Module -1 (SDLC)

1. What is Software?

Software is a collection of instructions, data, or computer programs that are used to run machines and carry out particular activities. Applications, scripts, and other programs that operate on a device are collectively referred to as "software." It can be compared to the flexible portion of a computer.

Application software and system software are the two main subcategories of software. Applications are pieces of software that carry out tasks or address certain needs. The hardware of a computer is run by system software, which also acts as a platform for other software to run on. Other forms of software include middleware, which stands between system software and applications, driver software, which controls computer peripherals and devices, and programming software, which offers the programming tools needed by software developers.

2. What is Application?

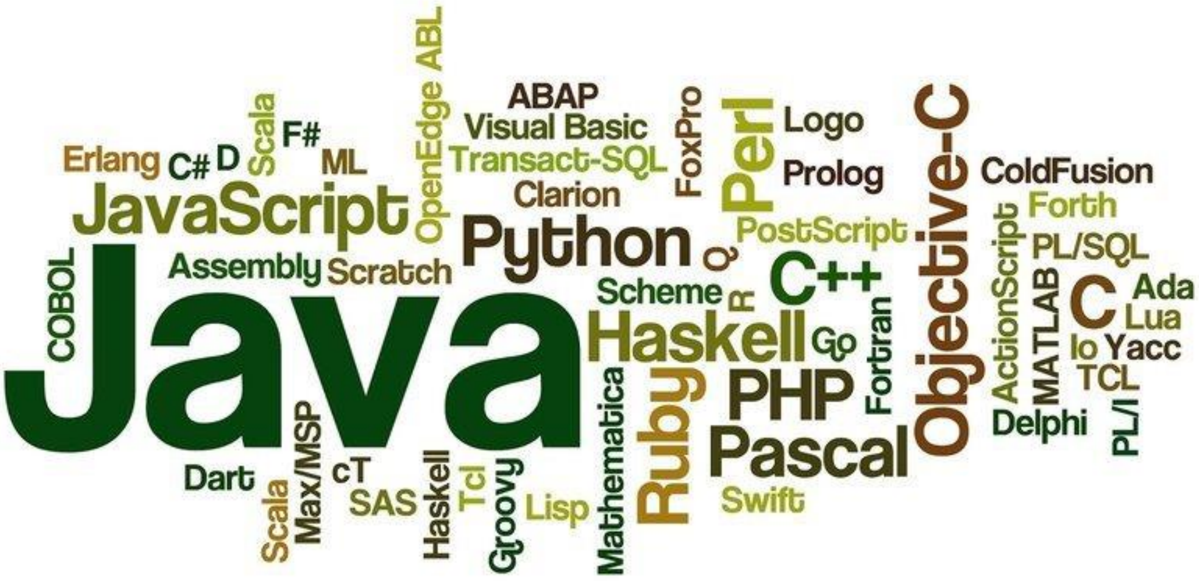
A computer software package known as an application, sometimes known as an application program or application software, carries out a specified task either directly for a user or, in certain situations, for another application. Applications might consist of a single program or a collection of programs. Applications run on computers using the operating system (OS) and other auxiliary software applications. An application-programming interface (API) allows a program to interact with and request services from other technologies.

Applications can differ in a number of ways, such as how they are created, the operating system they utilize, whether they are open source or proprietary, or even the market they serve. The examples of applications are as below:

- Word processors
- Database programs
- Web browsers
- Deployment tools
- Image editors
- Communication platforms

3. What is Programming?

A computer program is made up of code that is run by the computer to carry out specific tasks. Programmers wrote the code in this document. Giving machines a set of instructions outlining how a program should be executed is the process of programming.

