- Python is a interpreted language
- when we download the python package it contains...
    - Interpreter
    - support libraries.

interpreter -
interpreter is a program that takes the source code and data and generates a output of the code without generating the intermediated code.

Compiler -
compiler is a program that converts a souce code into lower level language or machine code called as a intermediate code and then produces the output of the code.

- Python program files are given names that end in .py
- technically... this naming scheme is required only for files that are `imported` but most Python files have .py names for consistency.

When we instruct Python to run the script, there are a few steps that Python carries out before our code actually starts crunching away.

1. it's first compiled to something called "byte code"
2. routed to something called a "virtual machine"

Byte code compilation

- Done Internally and almost completely hidden from us.
- when we execute a program Python first compiles your source code into a format known as byte code.Compilation is simply a translation step and byte code is a lower-level,platform-independent representation of your source code.
- Python translates each of source statements into a group of byte code instructions by decomposing them into individual steps. This byte code translation is performed to speed execution.
- byte code can be run much more quickly than the original source code statements in your text file.
- Python process store the byte code of your programs in files that end with a .pyc extension (".pyc" means compiled ".py").
- Prior to Python 3.2, we will able to see these files show up on your computer after runing programs in the same directorie of program. For instance, you'll notice a script.pyc after importing a script.py.
- In 3.2 and later, Python instead saves its .pyc byte code files in a subdirectory named `__pycache__` located in the directory where your source files reside, and in files whose names identify the Python version that created them (e.g., script.cpython-33.pyc).
- The new `__pycache__` subdirectory helps to avoid clutter for different Python versions installed on the same computer from preventing overwriting of each other's saved byte code.

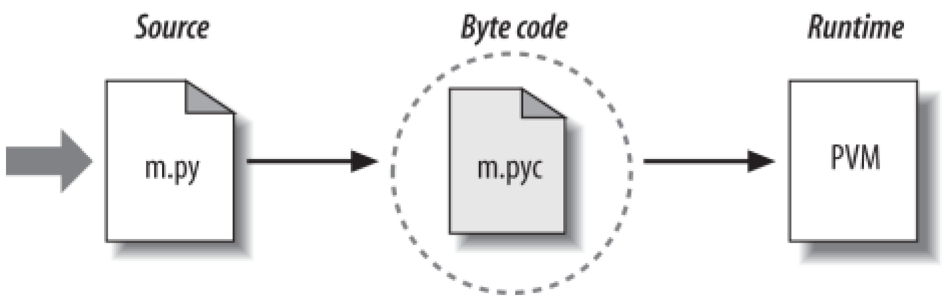
```
    Python saves byte code like this as a startup speed optimization. The next time we
run our program, Python will load the .pyc files and skip the compilation step, as
long as you haven't changed your source code since the byte code was last saved, and
aren't running with a different Python than the one that created the byte code.
```

```
1.  Source changes: Python automatically checks the last-modified timestamps of source
and byte code files to know when it must     recompile—if you edit and resave your
source code, byte code is automatically re-created the next time your program is run.

2.  Python versions: Imports also check to see if the file must be recompiled because
it was created by a different Python          version, using either a “magic” version
number in the byte code file itself in 3.2 and earlier, or the information present
  in byte code filenames in 3.2 and later.

If Python cannot write the byte code files to your machine, your program still works—
the byte code is generated in memory and simply discarded on program exit. However,
because .pyc files speed startup time, you’ll want to make sure they are written for
larger programs. Byte code files are also one way to ship Python programs—Python is
happy to run a program if all it can find are .pyc files, even if the original .py
source files are absent.
```

```
byte code is saved in files only for files that are imported, not for the top-l
evel files of a program that are only run as scripts means... byte code is also
never saved for code typed at the interactive prompt(CMD).
```

Python Virtual Machine (PVM)

- Once python program has been compiled to byte code or the byte code has been loaded from existing .pyc files), it is shipped off for execution to something generally known as the Python Virtual Machine.
- PVM is not a separate program, and it need not be installed by itself python itself contains it. the PVM is just a big code loop that iterates through your byte code instructions one by one, to carry out their operations. The PVM is the runtime engine of Python. it’s always present as part of the Python system, and it’s the component that truly runs your scripts.



