Quick Links for Python
Recent Articles
MCQ / Quizzes
Practice Problems

Basics
Introduction
New Generation Language
Keywords, Set 1 Set 2
Explore More...
Variables
Variables,Expressions & Functions
Global and Local Variables
Type Conversion
Explore More...
Operators
Increment and Decrement Operator
Teranry Operator & Divison Operator
Logical and Bitwise Not Operators on Boolean
Any & ALL

Operator Functions Set 1 & Set 2
Data Types
Introduction
Arrays Set 1, Set 2
String Methods Set 1, Set 2, Set 3
String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
Control Flow
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
Functions
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
Modules
Introduction

Numeric Functions & Logarithmic and Power functions
Calender Functions Set 1, Set 2
Complex Numbers Introduction & Important functions
Explore More...
Object Oriented Concepts
Class, Object and Members
Data Hiding and Object Printing
Inheritance, Subclass and super
Class method vs static method & Class or Static Variables
Explore More...
Exception Handling
Exception Handling
User-Defined Exceptions
Built-in Exceptions
Libraries and Functions
Timeit
Numpy Set 1, Set 2
Get and Post
import module & reload module
Collection Modules Deque, Namedtuple & Heap
Explore More...
Machine Learning with Python
Classifying data using Support Vector Machines(SVMs) in Python

K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
Misc
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
Applications and Projects
Creating a proxy webserver Set 1, Set 2
Send Messsage to FB friend
Twitter Sentiment Analysis & Whatsapp using Python
Desktop Notifier & Junk File Organizer
Explore More...

Difference between various Implementations of Python

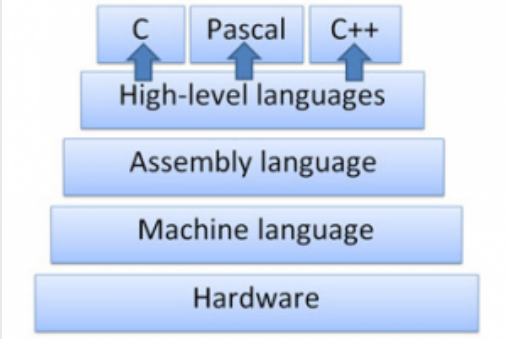
When we speak of Python we often mean not just the language but also the implementation. Python is actually a specification for a language that can be implemented in many different ways.

Background

Before proceeding further let us understand the difference between bytecode and machine code(native code).

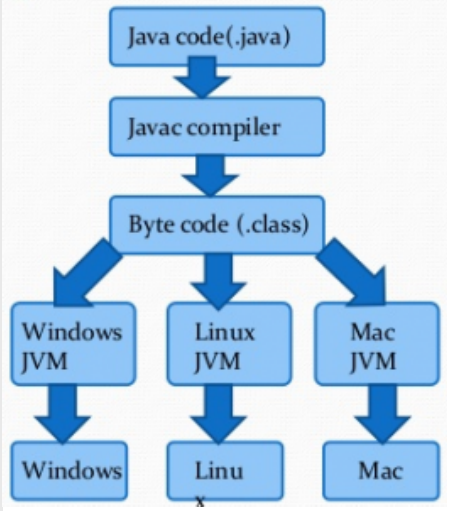
Machine Code(*aka native code*)

Machine code is set of instructions that directly gets executed by the CPU. Each instruction performs a very unique task, such as load or an logical operation on data in CPU memory. Almost all the high level languages such as C translate the source code into executable machine code with the help of Compilers, loaders and linkers. Every processor or processor family has its own machine code instruction set.



Bytecode

Bytecode is also binary representation executed by virtual machine (not by CPU directly). The virtual machine (which is written different for different machines) converts binary instruction into a specific machine instruction. One of the language that uses the concept of Bytecode is Java.



Machine Code is much faster as compared to Bytecode but Bytecode is portable and secure as compared to Machine Code.

Implementations of Python

Cpython

The default implementation of the Python programming language is Cpython. As the name suggests Cpython is written in C language. Cpython compiles the python source code into intermediate bytecode, which is executed by the Cpython virtual machine. CPython is distributed with a large standard library written in a mixture of C and Python. CPython provides the highest level of compatibility with Python packages and C extension modules. All versions of the Python language are implemented in C because CPython is the reference implementation.

Some of the implementations which are based on CPython runtime core but with extended behavior or features in some aspects are Stackless Python, wpython, MicroPython.

