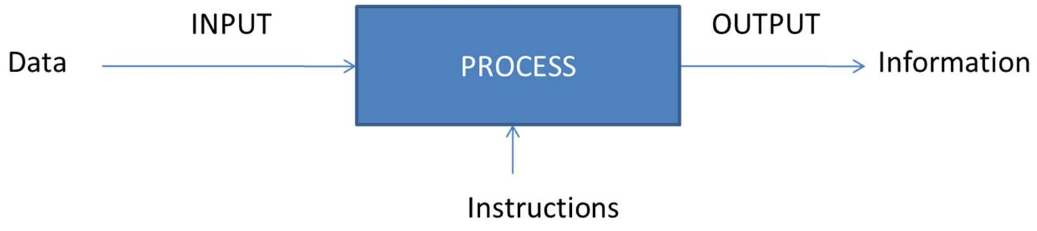
UNIT-I
Basics of Python Programming

Introduction to Computers

What is Computer?

The term computer is derived from the term **"compute".** Computer is a programmable electronic device that takes data and instruction as an input from the user and, process data, and provides useful information.



What is Data?

- Data is unprocessed facts and figures without any added interpretation or analysis.
- For e.g. "The price of crude oil is $80 per barrel".

What is Information?

- Information is data that has been interpreted so that it has meaning for the user.
- For e.g. "The price of crude oil has risen from $70 to $80 per barrel".

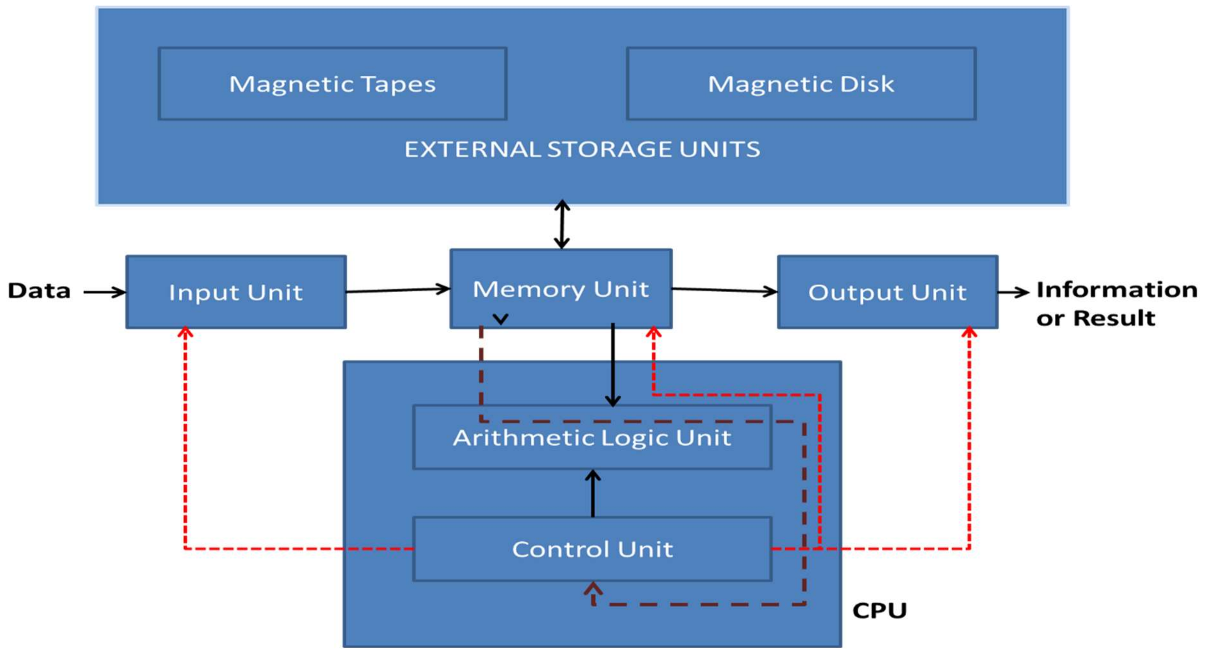
Characteristics of Computers

Automation, Speed, Accuracy, Diligence, Versatility, Storage, No Feelings and No I.Q.

Applications of Computers

- Word Processing, Internet, Digital Audio/Video Compression, Desktop Publishing, Traffic Control, Retail Business, Hospitals, Business and Industry, Weather Forecasting, Education, Online Banking, Robotics, Expert Systems.

Block Diagram of Digital Computers



Hardware

- Input Devices
- Output Devices
- Memory units
- CPU: ALU and CU
- BUS: Data and address Bus

Input Devices

Keyboard, Mouse, Scanner, Touch Screen, Joystick, Bar Code Reader, OMR (Optical Mark Recognition) and MICR (Magnetic ink character recognition)

Output Devices

Monitors, printers, speakers, projectors, and plotters

Memory

- Computer memory is any physical device capable of storing information temporarily or permanently.
- The memory is divided into large number of small parts called cells. Each location or cell has a unique address which varies from zero to memory size minus one.
- Units of Memory
  - bit (Binary Digits)- 0/1
  - Byte
    - 1 Byte = 8 bits
    - 1 KB = 1024 Bytes
    - 1 MB = 1024 KB
    - 1 GB = 1024 MB

Types of Memory

- Primary Memory
- Secondary Memory
- Cache Memory

Primary Memory

- Primary memory holds only those data and instructions on which computer is currently working.
- It is generally made up of semiconductor device.
- These memories are not as fast as registers.
- The data and instruction required to be processed reside in main memory.
- It is divided into two subcategories:
  - Random Access Memory (RAM)
  - Read Only Memory (ROM)

Random Access Memory (RAM)

- Is a volatile memory.
- Hold data temporarily as it requires continuous flow of electrical energy.
- Work with CPU to hold instructions and data to be processed.
- RAM is of two types.
  - Static RAM (SRAM)
  - Dynamic RAM (DRAM)

Static RAM (SRAM)

- The memory retains its contents if power is being supplied.

- Data is lost when the power gets down due to volatile nature.
- SRAM chips use a matrix of 6-transistors and no capacitors.
- Uses Transistors, so SRAM need not have to be refreshed on a regular basis.

