**Internal working of Python**

**Python** is an object-oriented programming language like Java.

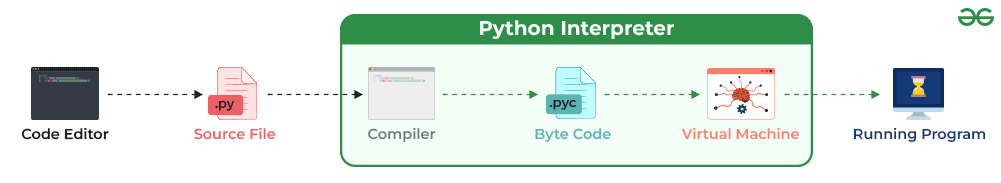
[Python](https://www.geeksforgeeks.org/python-programming-language/)is called an **interpreted language.**

Python uses code modules that are interchangeable instead of a single long list of instructions that was standard for functional programming languages.

The standard implementation of Python is called **“cpython”.** It is the default and widely used implementation of Python.

**Internal working of Python**

Python doesn’t convert its code into machine code, something that hardware can understand. It converts it into something called byte code. So within Python, compilation happens, but it’s just not in a machine language. It is into byte code (**.pyc or .pyo)** and this byte code can’t be understood by the CPU. So we need an interpreter called the Python virtual machine to execute the byte codes.



*Internal Working of Python*

**How is Python Source Code Converted into Executable Code**

The Python source code goes through the following to generate an executable code

* **Step 1:** The Python compiler reads a Python source code or instruction in the code editor. In this first stage, the execution of the code starts.
* **Step 2:** After writing Python code it is then saved as a .**py** file in our system. In this, there are instructions written by a Python script for the system.
* **Step 3: I**n this the compilation stage comes in which source code is converted into a byte code. Python compiler also checks the syntax error in this step and generates a .**pyc** file.
* **Step 4:** Byte code that is **.pyc** file is then sent to the Python Virtual Machine(PVM) which is the Python interpreter. PVM converts the Python byte code into machine-executable code and in this interpreter reads and executes the given file line by line. If an error occurs during this interpretation then the conversion is halted with an error message.
* **Step 5:** Within the PVM the bytecode is converted into machine code that is the binary language consisting of 0’s and 1’s. This binary language is only understandable by the CPU of the system as it is highly optimized for the machine code.
* **Step 6:**In the last step, the final execution occurs where the CPU executes the machine code and the final desired output will come as according to your program.

**How Python Internally Works?**

* **Code Editor:**Code Editor is the first stage of programs where we write our source code. This is human-readable code written according to Python’s syntax rules. It is where the execution of the program starts first.
* **Source code:**The code written by a programmer in the code editor is then saved as a .py file in a system. This file of Python is written in human-readable language that contains the instructions for the computer.
* **Compilation Stage:** The compilation stage of Python is different from any other programming language. Rather than compiling a source code directly into machine code. python compiles a source code into a byte code. In the compilation stage python compiler also checks for syntax errors. after checking all the syntax errors, if no such error is found then it generates a .pyc file that contains bytecode.
* **Python Virtual Machine(PVM):** The bytecode then goes into the main part of the conversion is the Python Virtual Machine(PVM). The PVM is the main runtime engine of Python. It is an interpreter that reads and executes the bytecode file, line by line. Here In the Python Virtual Machine translate the byte code into machine code which is the binary language consisting of 0s and 1s. The machine code is highly optimized for the machine it is running on. This binary language is only understandable by the CPU of a system.
* **Running Program:** At last, the CPU executes the given machine code and the main outcome of the program comes as performing task and computation you scripted at the beginning of the stage in your code editor.

**Python Libraries/Modules**

When you import libraries or modules in your Python program. Firstly python checks if the given module is built-in, and executes the corresponding C code. If the module is not built-in then the list of directories is defined in **sys. path**. the directory of the input script, and directories listed in the[**PYTHONPATH**](https://www.geeksforgeeks.org/pythonpath-environment-variable-in-python/). if a **.py** file corresponds to the modules imported, [Python](https://www.geeksforgeeks.org/python-programming-language/) creates a new module object, On executing the code in the **.py**file within the object’s namespace. Then Python compiles source code into byte code( the .**pyc** file), allowing for quicker execution

**Compiler Vs Interpreter**

In the system both the[compiler and interpreter](https://www.geeksforgeeks.org/compiler-vs-interpreter-2/) are the same they convert high-level code to machine code. The interpreter converts source code into the machine when the program runs in a system while a compiler converts the source code into machine code before the program runs in our system.

| **Compiler** | **Interpreter** |
| --- | --- |
| The compiler is faster, as conversion occurs before the program executes. | The interpreter runs slower as the execution occurs simultaneously. |
| Errors are detected during the compilation phase and displayed before the execution of a program. | Errors are identified and reported during the given actual runtime. |
| Compile code needs to be recompiled to run on different machines. | Interpreted code is more portable as it can run on any machine with the appropriate interpreter. |
| It requires more memory to translate the whole source code at once. | It requires less memory than compiled ones. |
| Debugging is more complex due to batch processing of the code. | Debugging is easier due to the line-by-line execution of a code. |