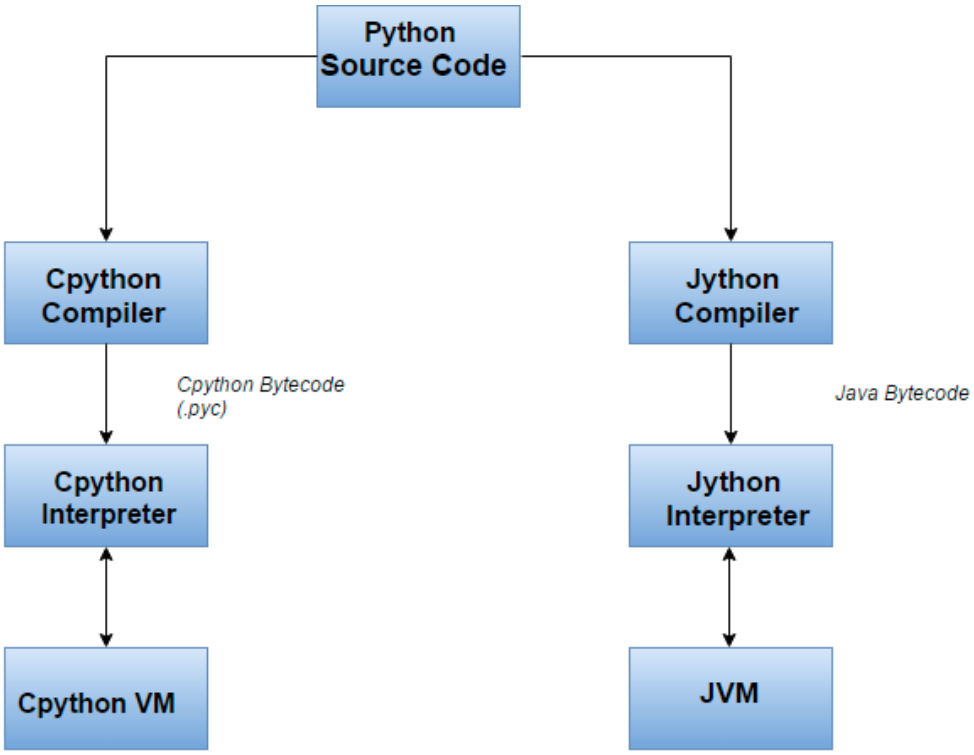
Stackless Python – CPython with an emphasis on concurrency using tasklets and channels (used by dspython for the Nintendo DS)

Other Implementations

There are some other implementations of the Python language too The only implementations that are known to be compatible with a given version of the language are **IronPython**, **Jython** and **PyPy**.

Jython

Jython is an implementation of the Python programming language that can run on the Java platform. Jython programs use Java classes instead of Python modules .Jython compiles into Java byte code, which can then be run by Java virtual machine. Jython enables the use of Java class library functions from the Python program. Jython is slow as compared to Cpython and lacks compatibility with CPython libraries.



IronPython

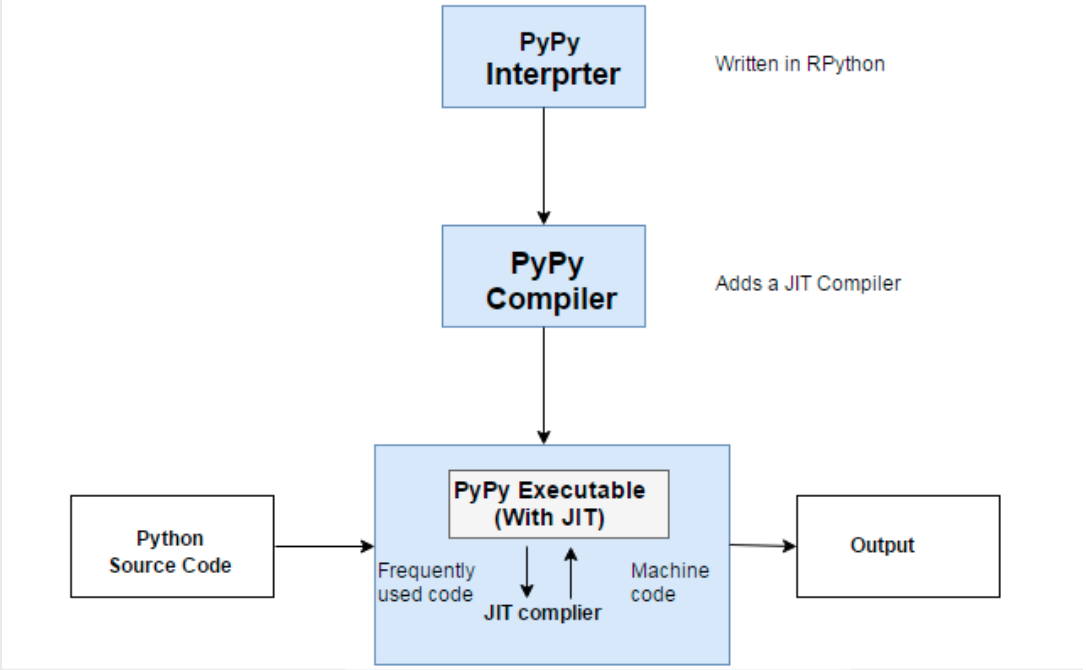
A Python implementation written in C# targeting Microsoft's .NET framework. Similar to Jython, it uses .Net Virtual Machine i.e Common Language Runtime. IronPython can use the .NET Framework and Python libraries, and other .NET languages can use Python code very efficiently. IronPython performs better in Python programs that use threads or multiple cores, as it has a JIT, and also because it doesn't have the Global Interpreter Lock.