Dynamic RAM (DRAM)

- DRAM, unlike SRAM, must be continually refreshed in order to maintain the data.
- DRAM is used for most system memory because it is cheap and small.
- Made up of memory cells which are composed of one capacitor and one transistor.

Read Only Memory (ROM)

- ROM stands for Read Only Memory.
- The memory from which we can only read but cannot write on it.
- This type of memory is non-volatile.
- The information is stored permanently in such memories during manufacture.
- A ROM stores such instructions that are required to start a computer.

ROM is of 4 types.

- MROM
- PROM
- EPROM
- EEPROM

MROM (Masked ROM)

- The very first ROMs were hard-wired devices that contained a pre-programmed set of data or instructions.
- These kinds of ROMs are known as masked ROMs which are inexpensive.

PROM (Programmable Read only Memory)

- PROM is read-only memory that can be modified only once by a user.
- The user buys a blank PROM and enters the desired contents using a PROM program.
- Inside the PROM chip there are small fuses which are burnt open during programming.
- Can be programmed only once and is not erasable.

EPROM (Erasable and Programmable Read Only Memory)

- The EPROM can be erased by exposing it to ultra-violet light for duration of up to 40 minutes.
- Usually, an EPROM eraser achieves this function.
- During programming, an electrical charge is trapped in an insulated gate region.
- The charge is retained for more than ten years because the charge has no leakage path.
- For erasing this charge, ultra-violet light is passed through a quartz crystal window (lid).
- This exposure to ultra-violet light dissipates the charge.

EEPROM (Electrically Erasable and Programmable Read Only Memory)

- The EEPROM is programmed and erased electrically.
- Can be erased and reprogrammed about ten thousand times.
- Both erasing and programming take about 4 to 10 ms (millisecond).
- In EEPROM, any location can be selectively erased and programmed.
- EEPROMs can be erased one byte at a time, rather than erasing the entire chip.
- The process of re-programming is flexible but slow.

Secondary Memory

- Hard Disk

- CD
- DVD
- USB Flash Drives
- Floppy Disk
- Magnetic Tape

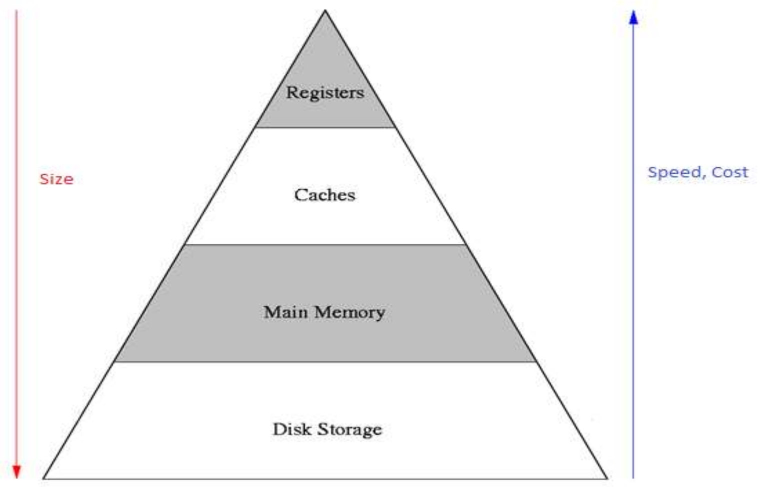
Characteristic of Secondary Memory

- These are magnetic and optical memories.
- Known as backup memory.
- Non-volatile memory.
- Data is permanently stored even if power is switched off.
- Used for storage of data in a computer.
- Computer may run without secondary memory.
- Slower than primary memories.

Cache Memory

- Cache memory is a very high-speed semiconductor memory which can speed up CPU.
- Acts as a buffer between the CPU and main memory.
- Used to hold those parts of data and program which are most frequently used by CPU.
- Parts of data and programs are transferred from disk to cache memory by operating system, from where CPU can access them.

Memory Hierarchy



Computer and its classification

Computers may be classified based on the following: -

1. Operating principles (based on their construction and working)
2. Applications
3. Size and capability (or classification into micro, mini, mainframe and supercomputers)
4. Number of Microprocessors
5. Word length and
6. Number of users

1. **Classification based on Operating Principles.**

A. Digital Computers

Operate essentially by counting. All quantities are expressed as discrete or numbers. Digital computers are useful for evaluating arithmetic expressions and manipulations of data (such as preparation of bills, ledgers, solution of simultaneous equations etc).

B. Analog Computers

An **analog computer** is a form of computer that uses the continuously changeable aspects of physical phenomena such as electrical, mechanical, or hydraulic quantities to model the problem being solved.

C. Hybrid Computers

Hybrid computers exhibit features of analog and digital computers. The digital component normally serves as the controller and provides logical operations, while the analog component normally serves as a solver of differential equations.

2. **Classification based on area of applications.**

A. Special Purpose Computers

A special purpose computer is designed only to meet the requirements of a particular task or application. The instructions needed to perform a particular task are permanently stored into the internal memory, so that it can perform the given task on a single command.

B. General Purpose Computers

A General-Purpose computer are designed to meet the needs of many different applications. In these computers, the instructions needed to perform a particular task are wired permanently into the internal memory. When one job is over, instructions for another job can be loaded into the internal memory for processing. This, a general-purpose machine can be used to prepare pay-bills, manage inventories, print sales report and so on.

3. **Classification digital Computer based on size and Capability.**

A. Microcomputers (Personal Computer)

A microcomputer is the smallest general purpose processing system. The older pc started 8 bit processor with speed of 3.7MB and current pc 64 bit processor with speed of 4.66 GB. Examples **IBM PC**s, **APPLE** computers

Microcomputer can be classified into 2 types.

1. Desktops
2. Portables

The difference is portables can be used while travelling whereas desktops computers cannot be carried around.

The different portable computers are.

Laptop, Notebooks, Palmtop (handheld), Wearable computers.

B. Minicomputer: A minicomputer is a medium-sized computer. That is more powerful than a microcomputer. These computers are usually designed to serve multiple users simultaneously (Parallel Processing). They are more expensive than microcomputers. Examples: Digital Alpha, Sun Ultra.

