**UNIT 1: Computer Fundamentals**

**Introduction to Computers**

 A computer is an electronic device which can process data to produce required result and can store the data for future use. Basically it has the following components

 i) Input device ii) Output device iii) Memory device

 iv) Arithmetic and Logic unit v) Control unit

Control Unit

Arithmetic & Logic Unit

Memory

Input Device

Output Device

Central Processing Unit

Data can be entered in a computer through the input device, the processor processes those and the result can be passed to output device or can be stored in storage device. The Control unit supervises the flow of information between the various units.

 Any task that can be performed systematically, using a precise step-by-step method, can be performed by a computer. To perform a particular task, the user must give the set of instructions and data to the computer. These instructions direct the computer to execute task.

**Characteristics of Computer**

* **Speed**

Computers work at an incredible speed. A powerful computer is capable of performing about 3-4 million simple instructions per second.

* **Accuracy**

In addition to being fast, computers are also accurate. Errors that may occur can almost always be attributed to human error.

* **Diligence**

Unlike human beings, computers are highly consistent. They do not suffer from human traits of boredom and tiredness resulting in lack of concentration. Computers, therefore, are better than human beings in performing voluminous and repetitive jobs.

* **Versatility**

Computers are versatile machines and are capable of performing any task as long as it can be broken down into a series of logical steps. The presence of computers can be seen in almost every sphere – Railway/Air reservation, Banks, Hotels, Weather forecasting and many more.

* **Storage Capacity**

Today’s computers can store large volumes of data. A piece of information once recorded (or stored) in the computer, can never be forgotten and can be retrieved almost instantaneously.

**Uses of computers**

 There are the following uses of computer are:

1. For filling online forms
2. For banking purpose
3. In hospitals like ICU
4. In coding and programming
5. For communication
6. For playing games
7. For designing
8. For education purpose
9. Uses in business and marketing
10. For storing important data
11. For getting information using internet
12. For multi-purpose use like video, songs, photos, etc.

**Types of Computers**

 The four basic types of computers are as under:

1. **​Supercomputer: -** The most powerful computers in terms of performance and data processing are the Supercomputers. These are specialized and task specific computers used by large organizations. These computers are used for research and exploration purposes, like NASA uses supercomputers for launching space shuttles, controlling them and for space exploration purpose. Popular Supercomputers are
* ​IBM’s Sequoia, in United States
* ​Fujitsu’s K Computer in Japan
* ​IBM’s Mira in United States
* ​IBM’s SuperMUC in Germany
* ​NUDT Tianhe-1A in China
1. **​Mainframe Computer: -** Mainframes are quite expensive and many large firms & government organizations uses Mainframes to run their business operations. The Mainframe computers can be accommodated in large air-conditioned rooms because of its size. Mainframes can process & store large amount of data. Banks educational institutions & insurance companies use mainframe computers to store data about their customers, students & insurance policy holders. Popular mainframe computers are

Fujitsu’s ICL VME Hitachi’s Z800

1. **​Minicomputer: -** Minicomputers are used by small businesses & firms. These are small machines and can be accommodated on a disk. Individual departments of a large company or organizations use Mini-computers for specific purposes. Popular mini computers are

K-202, Texas Instrument TI-990, SDS-92, IBM Midrange computers

1. **​Microcomputer: -** Desktop computers, laptops, personal digital assistant (PDA), tablets & smartphones are all types of microcomputers. These computers are the cheapest among the other three types of computers. The Micro-computers are specially designed for general usage like entertainment, education and work purposes. Well known manufacturers of Micro-computer are Dell, Apple, Samsung, Sony & Toshiba.

**Generation of Computer**

There are five generations of computer.

