

Python Fundamentals

day 1

Today's Agenda

- What is python?
- Why python is so famous?
- History of python
- Comparison with C, JAVA
- ALL, MLL, HLL
- Compiler vs Interpreter



What is python??

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming.

It is used for:

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

Why python is so famous and most recognized language?

First and foremost reason why Python is much popular because it is highly productive as compared to other programming languages like C++ and Java. It is much more concise and expressive language and requires less time, effort, and lines of code to perform the same operations.



The Python features like one-liners and dynamic type system allow developers to write very fewer lines of code for tasks that require more lines of code in other languages. This makes Python very easy-to-learn programming language even for beginners and newbies. For instance, Python programs are slower than Java, but they also take very less time to develop, as Python codes are 3 to 5 times shorter than Java codes.

Python is also very famous for its simple programming syntax, code readability and English-like commands that make coding in Python lot easier and efficient.

History of Python

Python was conceived in the late 1980s by **Guido van Rossum** at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system.



Python's name is derived from the **British comedy group Monty Python**, whom Python creator Guido van Rossum enjoyed while developing the language. Monty Python references appear frequently in Python code and culture; for example, the metasyntactic

variables often used in Python literature are *spam* and *eggs* instead of the traditional *foo* and *bar*. The official Python documentation also contains various references to Monty Python routines.

Comparison with C and JAVA

Let us consider an example of swapping of two numbers in all top three programming languages.

Let's SwAP it

Case 1 - C language

```
#include <stdio.h>

void main()
{
    int a = 10;
    int b = 20;
    int temp;

    printf("a = %d",a);
    printf("b = %d",b);
    temp = a;
    a = b;
    b = temp;
    printf("a = %d",a);
    printf("b = %d",b);
}
```



Case 2 - Java

```
class Test
{
    public static void main(String args[])
    {
        int a = 10;
        int b = 20;
        int temp;

        System.out.println("a = "+a);
        System.out.println("b = "+b);

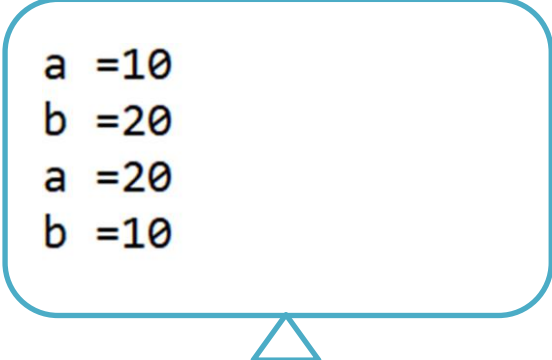
        temp = a;
        a = b;
        b = temp;

        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }
}
```

Case 3 - Python

```
a = 10;  
b = 20;  
print("a = ",a);  
print("b = ",b);  
b,a = a,b  
print("a = ",a);  
print("b = ",b);
```

Output-



```
a =10  
b =20  
a =20  
b =10
```

ALL, MLL, HLL

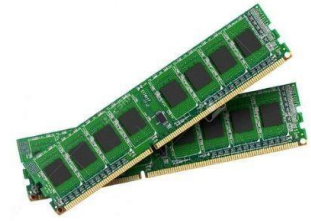
Before you learn any programming language, it is important for one to understand some of the basics about computer and what are the languages that a computer can understand.

Let's have a view on it.

A computer is a collection of hardware components. Let us consider here few hardware components such as:



microprocessor



RAM



Hard disk



GPU

Out of these and many other hardware components, the most important or the heart of the computer is the **Microprocessor or CPU**.

Microprocessor or CPU: A **microprocessor** is an electronic component that is used by a computer to do its work. It is a central processing unit on a single integrated circuit chip containing millions of very small components including transistors, resistors, and diodes that work together. They are created using a technology called as **Semiconductor technology**.

Semiconductor Technology??

Any device which is made up of transistors is referred to as working in Semiconductor Technology.

A **transistor** is a device that regulates current or voltage flow and acts as a switch or gate for electronic signals. The transistors have three terminals emitter, base and collector.

There are two types of transistors:

- 1) NPN transistor.
- 2) PNP transistor.



Transistors can only store voltages. There are two levels of voltages:

Low Voltage referred to as $\rightarrow 0V$

High Voltage referred to as $\rightarrow 5V$

If we see the same in Software engineer's view, he/she looks the two levels as:

Low Voltage referred to as $\rightarrow 0$

High Voltage referred to as $\rightarrow 1$

Therefore, in the perspective of a software engineer a Microprocessor or CPU can understand combinations of 0 and 1.

Programming Languages

Language is the main medium for communicating between the computer systems. A program is a collection of instructions that can be executed by a computer to perform a specific task.

There were several programming languages used to communicate with the Computer.

Case-1:

The world's first computer was invented in the year 1940's.

During that time the task of a programmer was not simple.