C. Mainframe computers: - Computers with large storage capacities and very high speed of processing (compared to mini- or microcomputers) are known as mainframe computers. They support many terminals for simultaneous use by several users like ATM transactions. They are also used as central host computers in distributed data processing system. Examples: - **IBM 370, S/390.**

D. Supercomputer: Supercomputers have extremely large storage capacity and computing speeds which are many times faster than other computers. A supercomputer speed is measured in terms of floating-point operations per second (flops), an operation is made up of numerous instructions. The supercomputer is mainly used for large scale numerical problems in scientific and engineering disciplines such as Weather analysis. Examples: **IBM Deep Blue.**

4. Classification based on number of microprocessors.

A. Sequential computers

Any task complete in sequential computers is with one microcomputer only. Most of the computers (today) we see are sequential computers where in any task is completed sequentially instruction after instruction from the beginning to the end.

B. Parallel computers

The parallel computer is relatively fast. New types of computers that use a large number of processors. The processors perform different tasks independently and simultaneously thus improving the speed of execution of complex programs dramatically. Parallel computers match the speed of supercomputers at a fraction of the cost.

5. Classification based on word-length.

A binary digit is called **"BIT"**. A word is a group of bits which is fixed for a computer. The number of bits in a word (or word length) determines the representation of all characters in these many bits. Word length leis in the range from 16-bit to 64-bitsf or most computers of today.

6. Classification based on number of users.

A. **Single User**: - Only one user can use the resource at any time.
B. **Multiuser**: - A single computer shared by several users at any time.
C. **Network**: - Several interconnected autonomous computers shared by several users at any time.

Algorithms

- The word "Algorithm" is derived from the Persian mathematician Muhammad ibn Musa al-Khwarizmi.
- An algorithm is a step-by-step description of how to arrive at solution of problem.
- It is the well-defined computational procedure that takes some or set of values as input, process it and produces some or set of values as output.

Characteristics of an Algorithm

- **Input:** It takes zero or input.
- **Output:** It produces one or more output.
- **Finiteness (Termination):** It should terminate after finite number of steps.
- **Definiteness:** Each instruction must be precise and clear. Its instruction must be unambiguous.

- **Effectiveness:** Each instruction must be very basic so that it can be carried out using pencil and paper.
  Example
- Algorithm to find sum of two numbers.
  1. Begin
  2. Input the Value of A and B.
  3. SUM = A+B.
  4. Display SUM.
  5. End.

Flow-Chart

- A flow-chart is a pictorial representation of an algorithm or process.
- Help the viewers to visualize the logic of the process.
- Often used by programmer as a program-planning tool fro organizing a sequence of steps necessary to solve a problem by a computer.

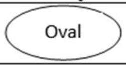
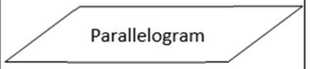
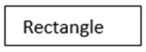
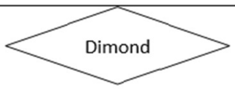
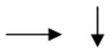
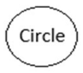
Advantage

- Better Communication, Proper program documentation, Efficient coding, Systematic debugging and testing.

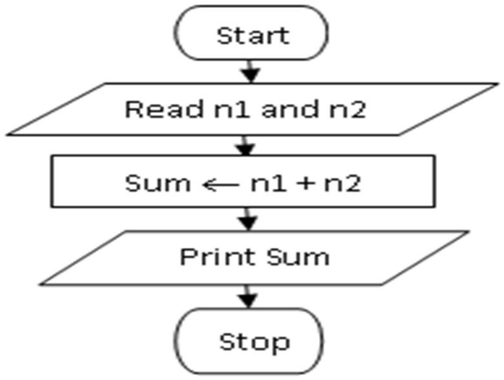
Limitations

- Time consuming and laborious to draw flowchart for complex problem.
- Any change in program logic will usually require a completely new flowchart.
- No standard for determining the amount of detail that should be included in flowchart.

Flow-Chart Symbols

S.No.	Name	Symbols	Meanings
i.	Start/Stop		It indicates the beginning and ending of the flowchart.
ii.	Input/output		It indicates the input/output operations.
iii.	Processing		It indicates the calculation or manipulate of data.
iv.	Decision		It indicates the decision making and branching.
v.	Flow lines		It indicates the direction of flow of instruction.
vi.	Connector		It joins one part of the flowchart with another part.

Example
Flow-Chart to find sum of two numbers.



Questions for Practice

1. Write the algorithm and draw the flowchart to compute the sum of two numbers.
2. Write the algorithm and draw the flowchart to compute the sum and average of five numbers.
3. Write the algorithm and draw the flowchart to compute the area of triangle using heron's formula.
4. Write the algorithm and draw the flowchart to compute the temperature in degree Fahrenheit when temperature in degree Celsius is given.
5. Write the algorithm and draw the flowchart to swap two numbers using 3rd variable.
6. Write the algorithm and draw the flowchart to swap two numbers without using 3rd variable.
7. Write the algorithm and draw the flowchart to check whether the number is odd or even.
8. Write the algorithm and draw the flowchart to compute the greater of two numbers.
9. Write the algorithm and draw the flowchart to compute the greatest of three numbers.
10. Write the algorithm and draw the flowchart to check whether the given year is leap year or not.
11. Write the algorithm and draw the flowchart to compute the sum of first N natural numbers.
12. Write the algorithm and draw the flowchart to compute the factorial of the given number.
13. Write the algorithm and draw the flowchart to compute the sum of digits of the given number.
14. Write the algorithm and draw the flowchart to compute the reverse of the given number. Also check the given number is in palindrome or not.
15. Write the algorithm and draw the flowchart to convert the decimal number to binary number.
16. Write the algorithm and draw the flowchart to convert the binary number to decimal number.

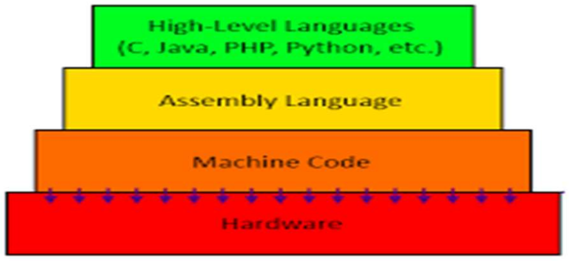
Programming Languages

Computer Program

- In computing, a program is a specific set of ordered operations performed by a computer to achieve a specific task.
- A *computer programming language* is a language used to write computer programs, which involve a computer performing computation or algorithm.