1. **First Generation of Computer**: - The first computer systems used vacuum tubes for circuitry and [magnetic drums](https://www.webopedia.com/TERM/M/magnetic_drum.html) for [memory](https://www.webopedia.com/TERM/M/memory.html). These computers were very expensive to operate and in addition to using a great deal of electricity, the first computers generated a lot of heat, which was often the cause of malfunctions. The UNIVAC and [ENIAC](https://www.webopedia.com/TERM/E/ENIAC.html) computers are examples of first-generation computing devices.
2. **Second Generation of Computer**: - The [transistors](https://www.webopedia.com/TERM/T/transistor.html) replace vacuum tubes in the second generation of computers. The transistor allowing computers to become smaller, faster, cheaper, more energy-efficient and more reliable than their first-generation predecessors. Second-generation computers still relied on punched cards for input and printouts for output.
3. **Third Generation of Computer**: - The [integrated circuit](https://www.webopedia.com/TERM/I/integrated_circuit_IC.html) was used in the third generation of computers. Instead of punched cards and printouts, users interacted with third generation computers through [keyboards](https://www.webopedia.com/TERM/K/keyboard.html) and [monitors](https://www.webopedia.com/TERM/M/monitor.html) and [interfaced](https://www.webopedia.com/TERM/I/interface.html) with an [operating system](https://www.webopedia.com/TERM/O/operating_system.html),
4. **Fourth Generation of Computer**: - The [microprocessor](https://www.webopedia.com/TERM/M/microprocessor.html) brought the fourth generation of computers, as thousands of integrated circuits were built onto a single silicon chip. The Intel 4004 chip, developed in 1971. In 1981 [IBM](https://www.webopedia.com/TERM/I/IBM.html) introduced its first computer for the home user, and in 1984 [Apple](https://webopedia.com/TERM/A/Apple_Computer.html) introduced the Macintosh. Microprocessors also moved out of the realm of desktop computers .
5. **Fifth Generation of Computer**: - Fifth generation computing devices, based on [artificial intelligence](https://www.webopedia.com/TERM/A/artificial_intelligence.html), are still in development, though there are some applications, such as [voice recognition](https://www.webopedia.com/TERM/V/voice_recognition.html), that are being used today. The use of [parallel processing](https://www.webopedia.com/TERM/P/parallel_processing.html) and superconductors is helping to make artificial intelligence a reality.

**UNIT 2: Basic Computer Organization**

**Units of a computer**

**Central Processing Unit**: - The Central Processing Unit (CPU) is the brain of a computer where all kinds of processing is done. This unit takes the input data from the input device and processes it according t the set of instructions. It has four major parts called ALU and Control Unit.

1. ALU: - It is responsible for carrying out arithmetical and logical operations. Arithmetic operation includes addition, subtraction, multiplication and division. Logical operation includes AND, OR, NOT and Exclusive OR operation. This unit further performs increment, decrement and shift and clearance operations as well.
2. Control Unit: - This unit is mainly used for generating the electronic control signals for the synchronization of various operations. All operation related to memory, input and output device are synchronized through the control signal generated by the control unit. It control all the operations of the computer.

**Memory hierarchy**

**Memory** : Any device that can stores data or instructions on the computer is called memory. A memory is made up of a large number of cells, with each cell capable of string one bit of information. There are two types of memory : volatile and nonvolatile. Volatile memory loses all information as soon as the computer is turned off. Nonvolatile memory does not lose its information when the computer is turned off. Different types of memory devices are :

 1. Random Access Memory (RAM): - RAM is volatile and is used in computer for its primary memory. All active programs and data are stored in RAM so that they are readily available and easily accessed by the CPU. There are different types of RAM

 i) Static RAM (SRAM): - In SRAM, it does not need to re-written data periodically. When the actual write command is executed, the data on the memory is refreshed or changed. It is very fast but expensive.

 ii) Dynamic RAM (DRAM): - In DRAM, data must be continually re-written. It re-write data several hundred time per second. It is cheap and small.

 2. Read Only Memory (ROM): - ROM is non-volatile that is data stored in this memory can be read but cannot be changed. Computer’s startup instructions and system BIOS are stored in ROM because user cannot disrupt the information. It is slower than RAM. Different types of ROM are

 i) Progarmmable ROM (PROM): - This type of ROM is programmed only one time using a special type of programming device called RPOM burner.

 ii) Erasable Programmable ROM (EPROM): - This type of ROM is erasable by shining a special ultra-violet light and can be reprogrammed.

 iii) Electrically Erasable Programmable ROM (EEPROM): - This ROM can be re-written through the use of a special software program. It also called flash ROM or flash BIOS.

 3. Cache: - Since the speed of CPU and RAM is different, CPU has to wait for the data requested from RAM. It reduces the performance of CPU. To overcome this problem cache memory is used. It is a very fast and small memory that store all data and instructions frequently used by the CPU. It is placed between RAM and CPU.