PyPy

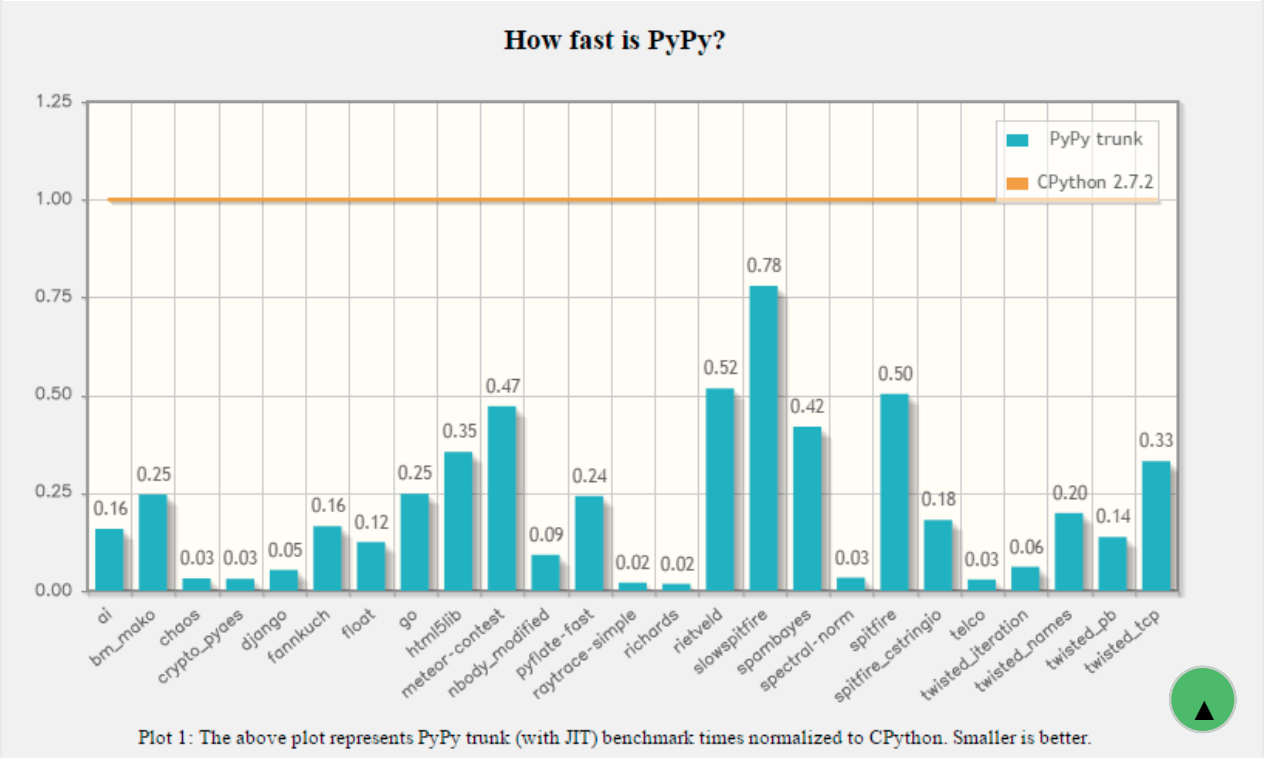
"If you want your code to run faster, you should probably just use PyPy." — Guido van Rossum (creator of Python)

Python is dynamic programming language. Python is said to be slow as the default CPython implementation complies the python source code in bytecode which is slow as compared to machine code(native code). Here PyPy comes in.

PyPy is an implementation of the Python programming language written in Python. The Interpreter is written in RPython (a subset of Python).



PyPy uses (just-in-time compilation). In simple terms JIT uses compilation methods to make interpreter system more efficient and fast. So basically JIT makes it possible to compile the source code into native machine code which makes it very fast.
PyPy also comes with default with support for stackless mode, providing micro-threads for massive concurrency. *Python is said to be approximately 7.5 times faster than Cpython.*



Plot 1: The above plot represents PyPy trunk (with JIT) benchmark times normalized to CPython. Smaller is better.

Image Refernce : http://speed.pypy.org/

Some other Implementations of Python are **CLPython**, **Pyston**, **Psyco**, **Cython**, **IPython**.

References:

- https://wiki.python.org/moin/PythonImplementations
- http://pypy.org/
- https://wiki.python.org/moin/IronPython
- http://www.jython.org/

This article is contributed by **Saurabh Daalia**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

(Login to Rate and Mark)

1 Average Difficulty : **1/5.0**
Based on **1** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!