Programming Languages

- Machine Language
- Assembly Language
- High level Languages



Machine Language

- **Machine language** is a collection of binary digits or bits that the computer reads and interprets.
- It is a low-level language.
- Instructions written in the form of 0 and 1.
- Machine language is the only language a computer is capable of understanding.
- Sometimes referred to as **machine code** or **object code.**
- Communicate directly with the computer.

Advantages

- Computation speed is very fast.
- Directly understandable by computer.

Disadvantages

- Writing a program is very time consuming.
- Error correction is a tedious process.

Assembly Language

- It is also a low-level language.
- Written in the form of symbolic codes (mnemonics).
- Symbolic codes are like- ADD, SUB, MUL, DIV, LOAD, STORE, etc.
- Each assembly language is specific to a particular computer architecture.
- Assembly language is converted into executable machine code by a utility program referred to as an *assembler.*

Advantages

- Easier to understand as compared to machine code.
- Easy to locate and correct errors.

Disadvantages

- Like machine language it is also machine dependent.
- Programmer should have knowledge of hardware.

High Level Language

- A **high-level language** is a programming **language** that enables a programmer to write programs that are independent of a particular type of computer.
- It is much closer to human language.
- It uses English words and/or mathematical notations.
- Examples: COBOL, FORTRAN, PASCAL, C, C++, JAVA etc.

Advantages

- User-friendly.
- Easier to learn.
- Require less time to code.
- Easier to maintain.

Disadvantages

- Slower than low level language.
- Needs a translator.

Generations of Programming Languages

- 1GL- Machine language.
- 2GL- Assembly language.
- 3GL- High level language (C, C++, JAVA, BASIC) (Procedural/object-oriented)
- 4GL- Structured Query languages (SQL).
- 5GL- Logic languages (LISP).

4-GL

- Fourth generation languages are also known as very high-level languages.
- They are non-procedural languages.
- The code comprising instructions are written in English-like sentences.
- Code is easier to maintain.
- Enhance the productivity of the programmer.
- Examples SQL (Structured Query Language).

5-GL

- These are centered on solving problems using constraints given to the problem, rather than using an algorithm written by a programmer.
- Widely used in AI and research.
- Enables the computer to solve the problem without the programmer.
- Contains visual tools to help in developing a program.
- Examples are LISP, Prolog, OPS5.

Translators

- Used to convert one programming language to another.
- Types of Translators:
  - Assembler
  - Compiler
  - Interpreter

Compiler

- It is a special type of program that translate the source code written in high level language (source language) into the low-level language (target language).

- The resultant code can be assembly code or the object code. It is used to create an executable program.
- It locates and reports syntax error in the program (if any). But it cannot fix the error by itself.

Assembler

- It is a special program that translate the code written in assembly language into an equivalent code in machine language.
- The result is the object file that can be executed.

Interpreter

- An ***interpreter*** is a computer program that directly executes without previously compiling them into a machine language program.
- It converts High level language into low level.

- Interpreter is one which converts a source program and executes it at the same time.
- It translates line by line.

Difference between Interpreter and Compiler

Interpreter	Compiler
Translated program one statement at a time	Scans the entire program and translates it into machine code
Takes less amount of time to analyze the source code but overall execution time is slower	Takes large amount of time to analyze the source code but overall execution time is comparatively faster.
No intermediate code generated	Generate intermediate code
Requires less memory	Requires more memory
Debugging is easy	Debugging is comparatively hard

Linker
- Linker is a program that takes one or more objects generated by compiler and combines them into a single executable code.

Loader
- Loader is an operating system module that loads files from secondary storage to main memory.
- A program that takes an input an executable program, loads it into main memory, and causes execution by loading the correct starting address into the computer register.

Code of Ethics and Professional Conduct

The Code is designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way.

1. GENERAL ETHICAL PRINCIPLES.

A computing professional should...

1.1 Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.
1.2 Avoid harm.
1.3 Be honest and trustworthy.
1.4 Be fair and act not to discriminate.
1.5 Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.
1.6 Respect privacy.

2. PROFESSIONAL RESPONSIBILITIES.

A computing professional should…

2.1 Strive to achieve high quality in both the processes and products of professional work.

2.2 Maintain high standards of professional competence, conduct, and ethical practice.
2.3 Know and respect existing rules pertaining to professional work.
2.4 Accept and provide appropriate professional review.
2.5 Perform work only in areas of competence.
2.6 Foster public awareness and understanding of computing, related technologies, and their consequences.
2.7 Access computing and communication resources only when authorized or when compelled by the public good.
2.8 Design and implement systems that are robustly and usably secure.

3. PROFESSIONAL LEADERSHIP PRINCIPLES.
3.1 Ensure that the public good is the central concern during all professional computing work.
3.2 Manage personnel and resources to enhance the quality of working life.
3.4 Create opportunities for members of the organization or group to grow as professionals.
4. COMPLIANCE WITH THE CODE.
A computing professional should...

4.1 Uphold, promote, and respect the principles of the Code.
Programming Code of Ethics
Key points of proper conduct for Computer Programmers-
A programmer must...

- never create or distribute malware.
- never write code that is intentionally difficult to follow.
- never write documentation that is intentionally confusing or inaccurate.
- never reuse copyrighted code unless the proper license is purchased, or permission is obtained.
- acknowledge (verbally and in source code comments) the work of other programmers on which the code is based, even if substantial changes are made.
- never intentionally introduce bugs with the intent of later claiming credit for fixing the bugs, or to stimulate the uptake of later versions.
- never write code that intentionally breaks another programmer's code for the purpose of elevating one's status.
- never hide known obstacles to a project's completion during any phase of development, especially the design phase.
- report any illegal activities of the employer.
- never falsely deny the presence of bugs.
- never reveal the secret corporate knowledge of an employer.
- never accept compensation from multiple parties for the same work unless permission is given.
- never conceal from the employer their financial interest in development resources.
- never maliciously injure the reputation of an employer or members of the development team.
- never take credit for another's work.
- never steal software, especially development tools.
- never install third-party applications without the user's permission.