 4. Secondary storage device: - All secondary storage devices are non-volatile that can store data and programs permanently. Different secondary storage devices are

 i) Hard disk: - Magnetic disks are smooth metal plates coated on both sides with a thin film of magnetic material. A set of such magnetic plates is fixed to a spindle one below the other to make up a disk pack. This pack is called hard disk. The drive also has a set of magnetic heads that read/write data on surface. Each surface is divided into some circular area called track. Each track is again divided into some sectors. Data is recorded on the surface of a disk. The storage capacity of a disk pack depends upon the number of recording surfaces, the bits per inch of track, the tracks per inch of surface and the block size. It is fast, reliable and low priced. It comes in many sizes and capacities. Nowadays (2004) 40 GB is the smallest capacity for PC.

 ii) Floppy disk: - Floppy disks are made of magnetic oxide-coated Mylar computer tape material. Tape material is of circular structure and 3.5 inches in diameter. Floppy disk is covered in a 3.5 inches square hard plastic envelope with a long slit for read-write head access and a hole in the center for mounting the disk drive hub. The tape is divided into 192 circular tracks, 9 sectors per track, 512 bytes on one side of the disk. The rotation speed of a floppy is of the order of 366 rmp with a transfer rate of 40 kilobytes/second. Floppy disks are compact, inexpensive and portable mainly used for micro and mini computer system.

 iii) Magnetic tape: - Magnetic tape memories are similar to the audio tape recorder, commonly used for storing large amounts of data in a predetermined orderly sequence. The tape medium contains data and is sealed in a cartridge. Two major technologies are used in magnetic tape. One is called Ultrium uses 12.5 mm width tape of length around 650 metres storing 400 GB in compressed form. The second uses 8 or 4 mm width tape of length around 150 metres with maximum capacity of around 20 to 100 GB compressed. Magnetic tape can transfer data at a speed of around 1 million bytes per second.

 iv) Compact disk: - The latest and the most popular high capacity secondary storage is known as laser disk technology. Compact disk also known as laser disk or optical disk is a shiny metal like disk whose diameter is 5.25 inches. It can store around 650 MB of data. Information in CD is written by creating pits of around 0.8 micrometre diameter in size, on the disk surface by shining a laser beam. The presence of pit represents 1 otherwise 0(zero). CD is rotated in the drive at a speed of 360 revolutions per minute. Generally CDs are read only but erasable CDs are also available.

**Registers**

 A computer register (CPU register) is one of a small set of data holding places that are part of the computer [processor](https://whatis.techtarget.com/definition/processor). A register may hold an [instruction](https://whatis.techtarget.com/definition/instruction), a storage address, or any kind of. A register must be large enough to hold an instruction - for example, in a [64-bit computer](https://searchdatacenter.techtarget.com/definition/64-bit-processor), a register must be 64 [bits](https://whatis.techtarget.com/definition/bit-binary-digit) in length.

**I/O devices:**

**Input device**: - Input device is responsible for reading data and instruction from user or input medium. It converts these data and instruction into a particular form that the computer can understand and process, and store into computer memory. Different types of input devices can be used according to the requirement of the user.

 a) Keyboard: - Keyboard is the most commonly used input device to enter alphanumeric character manually. It consists of different keys like alphanumeric keys, numeric keys, function keys and cursor movement keys. The pressing of different keys send a different character code corresponding to key depressed to the memory.

 b) Mouse: - Mouse is pointing device used to point any position on the monitor screen. The pointer can be used to select text, access menus and submenus, interact with other program etc. All windows-based program use mouse, making working with computers easier without typing commands.

2. **Output Device**: - Output device represent the result of processing from memory. It converts binary data store in computer memory to the form that is understand by user. Different types of output devices can be used according to the requirement of the user.

1. Monitor: - Monitors are of two types. First the typical monitor consist of a cathode ray tube (CRT) to display the input data, message and result of processing. Each character entered through the keyboard is displayed simultaneously on the CRT. A small square or underscore character, called cursor, indicates the location on the screen of the monitor. The second is the flat panel display, which can be used with a portable PC, notebook computer or hung on the wall. Both these type may be monochrome or multicolour.
2. Printer: - Printer is the most popular output device, which give permanent record of output data in human readable form. There are two categories of printer
3. Dot-Matrix printer: - It prints image as a combination of dots. The print head is composed of a matrix of tiny needles generally 9 pins or 24 pins that strike on inked ribbon to print the image on the paper. The dot density and the accuracy of positioning of the dots determine the print quality. It can print the character of any type of font.

i) Ink-jet printer: - It sprays small drops of ink through tiny nozzles onto paper to form required image. It provides a print resolution of around 360 dots per inch. Colour ink-jet printer contains four colours: cyan, magenta, yellow and black. Combining these four colours it can create almost all colour printing.

e. Plotter: - Plotter directly produces a hardcopy record of image on a removable medium in the form of a two-dimensional graphic representation. It uses a robotic arm to draw with coloured pens. A high degree of accuracy can be achieved with plotters even to within 1000th of an inch. It is mainly used to produce large pictures or images such as drawing graphs, making maps, construction plans for buildings, plotting contour lines and detailing of civil engineering structures.

