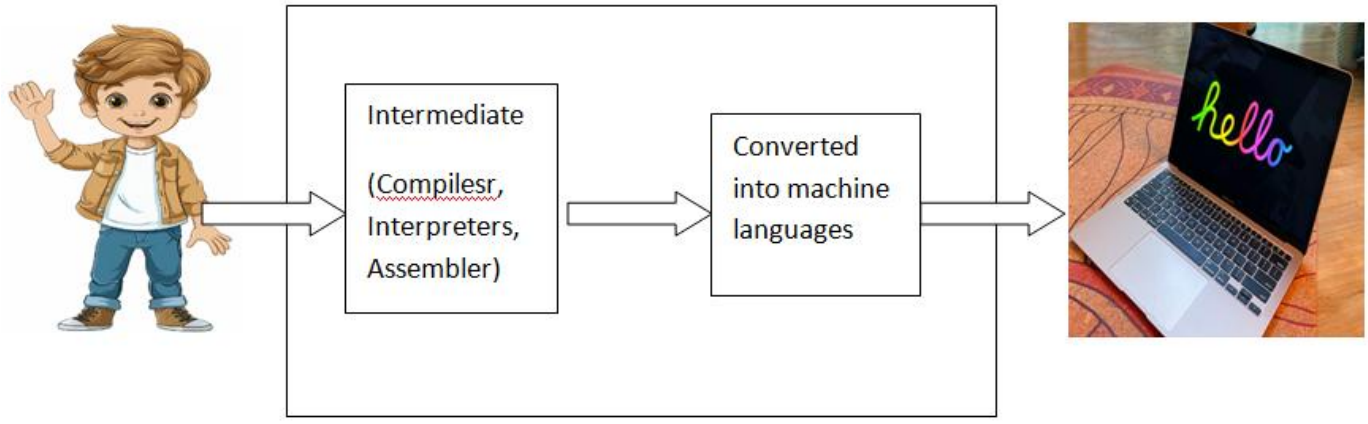
Language

Basics of languages

What is language ?
Why it is required ?
Type of languages ?
What is Low level languages (Machine & assembly language)?
What is high level languages ()?

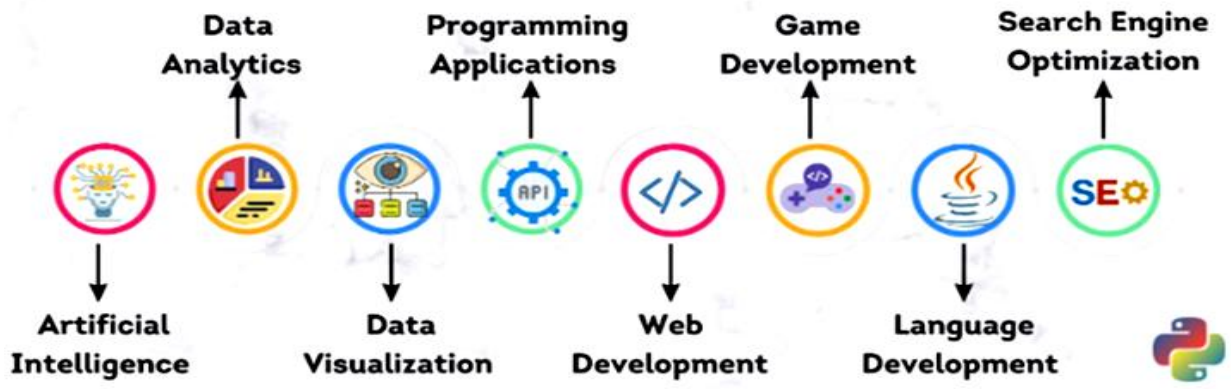


Python Introduction

Python is a :

1. Free and Open Source
2. General-purpose
3. High Level Programming language

That can be used for:



Features/Advantages of Python:

1. Simple and easy to learn
2. Procedure and object oriented
3. Platform Independent
4. Portable
5. Dynamically Typed
6. Both Procedure Oriented and Object Oriented
7. Interpreted
8. Vast Library Support

Syntex:----

Example 1:-

C:

```
#include<stdio.h>
void main()
{
        print("Hello world");
 }
```

Python:

```
print("Hello World")
```

Example 2:- To print the sum of 2 numbers

C:

```
#include <stdio.h>
void main()
{
        int a,b;
         a =10;
         b=20;
        printf("The Sum:%d",(a+b));
}
```

Python:

```
A,b=10,20
print("The Sum:",(a+b))
```

Limitations of Python:

1. **Performance and Speed:** Python is an interpreted language, which means that it is slower than compiled languages like C or Java. This can be a problem for certain types of applications that require high performance, such as real-time systems or heavy computation.
2. **Done Not have Support for Concurrency and Parallelism:** Python does not have built-in support for concurrency and parallelism. This can make it difficult to write programs that take advantage of multiple cores or processors.
3. **Static Typing:** Python is a dynamically typed language, which means that the type of a variable is not checked at compile time. This can lead to errors at runtime.
4. **Web Support:** Python does not have built-in support for web development. This means that programmers need to use third-party frameworks and libraries to develop web applications in Python
5. **Runtime Errors**

Python can take almost all programming features from different languages:--

1. Functional Programming Features from C
2. Object Oriented Programming Features from C++
3. Scripting Language Features from Perl and Shell Script
4. Modular Programming Features from Modula-3(Programming Language)

Flavors of Python or types of python interpretors:

1. CPython:

It is the standard flavor of Python. It can be used to work with C lanugage Applications

2. Jython or JPython:

It is for Java Applications. It can run on JVM

3. IronPython:

It is for C#.Net platform

4. PyPy:

The main advantage of PyPy is performance will be improved because JIT (just in time)compiler is available inside PVM.