IT Policy

Companies provides and maintains technological products, services and facilities like Personal Computers (PCs), peripheral equipment, servers, telephones, Internet and application software to its employees for official use. The Information Technology (IT) Policy of the organization defines rules, regulations and guidelines for proper usage and maintenance of these technological assets to ensure their ethical and acceptable use and assure health, safety and security of data, products, facilities as well as the people using them. It also provides guidelines for issues like purchase, compliance, IT support and grievance redressal of the employees pertaining to technological assets and services used for office work.

- Acceptable Use Policy
- Security awareness
- DR/BCP (Disaster Recovery, Business Continuity plan)
- Change management.
- Equipment Usage policy
- PC standards
- The Internet Usage Policy
- Information security Policy
- Email and chat Policy
- The Software Usage Policy

Object Oriented Programming

Object-oriented Programming is a programming paradigm which provides a means of structuring programs so that properties and behaviors are bundled into individual objects.

Features of Object-Oriented Programming

Class

- The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods.
- For example:
    - An employee class contains attributes and methods, i.e. an email id, name, age, salary, etc.

Object

- The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.
- Everything in Python is an object, and almost everything has attributes and methods.
- All functions have a built-in attribute __doc__, which returns the doc string defined in the function source code.

Encapsulation

- Encapsulation is also an important aspect of object-oriented programming.
- It is used to restrict access to methods and variables.
- In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Polymorphism

- Polymorphism contains two words "poly" and "morphs". Poly means many and Morphs means form, shape.
- It means that one task can be performed in different ways.
- For example
  - In a class animal, all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in the sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

Inheritance

- Inheritance is the most important aspect of object-oriented programming which simulates the real-world concept of inheritance.
- It allows to create a class which uses all the properties and behavior of another class.
- The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.
- It provides re-usability of the code.

Abstraction

- Data abstraction and encapsulation both are often used as synonyms.
- Both are nearly synonym because data abstraction is achieved through encapsulation.
- Abstraction is used to hide internal details and show only functionalities.
- Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

What is Python Language?

- Python is a high-level general-purpose, interpreted, interactive, object-oriented, and reliable language having wide range of applications from Web development, scientific and mathematical computing to desktop graphical user Interfaces.
- The syntax of the language is clean, and length of the code is relatively short.
- It allows to think about the problem rather than focusing on the syntax.

A Brief History of Python

- Python is an old language created by **Guido Van Rossum.**
- The design began in the late 1980s and was first released in February 1991.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
- Latest version of Python is Python 3.9.1 with documentation released on Oct. 5, 2020.

Features of Python Language

- A simple language which is easier to learn.

- Free and open source.
- Portability.
- Extensible and Embeddable
  - easily combine pieces of C/C++ or other languages with Python code.
- A high-level, interpreted language.
- Large standard libraries to solve common tasks.
- Object-oriented
  - Everything in Python is an object.
  - Object oriented programming (OOP) helps to solve a complex problem intuitively.
  - Structure supports such concepts as polymorphism, operation overloading, and multiple inheritance.
- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It supports automatic garbage collection.
- **Scalable**
  - Python provides a better structure and support for large programs than shell scripting.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.
- **Databases**
  - Python provides interfaces to all major commercial databases.

Getting Python

- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python
  - https://www.python.org/

- Python documentation can be downloaded from
  - https://www.python.org/doc/

Application Areas of Python

1. Web Development

Python offers numerous options for web development. For instance, Django, Pyramid, Flask, and Bottle are used for developing web frameworks and even advanced content management systems like Plone and Django CMS. These web frameworks are packed with standard libraries and modules which simplify tasks like content management, database interaction, and interfacing with internet protocols.

2. Game Development

Libraries like PySoy (a 3D game engine that supports Python 3) and PyGame are two Python-based libraries used widely for game development. Python is the foundation for popular games like Battlefield 2, Frets on Fire, World of Tanks, Disney's Toontown Online, Vega Strike, and Civilization-IV.

3. Scientific and Numeric Applications

Thanks to its massive library base, Python has become a crucial tool in scientific and numeric computing. In fact, Python provides the skeleton for applications that deal with computation and scientific data processing.

4. Artificial Intelligence and Machine Learning

AI/ML applications require a language that is stable, secure, flexible, and is equipped with tools that can handle the various unique requirements of such projects. Python has all these qualities, and hence, it has become one of the most favored languages of Data Science professionals.

Python's simplicity, consistency, platform independence, great collection of resourceful libraries, and an active community make it the perfect tool for developing AI and ML applications.

5. Desktop GUI

Python not only boasts of an English-like syntax, but it also features a modular architecture and the ability to work on multiple operating systems. These aspects, combined with its rich text processing tools, make Python an excellent choice for developing desktop-based GUI applications.

6. Software Development

Python packages and applications aim to simplify the process of software development. From developing complex applications that involve scientific and numeric computing to developing desktop and web applications, Python can do it all. This is the reason why Software Developers use Python as a support language for build control, testing, and management.

7. Enterprise-level/Business Applications

Python high performance, scalability, flexibility, and readability are just the features required for developing fully functional and efficient business applications.

8. Education programs and training courses

It has a short learning curve and hence, is an excellent choice for beginners. Python's easy learning curve and simplicity are the two main reasons why it is one of the most used programming languages in educational programs, both at beginner and advanced levels.

9. Language Development

Python's design and module architecture has been the inspiration behind the development of many new programming languages such as Boo, Swift, CoffeeScript, Cobra, and OCaml.

10. Operating Systems

Linux-based Ubuntu's Ubiquity Installer and Fedora and Red Hat Enterprise's Anaconda Installer are coded in Python. Even Gentoo Linux leverages Python Portage (package management system). Usually, Python is combined with the C programming language to design and develop operating systems.

11. Web Scraping Applications

Python is a nifty tool for extracting voluminous amounts of data from websites and web pages. The pulled data is generally used in different real-world processes, including job listings, price comparison, R&D, etc.

