How Python Executes your Code

In this PyDive Article, we will learn how the Python Interpreter executes the Program.

When you run your Python program, the Python interpreter read instructions from your program and executes them. Behind the scene, your Python source code is first compiled and converted to *bytecode*. Each of your source statement is translated into a group of bytecode instructions. This *bytecode* is responsible for the speedy execution. However, this process has been kept completely hidden from you.

When your python source file is compiled, the python will store the *bytecode* of your source in a file with *.pyc* extension. In python 3.2 or later, python saves these *bytecode* files in a subdirectory named __pycache__ in your current directory. These *.pyc* files are named as *<yourscript>.cpython-<pythonversion>.pyc* convention where, *<pythonversion>* is the python version which created this file.

This python *bytecode* file creation removes the overhead of conversion of your python source code to *bytecode*. Python uses this already compiled existing *bytecode* to execute when the program is executed next time.

Automatic Checking of Changes

Python automatically checks for the changes in the source file of the *bytecode* by checking the time-stamps of the source file and the *bytecode* file. Through this comparison it will know when to compile the python source file again.

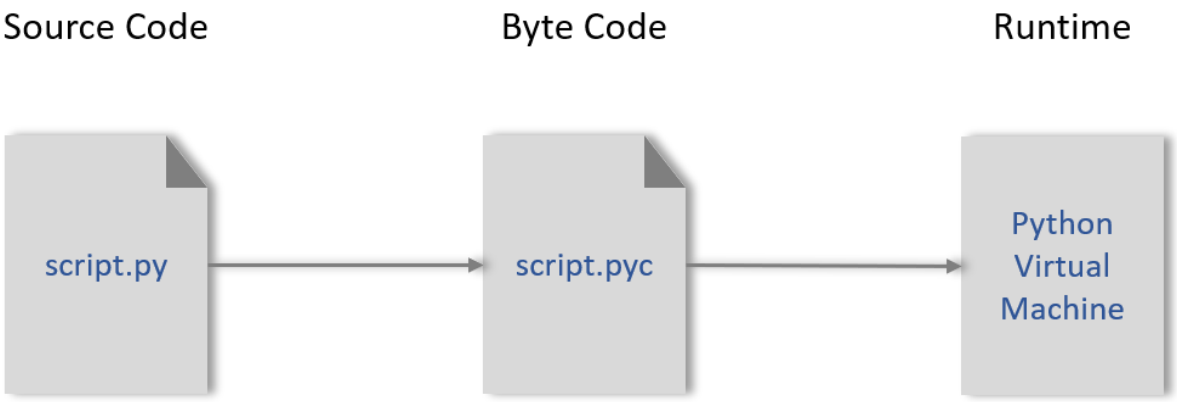
Also, python checks for the python version which created the *bytecode*. If the version differs from the one which is executing now the re-compilation takes place and new *bytecode* file is generated.

What if Python does not have 'write' Access

Well if Python does not have 'write' access to your system or working directory, still the program will run. The python will generate the *bytecode* in memory and later discards it when the program terminates.

Python Virtual Machine

The Python Virtual Machine is the runtime engine of Python and is responsible to run your scripts. It is the same what we know as 'Python Interpreter'. It is a program which read your *bytecode* and carry out each instruction by iterating over them.



Python Performance

The Python *bytecode* is a Python specific representation and is not a binary machine code. Hence, the interpreter still performs an internal compilation step. This is the reason why python code may not run as fast as C program. We can say that the python speed lies somewhere between traditional compiled language and traditional interpreted language.

Development Cycle

The Python's execution model does not separately define the development and execution environment. The system that compiles and that executes are the same. This increase the speed of development and hence adds more dynamicity to the language. It is very convenient for a python program to construct and execute another python program at runtime. Python allows you to modify python parts of a system onsite without needing to have or compile the entire system's code.

Author: Anidhya Bhatnagar

Source:
https://mylapuram.wordpress.com/2015/11/13/python-java-and-c-run-time/
Learning Python, Mark Lutz, 5th Edition