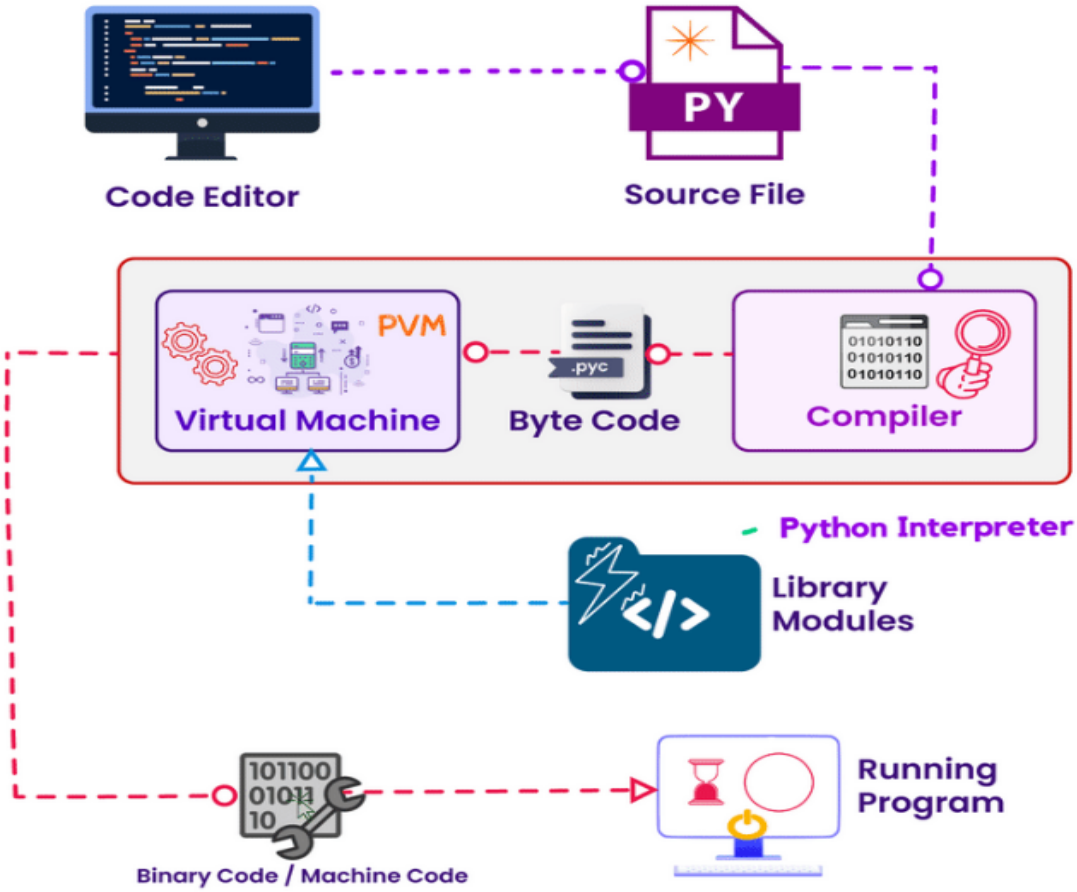
5. RubyPython

For Ruby Platforms

6. AnacondaPython

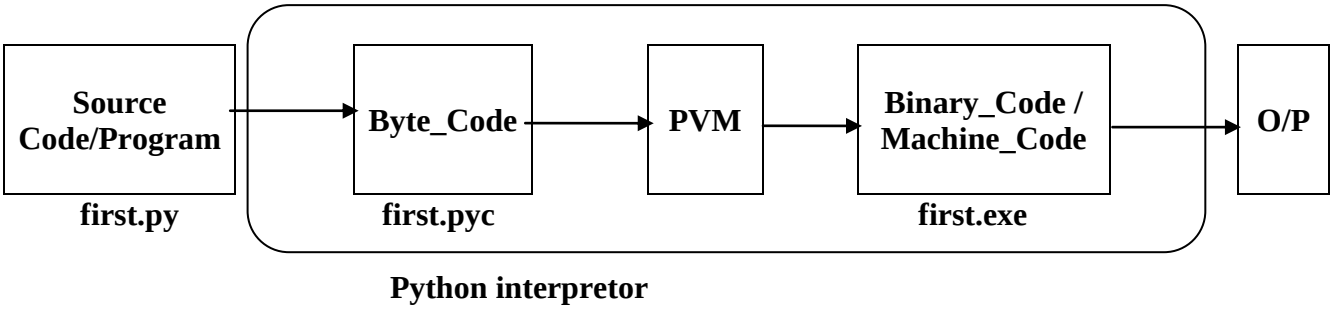
It is specially designed for handling large volume of data processing.