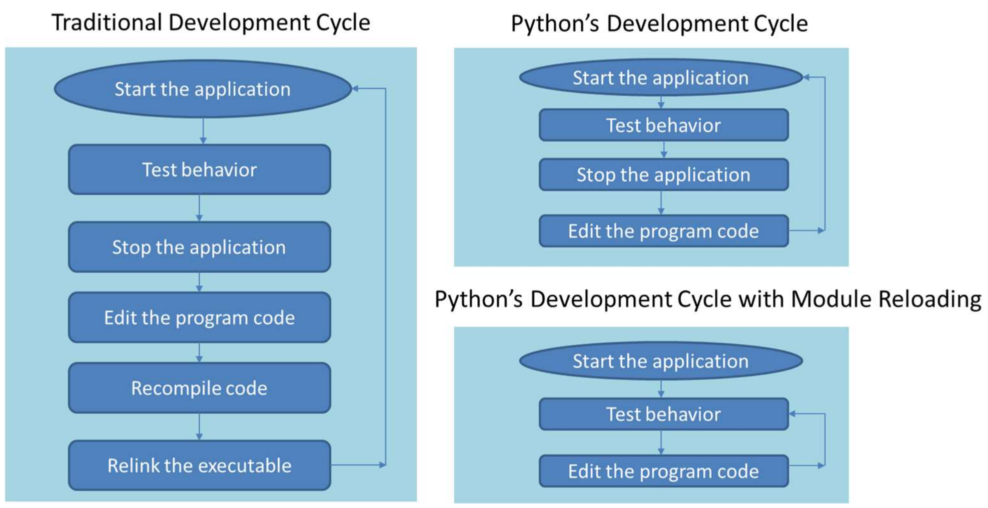
BeautifulSoup, MechanicalSoup, Scrapy, LXML, Python Requests, Selenium, and Urllib are some of the best Python-based web scraping tools.

12. Image Processing and Graphic Design Applications:

The programming language is used globally to design and build 2D imaging software like Inkscape, GIMP, Paint Shop Pro, and Scribus. Also, Python is used in several 3D animation packages such as Blender, Houdini, 3ds Max, Maya, Cinema 4D, and Lightwave, to name a few.

The Programming Cycle for Python

- Python's development cycle is dramatically shorter than that of traditional languages.
- In Python, there are no compile or link steps.
- Python programs simply import modules at runtime and use the objects they contain.
- Python programs run immediately after changes are made.
- And in cases where dynamic module reloading can be used, it is even possible to change and reload parts of a running program without stopping it at all.



Python IDE

IDLE is Python's Integrated Development and Learning Environment.
IDLE has the following features:

- coded in 100% pure Python, using the tkinter GUI toolkit.
- cross-platform works mostly the same on Windows, Unix, and macOS
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features.
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces.
- configuration, browsers, and other dialogs

Menus
IDLE has two main window types, the Shell window, and the Editor window. It is possible to have multiple editor windows simultaneously. On Windows and Linux, each has its own top menu. Each menu documented below indicates which window type it is associated with.

Output windows, such as used for Edit => Find in Files, are a subtype of editor window. They currently have the same top menu but a different default title and context menu.

Example of some IDE of Python

- **Spyder**
  - https://www.spyder-ide.org/
- **IDLE**
  - https://docs.python.org/3/library/idle.html
- **Sublime Text 3**
  - https://www.sublimetext.com/3
- **Jupyter**
  - https://jupyter.org/install.html

Interacting with Python Programs

1. Download Python IDE.
2. Run the installer to install Python on the computer.
3. Go to: File > New. Then save the file with .py extension.
4. For example, hello.py
5. Write Python code in the file and save it.
6. Then Go to Run > Run 'hello' or simply click F5 to run it.

Elements of Python

- Keywords and Identifiers
- Variables
- Data types and type conversion
- Operators in Python
- Expressions in Python

Keywords

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- All the keywords except True, False and None are in lowercase and they must be written as they are.
- Example
  - and, as, assert, await, break, class, continue, def, del, elif, else, except, finally, for, global, if, import, in, is, lambda, not, or, pass, raise, return, try, while, with, yield

Identifiers

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.

Rules for writing Identifiers.

- Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.
- An identifier cannot start with a digit.

- Keywords cannot be used as identifiers.
- Cannot use special symbols like !, @, #, $, % etc. in identifier.
- An identifier can be of any length.

Variables

- A variable is a location in memory used to store some data (value).
- Unique names are given to them to differentiate between different memory locations.
- No need to declare a variable before using it. The declaration happens automatically when a value is assigned to a variable.
- The Assignment operator (=) is used to assign values to variables.
- The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.
- For example

```
>>> num = 347           # An integer assignment
>>> x   = 45.89          # A floating point
>>> name   = “Aman"     # A string
```

Multiple Assignment in Variables

- A single value may be assigned to several variables simultaneously.
- For example
  `>>> a = b = c = 1`
- Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location.
- It also allows to assign multiple objects to multiple variables.
- For example −
  `>>> a, b, c = 23, 29.45, “Ram”`
- Here, two integer objects with values 23 and 29.45 are assigned to variables a and b respectively, and one string object with the value “Ram" is assigned to the variable c.

Data types

- Every value in Python has a datatype, that are used to define the operations possible on them and the storage method for each of them.
- Since everything is an object in Python programming, data types are classes and variables are instance (object) of these classes

Standard Data types

Data Types	**Keyword**
Text Type	Str
Numeric Types	int, float, complex
Sequence Types	list, tuple, range
Mapping Type	Dict
Set Types	Set
Boolean Types	Bool

Binary Types	Bytes

Numbers

- They store numeric values. Number objects are created when a value is assigned to them. For ex:

```
>>> var1 = 1
>>> var2 = 10
```

- They can be deleted the reference to a number object by using the del statement. The syntax of the del statement is

```
del var1[,var2[,var3[....,varN]]]]
```

- A single object or multiple objects can be deleted by using the del statement. For example

```
>>> del var
>>> del var_a, var_b
```

Types of Number Types

1. int (signed integers)
2. float (floating point real values)
3. complex (complex numbers)

Examples of Number Types

Int	Float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
0o80	32.3+e18	.876j
-0o490	-90	-.6545+0J
-0x260	-32.54E100	3e+26J

Strings

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.
- For example

```
>>> str = 'Hello World!'
>>> print(str)          # Prints complete string
>>> print(str[0])       # Prints first character of the string
>>> print(str[2:5])     # Prints characters starting from 3rd to 5th
>>> print(str[2:])      # Prints string starting from 3rd character
>>> print(str * 2)      # Prints string two times
>>> print(str + 'TEST') # Prints concatenated string
```

Lists

- A List is an ordered sequence of item, it contains items separated by commas and enclosed within square brackets ([]).