**UNIT 3: Planning the Computer Program**

Concept of problem solving

Problem definition

Program design

Debugging

Types of errors in programming

Documentation

**UNIT 4: Techniques of Problem Solving**

**Flowcharting**

Flowchart is the pictorial representation of the algorithm. It is primarily used as an aid in formulating and understanding algorithm. The sequencing of instructions and repetition of groups of instructions may be quickly seen by inspecting a flowchart.

 The following shapes are used for various purpose of flowchart :

i) Start / Stop ii) Input / Output

iii) Processing iv) Condition

**Decision table**

A decision table is a table of contingencies for defining a problem and the actions to be taken. It is a single representation of the relationships between conditions and actions.

 A decision table consists of two parts: stub and entry. The stub part is divided into an upper quadrant called the condition stub and a lower quadrant called the action stub. The entry part is also divided into an upper quadrant, called the condition entry and a lower quadrant called the action entry.

Rules: -

1. A decision should be given a name
2. The logic of the decision table is independent of the sequence in which the condition rules are written, but action takes place in the order in which the events occur.
3. Standardized language must be used consistently
4. Duplication of terms or meaning should be eliminated.

Condition Stub

Condition entry

Action stub

Action entry

**Algorithms**

 An algorithm is a set of instructions, typically to solve a problem or perform a computation. an algorithm can be expressed within a finite amount of space and time[[1]](https://en.wikipedia.org/wiki/Algorithm#cite_note-1) and in a well-defined formal language[[2]](https://en.wikipedia.org/wiki/Algorithm#cite_note-2) for calculating a [function](https://en.wikipedia.org/wiki/Function_%28mathematics%29).

**Structured programming concepts**

 Structured programming is a [programming paradigm](https://en.wikipedia.org/wiki/Programming_paradigm) aimed at improving the clarity, quality, and development time of a [computer program](https://en.wikipedia.org/wiki/Computer_program) by making extensive use of the structured control flow constructs of selection ([if/then/else](https://en.wikipedia.org/wiki/Conditional_%28computer_programming%29)) and repetition (while and [for](https://en.wikipedia.org/wiki/For_loop)), [block structures](https://en.wikipedia.org/wiki/Block_%28programming%29), and [subroutines](https://en.wikipedia.org/wiki/Subroutines).

**Programming methodologies**

 When programs are developed to solve real-life problems like inventory management, payroll processing, student admissions, examination result processing, etc. they tend to be huge and complex. The approach to analyzing such complex problems, planning for software development and controlling the development process is called **programming methodology**.

## Types of Programming Methodologies

There are many types of programming methodologies prevalent among software developers −

### Procedural Programming

Problem is broken down into procedures, or blocks of code that perform one task each. All procedures taken together form the whole program. It is suitable only for small programs that have low level of complexity.

### Object-oriented Programming

Here the solution revolves around entities or objects that are part of problem. The solution deals with how to store data related to the entities, how the entities behave and how they interact with each other to give a cohesive solution.

### Functional Programming

Here the problem, or the desired solution, is broken down into functional units. Each unit performs its own task and is self-sufficient. These units are then stitched together to form the complete solution.

### Logical Programming

Here the problem is broken down into logical units rather than functional units. **Example:** In a school management system, users have very defined roles like class teacher, subject teacher, lab assistant, coordinator, academic in-charge, etc. So the software can be divided into units depending on user roles. Each user can have different interface, permissions, etc.

Software developers may choose one or a combination of more than one of these methodologies to develop a software. Note that in each of the methodologies discussed, problem has to be broken down into smaller units. To do this, developers use any of the following two approaches −

* Top-down approach
* Bottom-up approach

## Top-down or Modular Approach

The problem is broken down into smaller units, which may be further broken down into even smaller units. Each unit is called a **module**. Each module is a self-sufficient unit that has everything necessary to perform its task.

## Bottom-up Approach

In bottom-up approach, system design starts with the lowest level of components, which are then interconnected to get higher level components. This process continues till a hierarchy of all system components is generated. However, in real-life scenario it is very difficult to know all lowest level components at the outset. So bottom up approach is used only for very simple problems.

**UNIT 5: Overview of Programming**

Structure of a Python Program,

Elements of Python

**UNIT 6: Introduction to Python**

**Python Interpreter**

 Python is an interpreter language. It means it executes the code line by line.