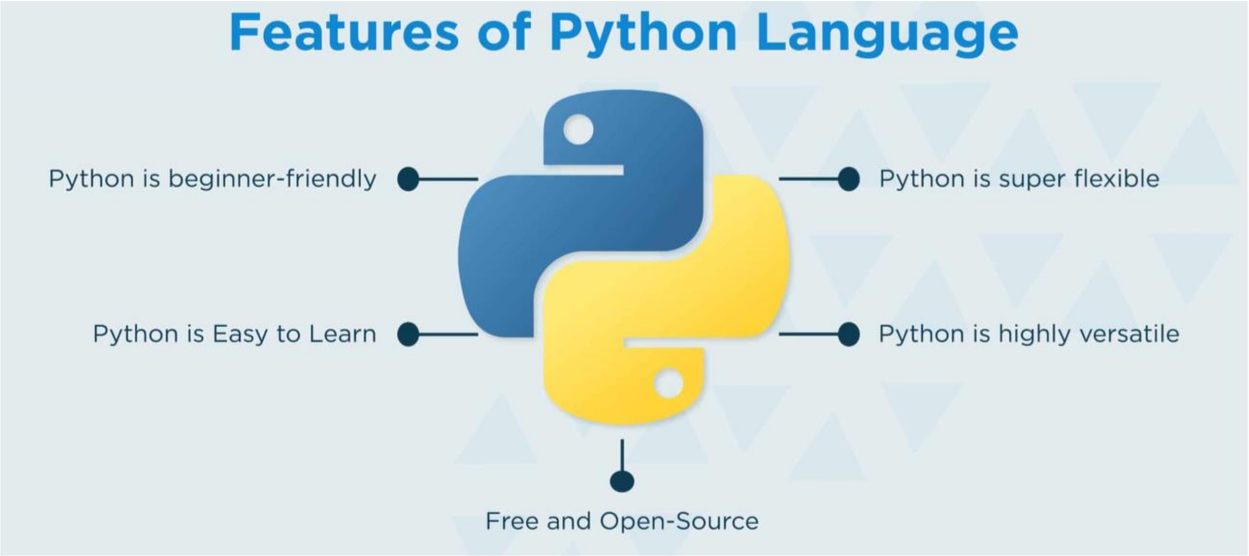


In the beginning, programmers will create source code using an IDE or code editor. This is a group of computer programs written in a language that other programmers can understand. In order for machines to comprehend the instructions and run the program, source code must be translated into machine language. Compiling is the process of translating source code into machine language. C and C++ are two examples of compiled programming languages. Other languages do not make use of compilers. These languages will instead employ an interpreter to read and run the code. JavaScript and PHP are two examples of interpreted programming languages.

The computer program can run after the code has been run. Websites, database systems, video games, and word processors are a few examples of the various kinds of computer programs. These computer programs let us to communicate with a variety of software-based products and services, such as phones, websites, and computers.

4. What is Python?

Python is an interpreted, object-oriented, high-level, dynamically semantic programming language. It is particularly desirable for Rapid Application Development as well as for usage as a scripting or glue language to tie existing components together due to its high-level built-in data structures, dynamic typing, and dynamic binding. Python's straightforward syntax prioritizes readability and makes it simple to learn, which lowers the cost of program maintenance. Python's support for modules and packages promotes the modularity and reuse of code in programs. For all popular platforms, the Python interpreter and the comprehensive standard library are freely distributable and available in source or binary form.


Features of Python Language
Python is beginner-friendly
Python is super flexible
Python is Easy to Learn
Python is highly versatile
Free and Open-Source

The edit-test-debug cycle is extraordinarily quick because there is no compilation step. Python programs are simple to debug since a bug or incorrect input never causes a segmentation failure. Instead, the interpreter raises an exception when it finds a mistake. The interpreter prints a stack trace if the application does not catch the exception. Setting breakpoints, evaluating arbitrary expressions, inspecting local and global variables, stepping through the code one line at a time, and other features are all possible with a source level debugger. Python's ability to perform introspection is demonstrated by the debugger, which is developed in Python. On the other hand, adding a few print statements to the source code is frequently the easiest way to debug a program due to the short edit-test-debug cycle.