- To some extent, lists are like arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- Declaring a list is straight forward. Items separated by commas are enclosed within brackets [ ].

```
a = [1, 2.2, 'python']
```

  The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

```
>>>  list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
>>> tinylist = [123, 'john’]
>>> print(list)            # Prints complete list
>>> print(list[0])         # Prints first element of the list
>>> print(list[1:3])       # Prints elements starting from 2nd till 3rd
>>> print(list[2:])        # Prints elements starting from 3rd element
>>> print(list * 2)        # Prints list two times
>>> print(list + tinylist) # Prints concatenated lists
```

Tuple

- A tuple is like the list. A tuple consists of several values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- The only difference is that tuples are immutable. Tuples once created cannot be modified.
- Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically.

```
>>> tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
>>> tinytuple = (123, 'john’)
>>> print(tuple)               # Prints complete tuple
>>> print(tuple[0])            # Prints first element of the tuple
>>> print(tuple[1:3])          # Prints elements starting from 2nd till 3rd
>>> print(tuple[2:])           # Prints elements starting from 3rd element
>>> print(tuple * 2)           # Prints tuple two times
>>> print(tuple + tinytuple)   # Prints concatenated tuples
```

Set

- Set is an unordered collection of unique items.
- Set is defined by values separated by comma inside braces { }.
- Items in a set are not ordered.
- Set operations can be performed like union, intersection on two sets. Set have unique values. They eliminate duplicates.

- indexing has no meaning as set are unordered collection. Hence the slicing operator [] does not work.
- For example

```
>>>a = {1,2,2,3,3,3}
>>>type(a)
<class 'set'>
>>> a
{1, 2, 3}
```

Dictionary

- Dictionary is an unordered collection of key-value pairs.
- It is generally used when we have a huge amount of data.
- Dictionaries are optimized for retrieving data. It is must to know the key to retrieve the value.
- A dictionary key can be almost any Python type but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are defined within braces {} with each item being a pair in the form key: value. Key and value can be of any type.
- For example

```
>>> d = {1:'value','key':2}
>>> type(d)
<class 'dict'>
```

Type conversion

- The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.
- Python has two types of type conversion.
  - Implicit Type Conversion
  - Explicit Type Conversion

Implicit Type conversion

- Python automatically converts one data type to another data type without any user involvement.
- It always converts smaller data types to larger data types to avoid the loss of data.
- Example

```
>>> a = 4
>>> b = 2.3
>>> c = a + b
```

- In above example, data type of a and b are int and float, respectively. Data type of c will be float and its value is 6.3.

Explicit Type conversion

- In Explicit Type Conversion, users convert the data type of an object to required data type.
- The predefined functions like int (), float (), str () are used to perform explicit type conversion.

- This type of conversion is also called typecasting because the user casts/changes the data type of the objects.
- Syntax

  <datatype>(expression)
- Example

```
>>> a = 2.6
>>> c = int(a)
>>> c
2
```

Function of type conversion

Function	Description
int(x)	Convert x to an integer
float(x)	Convert x to a float number
str(x)	Convert x to a string
tuple(x)	Convert x to a tuple
list(x)	Convert x to a list
set(x)	Convert x to a set
ord(x)	Convert x to ASCII code
bin(x)	Convert x to a binary
oct(x)	Convert x to an octal
hex(x)	Convert x to a hexadecimal
chr(x)	Convert x to a character

Operators in Python

- Arithmetic Operators
- Relational Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operators

Operator	Description	Example (a=5, b=3)
+	Adds values on either side of the operator.	a + b = 8
-	Subtracts right hand operand from left hand operand.	a – b = 2
*	Multiplies values on either side of the operator	a * b = 15
/	Divides left hand operand by right hand operand	a / b = 1.6667
%	Divides left hand operand by right hand operand and	a % b = 2
**	Performs exponential (power) calculation on operators	a**b =5^3= 125
//	The division of operands where the result is the quotient in which the digits after the decimal point are removed.	a//b=1

Relational Operators

It is used to compare arithmetic expressions.

Operator	Description	Example a=5, b=3
==	If the values of two operands are equal, then the condition becomes true	a == b is False
!=	If values of two operands are not equal, then condition becomes true.	a!=b is True
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	a > b is True
<	If the value of left operand is less than the value of right operand, then condition becomes true.	a < b is False
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	a >= b is True
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	a <= b is False

Assignment Operators

Operator	**Description**
=	a = b Assigns values from right side operands(b) to left side operand (a)
+=	a+=b is same as a = a + b It adds right operand to the left operand and assign the result to left operand
-=	a-=b is same as a = a – b It subtracts right operand from the left operand and assign the result to left operand
*=	a*=b is same as a = a * b It multiplies right operand with the left operand and assign the result to left operand
/=	a/=b is same as a = a / b It divides left operand with the right operand and assign the result to left operand

%=	a%=b is same as a = a % b It takes modulus using two operands and assign the result to left operand
**=	a**=b is same as a = a ** b Performs exponential (power) calculation on operators and assign value to the left operand
//=	a//=b is same as a = a //b It performs floor division on operators and assign value to the left operand

Logical Operators

It is used to combine relational expressions.

Logical operators are the and, or, not operators.

Operator	Description
and	True if both the operands are true
or	True if either of the operands is true
not	True if operand is false (complements the operand)

a	b	a and b	a or b	not a
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Bitwise Operators

- Bitwise operators manipulate the data at bit level.
- These are applicable on integer values.
- Types
  - & (Bitwise and operator)
  - | (Bitwise or operator)
  - ^ (Bitwise XOR operator)
  - ~ (Bitwise one's complement operator)
  - << (Bitwise left-shift operator)
  - >> (Bitwise right-shift operator)

a	b	a & b	a | b	a ^ b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Operator	**Let a = $(92)_{10}$ =$(0101\ 1100)_2$ and b = $(14)_{10}$ = $(0000\ 1110)_2$**	
&	c = (a & b) = 92 & 14	0101 1100 & 0000 1110

		--------------------- 0000 1100 = $(12)_{10}$
I	c = (a | b) = 92 | 14	0101 1100 | 0000 1110 --------------------- 0101 1110 = $(94)_{10}$
^	c = (a ^ b) = 92 | 14	0101 1100 ^ 0000 1110 --------------------- 0101 0010 = $(82)_{10}$
~	c = ~a = ~92 [~n= - (n+1)]	c = ~(0101 1100) = 1010 0011
<<	c = a<< 1 = 46<< 1 [x<<y=x*2^y]	C = 00101 110 << 1 = 01011100 = $(92)_{10}$
>>	c = a >> 2 = 92>> 2 [x<<y=x//2^y]	C = 0101 1100 >> 2 = 0001 0111 = $(23)_{10}$

Membership Operators

- **in** and **not in** are the membership operators in Python.
- They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).
- In a dictionary, we can only test for presence of key, not the value.