**Using Python as calculator**

**Python shell**

 Python is an interpreter language. It means it executes the code line by line. Python provides a Python Shell (also known as Python Interactive Shell) which is used to execute a single Python command and get the result.

Python Shell waits for the input command from the user. As soon as the user enters the command, it executes it and displays the result.

We created a program and submitted it to the Python interpreter for execution. We can interact with the interpreter directly, typing in Python statements and expressions for its immediate execution. We also can type commands into the Python Shell pane, and the interpreter will attempt to execute them. The interpreter prompts the user for input with three greater-than symbols (>>>). This means the user typed in the text on the line prefixed with >>>. Any lines without the >>> prefix represent the interpreter’s output, or feedback, to the user.

**Indentation**

Python programs are structured through indentation. In Python, blocks of codes are defined by indentation not as a matter of style or preference but as a rigid language requirement. This principle makes Python codes more readable and understandable. A block of code can be easily identified when you look at a Python program as they start on the same distance to the right. If it has to be more deeply nestled, you can simply indent another block further to the right.

 To indicate a block of code in **Python**, you must **indent** each line of the block by the same whitespace. The two lines of code in the while loop are both **indented** four spaces. ... For **example**, j=1 and while(j<=5): is not **indented**, and so it is not within while block. So, **Python** code structures by **indentation**.

**Atoms**

 In Python, an atom is something that has a value. Identifiers, literals, strings, lists, tuples, sets, dictionaries, etc. are all atoms. An expression in Python, is any valid combination of operators and atoms. It is composed of one or more operations.

**Keywords**

Python keywords are reserved words in Python that should not be used as variable, constant, function name, or identifier in your code. For example: *assert* *break class continue def del elif else except exec finally*etc.

**Identifiers**

A Python Identifier is a name given to a function, class, variable, module, or other objects that we will be using in our Python program. Any entity we will be using in Python should be appropriately named or identified as they will form part of the program. Python naming conventions are:

i) An identifier can be a combination of uppercase letters, lowercase letters, underscores, and digits (0-9). Hence, the following are valid identifiers: myClass, my\_variable, var\_1, and print\_hello\_world.

ii) Special characters such as %, @, and $ are not allowed within identifiers. An identifier should not begin with a number. Hence, 2variable is not valid, but variable2 is acceptable.

iii) Python is a case-sensitive language and this behavior extends to identifiers. Thus, Labor and labor are two distinct identifiers in Python.

iv) We cannot use Python keywords as identifiers.

v) Class identifiers begin with an uppercase letter, but the rest of the identifiers begin in lowercase.

vi) We can use underscores to separate multiple words in your identifier.

**Literals**

Generally, literals are **a** notation for representing a fixed value in program**.**. They can also be defined as raw value or data given in variables or constants. The following numbers and literals are recognized by Python

* 1. Octal literals (base 8) To indicate an octal number, you will use the prefix 0o or 0O (zero followed by either a lowercase or uppercase letter ‘o’).
	2. Hexadecimal literals (base 16) To indicate hexadecimal literals, you will use the prefix ‘0X’ or ‘0x” (zero and uppercase or lowercase letter ‘x’).
	3. Binary literals (base 2) To signify binary literals, you’ll use the prefix ‘0B’ or ‘0b’ (zero and uppercase or lowercase ‘b’).

**Strings**

 A string is a sequence of Unicode characters that may be a combination of letters, numbers, and special symbols. To define a string in Python, you can enclose the string in matching single or double quotes:

>>>string1 = “I am enclosed in single quotes.” >>>string2 = “I am enclosed in double quotes.”

**Operators**

 Python operators allow programmers to manipulate data or operands. Here are the types of operators supported by Python:

* 1. **Arithmetic Operators**

|  |  |
| --- | --- |
| + | Addition adds the value of the left and right operands |
| - | Subtraction subtracts the value of the right operand from the value of the left operand |
| \* | Multiplication multiplies the value of the left and right operand |
| / | Division divides the value of the left operand by the right operand |
| \*\* | Exponent performs exponential calculation |
| % | Modulus returns the remainder after dividing the left operand with the right operand |
| // | Floor Division division of operands where the solution is a quotient left after removing decimal numbers |

2**.Assignment Operators**:

