid, where processing is moved from one PC to another as needed or a large task is divided into subtasks, and allotted to various nodes for parallel processing.

  To set up a grid, by connecting numerous nodes in terms of data as well as CPU, a middleware is required to implement the distributed processor architecture.

  The Globus toolkit is one such software toolkit used for building grids and it is open source.

➢ **Block Chain Technology**

- A block chain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computers systems on the block chain.
- The block chain technology works on the concept of decentralised and shared database where each computer has a copy of the database.
- The most popular application of blockchains technology is in digital currency. However, due to its decentralised nature with openness and security, block chains are being seen as one of the ways to ensure transparency, accountability and efficiency in business as well as in governance systems.
- The bock chain technology can be used in diverse sectors, such as banking, media,telecom, travel and hospitability and other areas.

❑❑