Operator	Description	Example x = {2,3,5}	Result
in	True if value/variable is found in the sequence	5 in x	True
not in	True if value/variable is not found in the sequence	5 not in x	False

Identity Operators

- Identity operators compare the memory locations of two objects.
- **is** and **is not** are the identity operators in Python.
- They are used to check if two values (or variables) are located on the same part of the memory.
- Two variables that are equal does not imply that they are identical.

Operator	Description	Example a = 'Hello' b = 'Hello'	Result
is	True if the operands are identical (refer to the same object)	a is b	True
is not	True if the operands are not identical (do not refer to the same object)	a is not b	False

Operator Precedence and associativity

- When an expression contains two or more than two operators, then it is evaluated based on operator precedence and associativity.
- Each operator of same precedence is grouped in same level and operator of higher precedence is evaluated first.

- Two or more operators having equal precedence are evaluated either left-to-right(LTR) or right-to-left(RTL) based on their associativity.

Operator	Description
()	Parenthesis
**	Exponentiation
~,+,-	Unary operators
*,/,//,%	Arithmetic multiply, division, floor and modulo division
+, -	Addition and subtraction
>>,<<	Bitwise left and right shift operator
&	Bitwise and operator
^	Bitwise Ex-or operator
|	Bitwise or operator
<=,>=,<,>	Relational inequality operators
==, !=	Equal and not equal operators
=,*=,/=,//=,%=,+=,-=,**=,&=	Assignment operators
is, is not	Identity operators
in, not in	Membership operators
not	Logical not operator
and	Logical and operator
or	Logical or operator

Expressions in Python

- Expressions are representations of value.
- They are different from statement in the fact that statements do something while expressions are representation of value.
- Python expression contains identifiers, literals, and operators.
- For Example
    - x = a*b + c/d –f is an expression.

Daily Quiz

1. In Python, 'Hello', is the same as "Hello"
    a) True
    b) False

2. __________Operator is used to multiply numbers.

3. Which of the following is not a keyword in Python language?
   a) val
   b) raise
   c) try
   d) with

4. Which of the following declarations is incorrect in python language?
   a) xyzp = 5,000,000
   b) x y z p = 5000 6000 7000 8000
   c) x,y,z,p = 5000, 6000, 7000, 8000
   d) x_y_z_p = 5,000,000

5. Which of the following operators is the correct option for power(ab)?
   a) a ^ b
   b) a**b
   c) a ^ ^ b
   d) a ^ * b

6. Which of the following precedence order is correct in Python?
   a) Parentheses, Exponential, Multiplication, Division, Addition, Subtraction
   b) Multiplication, Division, Addition, Subtraction, Parentheses, Exponential
   c) Division, Multiplication, Addition, Subtraction, Parentheses, Exponential
   d) Exponential, Parentheses, Multiplication, Division, Addition, Subtraction

7. Which one of the following has the same precedence level?
   a) Division, Power, Multiplication, Addition and Subtraction
   b) Division and Multiplication
   c) Subtraction and Division
   d) Power and Division

8. What will be the output of the following function?
round(4.576)
   a) 4
   b) 5
   c) 576
   d) 5

9. What will be the output of the following code?
import math
abs(math.sqrt(36))
   a) Error
   b) -6
   c) 6
   d) 6.0

10. Which of the following is false statement in python
   a) int(144)==144
   b) int('144')==144
   c) int(144.0)==144
   d) None of the above

MCQs

1. What is the correct file extension for Python files?
   a) .pyt
   b) .py
   c) .pt
   d) .pyth
2. What is the output for −
   'python ' [-3]?
   a) 'o'
   b) 't'
   c) 'h'
   d) Negative index error.
3. How do you create a variable with the numeric value 5?
   a) x = 5
   b) x = int(5)
   c) Both option (a) and (b) are correct
   d) No option is correct
4. Which one is NOT a legal variable name?
   a) My_var
   b) My-var
   c) _Myvar
   d) Myvar
5. Which one is a legal identifier?
   a) ab#cd
   b) abc
   c) S.I
   d) Area of circle
6. Which operator can be used to compare two values?
   a) <>
   b) ><
   c) ==
   d) =
7. What error occurs when you execute the following Python code snippet?
   apple = mango
   a) SyntaxError
   b) NameError
   c) ValueError
   d) TypeError
8. Which among the following list of operators has the highest precedence?
   **, %, /, <<, >>, |
   a) <<, >>
   b) **
   c) |
   d) %, /

9. Which character is used in Python to make a single line comment?
   a) /
   b) //
   c) #
   d) !
10. Which of these collections defines a SET?
   a) {1, 2, 3, 4}
   b) {‘1’:’12’,’2’:’45’}
   c) (1,2,3,4)
   d) [1,2,3,4]

Old Question Papers

1. What is the difference between list and tuples? [AKTU 2019-2020(odd), 2 marks]
2. In some languages, every statement ends with semicolon (;). What happens if you put a semi-colon at the end of a Python statement? [AKTU 2019-2020(odd), 2 marks]
3. Mention five benefits of using Python. [AKTU 2019-2020(odd), 2 marks]
4. How is Python an interpreted language? [AKTU 2019-2020(odd), 2 marks]
5. What type of language is Python? [AKTU 2019-2020(odd), 2 marks]
6. Define floor division with example. [AKTU 2019-2020(odd), 2 marks]
7. Write short notes with example: The Programming cycle for Python, Elements of Python, Type conversion in Python, operator precedence and Boolean expression. [AKTU 2019-2020(odd), 10 marks]

Expected Questions for Exam

1. Define operator. Explain various types of operators in Python with suitable example.
2. Explain standard data types in Python.
3. Explain Programming cycle of Python.
4. Explain the concepts object-oriented programming?
5. Explain the type conversion in Python.