 These operators are useful when assigning values to variables:

|  |  |
| --- | --- |
| = | assigns the value of the right operand to the left operand |
| += add and | adds the value of the right and left operand and assigns the total to the left operand |
| -= subtract and | deducts the value of the right operand from the value of the left operand and assigns the new value to the left operand |
| \*= multiply and | multiplies the left and right operand and assigns the product to the left operand |
| /= divide and | divides the left operand with the value of the right operand and assigns the quotient to the left operand |
| \*\*= exponent | performs exponential operation on the left operand and assigns the result to the left operand |
| //= floor division and | performs floor division on the left operand and assigns the result to the left operand |

**Relational or Comparison Operators**

 Relational operators evaluate values on the left and right side of the operator and return the relation as either True or False.

|  |  |
| --- | --- |
| == | is equal to |
| < | is less than |
| > | is greater than |
| <= | is less than or equal to |
| >= | is greater than or equal to |
| != | is not equal to |
|  |  |

**Logical Operators**

 Python supports 3 logical operators:

**or** : x or y If the first argument, x, is false, then it evaluates the second argument, y. Else, it evaluates x

**and** : x and y If x is false, then it evaluates x. Else, if x is true, it evaluates y

**not**.. not x If x is false, then it returns True. If x is true, it returns False.

**Identity Operators**

 The identity operators **is** and **is not** can be used to test for None:

>>> a = [11, None, 'abc', None, {}]

 >>> a [11, None, 'abc', None, {}]

>>> count = 0

>>> for item in a: ...     if item is None: ...         count += 1 ...

>>>

 >>> print count

Bitwise Operators

**Membership Operators**

|  |  |  |
| --- | --- | --- |
| in | Returns True if a sequence with the specified value is present in the object | x in y |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y |

Data Type:

 Python handles several data types to facilitate the needs of programmers and application developers for workable data. These include strings, numbers, Booleans, lists, date, and time.

* + 1. **String**: - A string is a sequence of Unicode characters that may be a combination of letters, numbers, and special symbols. To define a string in Python, you can enclose the string in matching single or double quotes:

>>>string1 = “I am enclosed in single quotes.”

 >>>string2 = “I am enclosed in double quotes.”

* + 1. **Number**: One of the many conveniences of using Python is that we don’t really have to declare a numeric value to distinguish its type. Python can readily tell one data type from another when we write and run our statement. It has four built-in numeric data types. Python 3 supports three types: integer, floating-point numbers, and complex numbers. Long integers (‘long’) no longer form a separate group of integers but are included in the ‘int’ or integer category.

**Integer** (int) Integers are whole numbers without decimal point. They can be positive or negative as long as they don’t contain a decimal point that would make a number a floating number, a distinct numeric type.

**Floating-point numbers** Also known as floats, floating-point numbers signify real numbers. Floats are written with a decimal point that segregates the integer from the fractional numbers. They may also be written in scientific notation where the uppercase or lowercase letter ‘e’ signifies the 10 th power: >>>6.2e3 6200.0

**Complex numbers**: Complex numbers are pairs of real and imaginary numbers. They take the form ‘a + bJ’ where ‘a’ is a float and the real part of the complex number. On the other side is bJ where ‘b’ is a float and J or its lowercase indicates the square root of an imaginary number, -1. This makes ‘b’ the imaginary part of the complex number. Here are examples of complex numbers at work:

 >>>a = 2 + 5j

**Date and Time** Most applications require date and time information to make it work efficiently and effectively. In Python, you can use the function datetime.now() to retrieve the current date and time. The command datetime.now() calls on a built-in Python code which gives the current date and time. To get the date and time from Python, encode the following on the command prompt:

>>> from datetime import datetime

>>> datetime.now()

datetime.datetime(2016, 3, 10, 2, 16, 19, 962429)

**Boolean Data Type** Comparisons in Python can only generate one of two possible responses: True or False. These data types are called Boolean. Example

bool\_1 = 4 == 2\*3

bool\_2 = 10 < 2 \* 2\*\*3

bool\_3 = 8 > 2 \* 4 + 1

print(bool\_1)

print(bool\_2)

print(bool\_3)

The Python Shell will display these results: False True False

**Lists** A list is a data type that can be used to store any type and number of variables and information. We can define and assign items to a list with the expression:

my\_list = [item\_1, item\_2, item\_3]

Python also allows creation of an empty list: my\_list = []

Create a list of colors: colors = [“red”, “orange”, “yellow”, “green”, “indigo”, “white”] Since this is an indexed list, the first item on colors has zero as its index. To access the first item on the list, you can print the color with the command:

>>> print(colors[0])

 red

**UNIT 7: Creating Python Programs**

**Input** **Function**

 Programs usually require input that can come from different sources: keyboard, mouse clicks, database, another computer’s storage, or the internet. Since the keyboard is the most common way to gather input, Python provided its users the input() function. This function has an optional parameter called the prompt string. Once the input function is called, the prompt string will be displayed on the screen and the program flow stops until the user has entered an input. The input is then interpreted and the input() function returns the user’s input as a string.