For example, if they wanted microprocessor to perform any operation then they had to use combinations of 0's and 1's.

During this time all the programs were written in the language called as **Machine Level Language**. It is one of the low-level programming languages.



Codes written in 1940's as

To perform addition of two numbers: 0110110

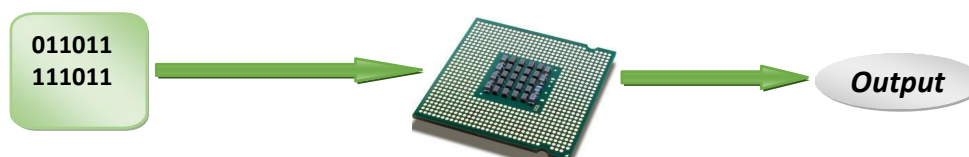
To perform subtraction of two numbers: 1110111

To perform multiplication of two numbers: 1010101

To perform division of two numbers: 0100100

MLL

microprocessor



The machine level code was taken as input and given to the microprocessor as the machine understands the binary value code and it gives the output.

The main **advantage** of using Machine language is that there is no need of a translator to translate the code, as the Computer directly can understand.

The **disadvantage** was, it was difficult for a programmer to write the code or remember the code in this type of language.

Case-2:

The problem with Machine level code approach was decided to be changed in the year 1950's.

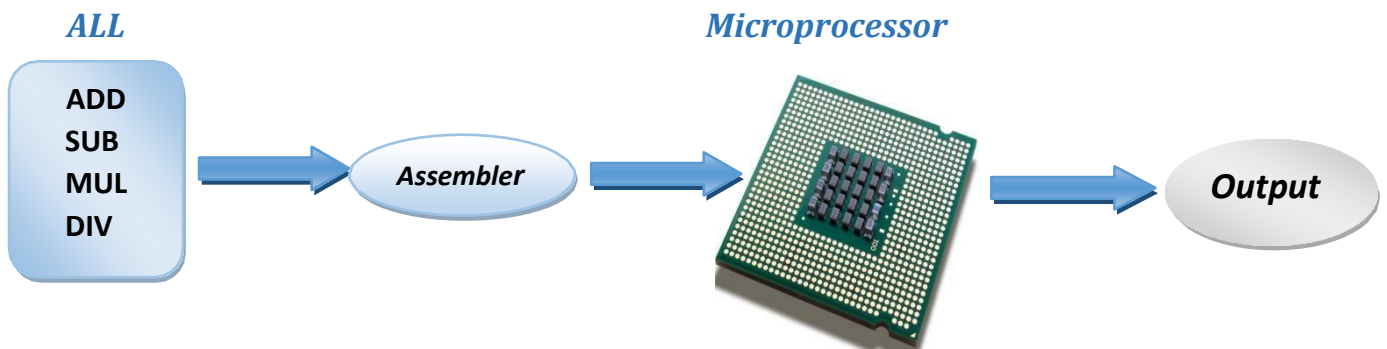
They thought that instead of writing a long sequence of 0's and 1's a **single instruction** can be given.

For example we use,

Codes in 1950's written as

To perform addition of two numbers:	ADD
To perform subtraction of two numbers:	SUB
To perform multiplication of two numbers:	MUL
To perform division of two numbers:	DIV

This approach of writing code is what called as **Assembly Level Language**. Instead of using numbers like in Machine languages here we use words or names in English forms.



An Assembler is software which takes Assembly Level Language (ALL) programs as input and converts it into Machine Level Language (MLL) program.

Case-3:

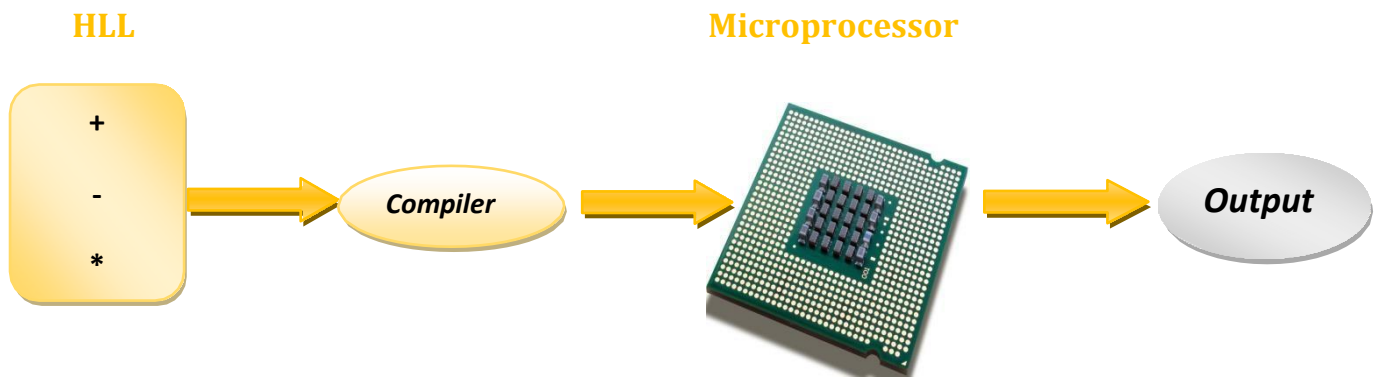
People always want the things to be simple and easier so, in 1960's they came up with next type of language called **High Level Programming Language**.