Python Internal working



Python is a high-level, interpreted programming language with a clear syntax, making it user-friendly and widely used in many domains. Here's a breakdown of

Each of these steps occurs behind the scenes, making Python a powerful and flexible language.



Examples:---
first.py:---

```
a = 10
b = 10
print("Sum ", (a+b))
```

The execution of the Python program involves 2 Steps:

- Compilation
- Interpreter

Compilation

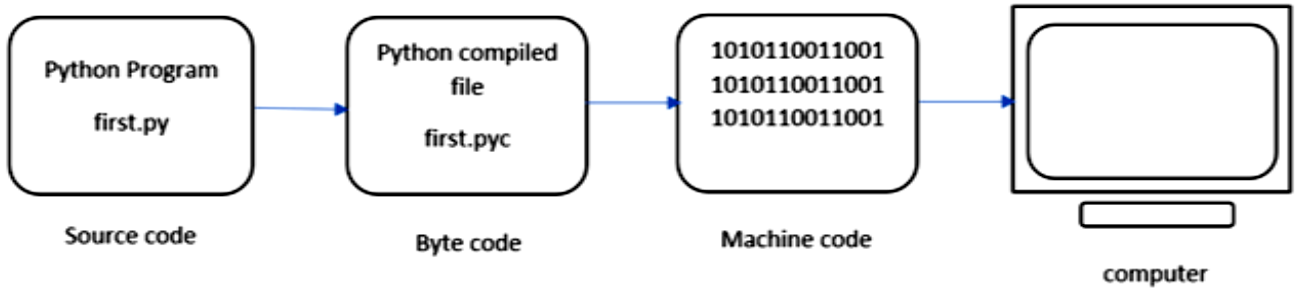
The program is converted into **byte code.** Byte code is a fixed set of instructions that represent arithmetic, comparison, memory operations, etc. It can run on any operating system and hardware. The byte code instructions are created in the **.pyc** file. The .pyc file is not explicitly created as Python handles it internally but it can be viewed with the following command:

```
PS E:\Python_data> python -m py_compile first.py
```

-m and py_compile represent module and module name respectively. This module is responsible to generate .pyc file. The compiler creates a directory named __pycache__ where it stores the first.cpython-310.pyc file.

Interpreter

The next step involves converting the byte code (.pyc file) into machine code. This step is necessary as the computer can understand only machine code (binary code). Python Virtual Machine (PVM) first understands the operating system and processor in the computer and then converts it into machine code. Further, these machine code instructions are executed by processor and the results are displayed.



However, the interpreter inside the PVM translates the program line by line thereby consuming a lot of time. To overcome this, a compiler known as Just In Time (JIT) is added to PVM. JIT compiler improves the execution speed of the Python program. This

compiler is not used in all Python environments like CPython which is standard Python software.

To execute the first.cpython-310.pyc we can use the following command:

```
PS E:\Python_data\__pycache__> python first1.cpython-310.pyc
```

view the byte code of the file – first.py we can type the following command as : first.py:

```
x = 10
y = 10
z=x+y
print(z)
```

The command **`python -m dis first.py`** disassembles the Python bytecode generated from the source code in the file `first.py`.

- **python**: This is the command to invoke the Python interpreter.
- **-m dis**: This uses Python's built-in dis module to disassemble the Python bytecode.
    - dis stands for **disassembler**. It translates Python bytecode back into a more readable form, showing the low-level instructions that the Python Virtual Machine (PVM) executes.
- **first.py**: This is the Python script file whose bytecode will be disassembled.

When you run this command, Python compiles `first.py` into bytecode (if not already compiled), and the `dis` module disassembles it. This helps you understand the internal bytecode instructions that Python generates from your source code.

```
PS C:\Users\neera\Desktop\online_class> py -m dis .\first.py
  0           0 RESUME                   0

  1           2 LOAD_CONST               0 (10)
              4 STORE_NAME               0 (x)

  2           6 LOAD_CONST               1 (20)
              8 STORE_NAME               1 (y)

  3          10 LOAD_NAME                0 (x)
             12 LOAD_NAME                1 (y)
             14 BINARY_OP                0 (+)
             18 STORE_NAME               2 (z)

  4          20 PUSH_NULL
             22 LOAD_NAME                3 (print)
             24 LOAD_NAME                2 (z)
             26 CALL                     1
             34 POP_TOP
             36 RETURN_CONST             2 (None)
PS C:\Users\neera\Desktop\online_class>
```

- **LOAD_CONST**: Loads a constant value (like numbers 10 and 20).
- **STORE_NAME**: Stores the value in a variable (like x, y, or z).
- **LOAD_NAME**: Loads the value of a variable from memory.
- **BINARY_ADD**: Adds two values (in this case, the values of x and y).
- **CALL_FUNCTION**: Calls a function (like print).
- **RETURN_VALUE**: Returns from a function (in this case, the main program).