 Here is a sample program that collects keyboard input for name and age:

name = input(“May I know your name? “)

print(“It’s a pleasure to meet you ” + name + “!”)

age = input(“Your age, please? “)

print(“So, you’re ” + age + ” years old, ” + name + “!”)

**Output Function:**

 The print() Function Python 3 turned print from a statement into a function. We must always enclose our print parameters within the round parentheses. Examples: print(“This is Python 3 print function) print(s) print(5) The print() function can print any number of values within the parentheses; they must be separated by commas. For example: a = 3.14 b = “age” c = 32 print(“a = “, a, b, c)

**Control statements**

 Control Statements are used to perform actions or calculations based on whether a condition is evaluated as true or false. If-then-else statements or conditional expressions are essential features of programming languages and they make programs more useful to users. The if-then-else statement in Python has the following basic structure:

if condition1:

block1\_statement

elif condition2:

 block2\_statament

else:

block3\_statement

This structure will be evaluated as: If condition1 is True, Python will execute block1\_statement. If condition1 is False, condition2 will be executed. If condition2 is evaluated as True, block2\_statement will be executed. If condition2 turns out to be False, Python will execute block3\_statement.

**Loop**:

A loop is a programming construct that enables repetitive processing of a sequence of statements. Python provides two types of loops to its users: the ‘for loop’ and the ‘while loop’. The ‘for’ and ‘while’ loops are iteration statements that allow a block of code (the body of the loop) to be repeated a number of times.

**The For Loop**

 Python implements an iterator-based ‘for loop’. It is a type of ‘for loop’ that iterates over a list of items through an explicit or implicit iterator. The loop is introduced by the keyword ‘for’ which is followed by a random variable name which will contain the values supplied by the object. This is the syntax of Python’s ‘for loop’:

for variable in list:

statements

else:

statements

**While Loop**

while (Control Statement):

statements

statements

**do while Loop**

do

 Statement

while (Control Statement):

statements

**Break and Continue Statement**:

  Python break and continue are used inside the loop to change the flow of the loop from its standard procedure

**Break Statement**

 Break statement is used to terminate the loop while it control statement is still true.

 .

while (Control Statement):

if (control statement):

 break

statements

**Continue Statement:**

 The **continue** statement skips the code that comes after it, and the control is passed back to the start for the next iteration.

while (Control Statement):

if (control statement):

 continue:

statements

**UNIT 8: Structures (10 Lectures)**

Numbers

Strings

Lists

Tuples,

Dictionary,

Date & Time,

**Modules**

 In Python, **Modules** are simply files with the “.py” extension containing Python code that can be imported inside another Python Program. In simple terms, we can consider a module to be the same as a code library or a file that contains a set of functions that we want to include in our application. With the help of modules, we can organize related functions, classes, or any code block in the same file. So, It is considered a best practice while writing bigger codes for production-level projects in Data Science is to split the large Python code blocks into modulescontaining up to 300–400 lines of code.

 The module contains the following components:

* Definitions and implementation of classes,
* Variables, and
* Functions that can be used inside another program.

## Create Python Modules

To create a module, we have to save the code that we wish in a file with the file extension **“.py”**. Then, the name of the Python file becomes the name of the module.

## Use Python Modules

To incorporate the module into our program, we will use the **import keyword,** and to get only a few or specific methods or functions from a module, we use the **from keyword.**

## Advantages of Modules

Some of the advantages while working with modules in Python is as follows:

### Reusability

Working with modules makes the code reusable.

### Simplicity

The module focuses on a small proportion of the problem, rather than focusing on the entire problem.

### Scoping

A separate namespace is defined by a module that helps to avoid collisions between identifiers.

**Defining Functions**

 A **function** is simply a container for a few lines of code that perform a simple task. All programming languages have functions as a common feature. Functions allow developers to write blocks of code that perform specific and reusable tasks. A function can be executed as many times as the developer wants throughout their code.

In Python, there are two types of functions:

* Built-in functions
* User-defined functions

**Built-in functions** are those functions pre-installed in Python. Examples include:

* **print()**: to print an object to the terminal.
* **help()**: to ask for help.
* **min()**: to get the minimum value.