High Level Languages are written in a form that is close to our human language, enabling the programmer to just focus on the problem being solved.

For example we use,

Codes in 1960's written as

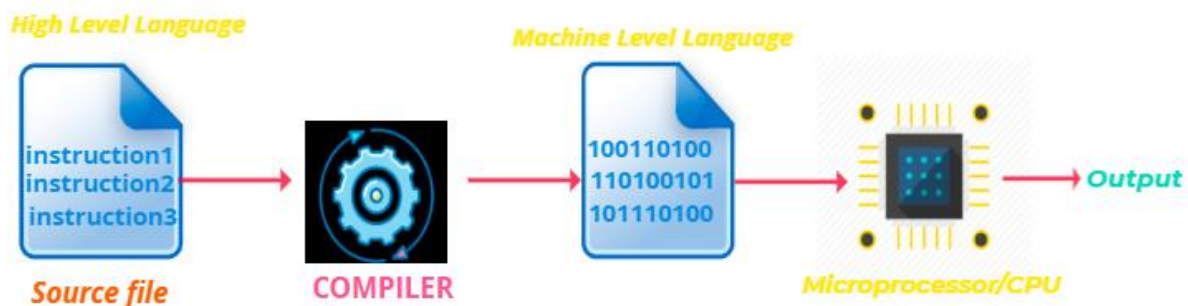
To perform addition of two numbers:	+
To perform subtraction of two numbers:	-
To perform multiplication of two numbers:	*
To perform division of two numbers:	/



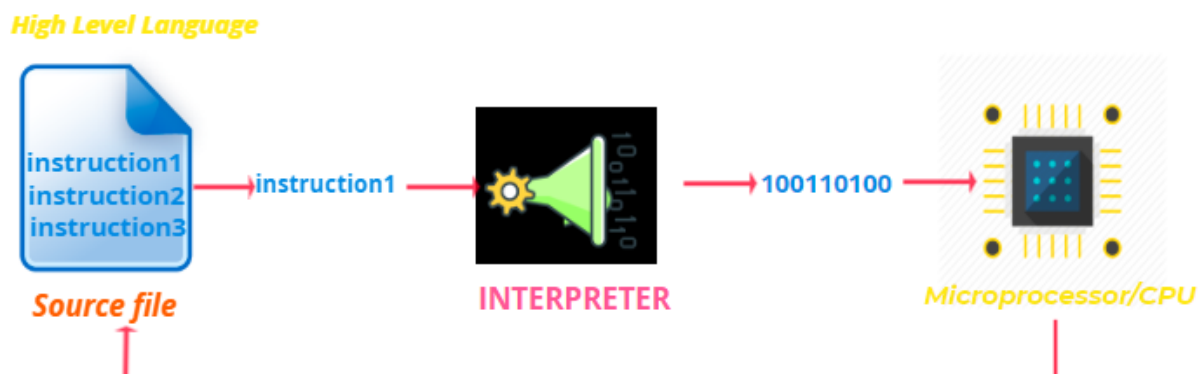
A compiler is software which takes High Level Language (HLL) programs as input and converts it into Machine Level Language (MLL) program.

Compiler vs Interpreter

Compilation



Interpretation



Interpreters and compilers are very similar in structure. The main difference is that compilation is the process of converting a HLL program into MLL using software called as compiler. In compilation the entire HLL program is converted to MLL in one shot by the compiler and hence all the MLL instructions are readily available for CPU.

Interpretation is the process of converting HLL program into MLL using software called as interpreter. In interpretation, at any given point of time only a single line of HLL program is converted to MLL which is then executed by CPU. Post execution the next line is converted and executed. This process repeat itself till all lines in the program are converted and executed. Because of this, CPU does not have all lines of program readily available for execution.

An interpreter will typically generate an efficient Machine Level representation and immediately evaluate it.

Difference between Compiler and Interpreter

No	Compiler	Interpreter
1	Compiler Takes Entire program as input	Interpreter Takes Single instruction as input .
2	Intermediate Object Code is Generated	No Intermediate Object Code is Generated
3	Conditional Control Statements are Executes faster	Conditional Control Statements are Executes slower
4	Memory Requirement : More (Since Object Code is Generated)	Memory Requirement is Less
5	Program need not be compiled every time	Every time higher level program is converted into lower level program
6	Errors are displayed after entire program is checked	Errors are displayed for every instruction interpreted (if any)