**User-defined functions** are those functions defined by the user. Examples include:

* def greet\_customer()
* def calculate()
* def take\_square()

They all are defined by the user to help them perform certain tasks when writing a program.

 Defining a function in Python simply means to create a function. Functions in Python are created and denoted using the keyword def, followed by a function name. For example, my\_function() followed by a set of parenthesis().

When a function is defined, every letter of the function must be in lowercase.

def my\_function():
  print("Hello from a function")

## Calling a Function

To call a function, use the function name followed by parenthesis:

def my\_function():
  print("Hello from a function")

**my\_function()**

## Arguments

Information can be passed into functions as arguments. Arguments are specified after the function name, inside the parentheses. We can add as many arguments as you want, just separate them with a comma. By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, we have to call the function with 2 arguments, not more, and not less. If we do not know how many arguments that will be passed into our function then we can add a \* before the parameter name in the function definition. This way the function will receive a tuple of arguments, and can access the items accordingly:

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function.

**Exit function**,

 The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

**Default arguments**

 A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

**UNIT 9: Introduction to Advanced Python**

## OOP in Python

Python is a great programming language that supports OOP. We can define a class with attributes and methods, which can be called. Python offers a number of benefits compared to other programming languages. It's a dynamic language, with high-level data types. It does not require the programmer to declare types of variables and arguments.

### How to create a class

We can use the class keyword to define a class, followed by the class name and a colon. Inside the class, an  \_\_init\_\_ method has to be defined with def. This is the initializer that can later use to instantiate objects.. \_\_init\_\_ must always be present! It takes one argument: self, which refers to the object itself. Inside the method, the pass keyword is used.

class MyClass :

 x=5

class ClassName :

 instructions

class Person:
   def \_\_init\_\_(self, name, age):
     self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)

### Instantiating objects

To instantiate an object, type the class name, followed by two brackets. You can assign this to a variable to keep track of the object.

Objects and Classes

Inheritance

Regular Expressions

Event Driven Programming,

GUI Programming

Program in Python

* + - 1. Enter two numbers and calculate summation

x = input("Type a number: ")
y = input("Type another number: ")

sum = int(x) + int(y)

print("The sum is: ", sum)

* + 1. Enter three numbers and calculate average

X = input(“Enter first number”)

Y = input(“Enter second number”)

Z = input (“Enter third number”)

avg = (X+Y+Z)/3

Print(“The average = “, avg)

* + 1. Write a python program to convert the given temperature from Fahrenheit to Celsius and vice versa
* c = float(input("Enter temperature in Celsius : "))
f = (c \* 9/5) + 32
print("%.2f Celsius is equal to %.2f Fahrenheit" %(c, f))

 f = float(input("Enter temperature in Fahrenheit : "))
 c = (f - 32) \* 5/9
 print("%.2f Fahrenheit is equal to %.2f Celsius" %(f, c))

4. Write a program to find out largest among three numbers

# Python program to find the largest number among the three input numbers

# take three numbers from user

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

num3 = float(input("Enter third number: "))

if (num1 > num2) and (num1 > num3):

 largest = num1

elif (num2 > num1) and (num2 > num3):

 largest = num2

else:

 largest = num3

print("The largest number is",largest)

1. Enter a number and calculate factorial

num = int(input("Enter a number: "))

factorial = 1

**if** num < 0:

     **print**(" Factorial does not exist for negative numbers")

**else**:

    **for** i **in** range(1,num + 1):

        factorial = factorial\*i

    **print**("The factorial of",num,"is",factorial)

1. Display Fibonacci series up to n terms

n\_terms = int(input ("How many terms the user wants to print? "))

n\_1 = 0

n\_2 = 1

count = 0

**if** n\_terms <= 0:

    **print** ("Please enter a positive integer, the given number is not valid")

**elif** n\_terms == 1:

    **print** ("The Fibonacci sequence of the numbers up to", n\_terms, ": ")

    **print**(n\_1)

**else**:

    **print** ("The fibonacci sequence of the numbers is:")

    **while** count < n\_terms:

        **print**(n\_1)

        nth = n\_1 + n\_2

        n\_1 = n\_2

        n\_2 = nth

        count += 1

1. Enter a number and display whether it is prime number or not

 number = int(input("Enter any number: "))

 p=1

 if number > 1:

 for i in range(2, number):

 if (number % i) == 0:

 p=0

 break

 if (p == 1):

 print(number, "is a prime number")

 else:

 print(number, "is not a prime number")

 else:

 print(number, "is not a prime number")