Performance implications

as compare to C and C++ model there is a few differences in the Python model.

1. there is usually no build or “make” step in Python.code runs immediately after it is written.
2. Python byte code is not binary machine code. Byte code is a Python-specific representation. This is REason why some Python code may not run as fast as C or C++ code.

Development implications

```
    there is really no distinction between the development and execution environments
in python.the systems that compile and execute your source code are really one and the
same. for Ex. in C or C++ firstly code is compiled by compiler and executed in OS. but
in python source code is translated to byte-code and run by the Python Virtual
Machine.  This makes for a much more rapid development cycle. There is no need to
precompile and link before execution may begin. simply type and run the code. This
also adds a much more dynamic flavor to the language. This structure is also why
Python lends itself to product customization—because Python code can be changed on the
fly, users can modify the Python parts of a system onsite without needeing to have or
compile the entire system’s code.
```

Python Implementation Alternatives

```
there are mainly five implementations of the Python language
```

1. CPython
2. Jython
3. IronPython
4. Stackless
5. PyPy

- CPython is the standard implementation of Python language.
- All the other Python implementations have specific purposes and roles, though they can often serve in most of CPython’s capacities too. All implement the same Python language but execute programs in different ways.

CPython - The standard

- it is original and standard implementation of Python is usually called CPython.
- Python that we fetch from http://www.python.org (http://www.python.org) is CPython.
- it is the most complete and be more up-to-date and robust implementation than the alternative systems.

Jython - Python for Java

- The Jython system (originally known as JPython) is an alternative implementation of the Python language targeted for integration with the Java programming language.
- Jython consists of Java classes that compile Python source code to Java byte code and then route the resulting byte code to the Java Virtual Machine (JVM). Programmers still code Python statements in .py text files as usual.
- Jython’s goal is to allow Python code to script Java applications.Its integration with Java is remarkably seamless. Because Python code is translated to Java byte code, it looks and feels like a true Java program at runtime.
- Jython includes integration support that allows Python code to import and use Java classes as though they were coded in Python, and Java code to run Python code as an embedded language.

IronPython - Python for .NET

- third implementation of Python and newer than both CPython and Jython.
- IronPython is designed to allow Python programs to integrate with applications coded to work with Microsoft's .NET Framework.
- IronPython allows Python programs to act as both client and server components, gain accessibility both to and from other .NET languages, and leverage .NET technologies such as the Silverlight framework from their Python code.

Stackless - Python for concurrency

- Stackless Python system is an enhanced version and reimplementation of the standard CPython language oriented toward concurrency.
- Stackless Python can make Python easier to port to small stack architectures, provides efficient multiprocessing options, and fosters novel programming structures such as coroutines.

PyPy - Python for speed

- The PyPy system is another standard CPython reimplementation which focused on performance. It provides a fast Python implementation with a JIT (just-in-time) compiler.
- A JIT is really just an extension to the PVM that translates portions of your byte code all the way to binary machine code for faster execution. It does this as your program is running, not in a prerun compile step, and is able to created type-specific machine code for the dynamic Python language by keeping track of the data types of the objects your program processes. By replacing portions of your byte code this way, your program runs faster and faster as it is executing.
- it also provides tools for a "sandbox" model that can run untrusted code in a secure environment, and by default includes support for the prior section's Stackless Python systems and its microthreads to support massive concurrency.

- PyPy currently claims a 5.7X speedup over CPython
- PyPy today clocks in at 10X faster than CPython 2.7, and 100X faster than CPython 3.X.
- memory space is also optimized in PyPy - PyPy requiring 247MB and completing in 10.3 seconds compared to CPython's 684 MB and 89 seconds for same Program.

Execution Optimization Tools

1. Cython: A Python/C hybrid
2. Shed Skin: A Python-to-C++ translator
    - Shed Skin is an emerging system that takes a different approach to Python program execution.it attempts to translate Python source code to C++ code. then our computer's C++ compiler compiles it into machine code
3. Psyco: The original just-in-time compiler
    - Psyco is a specializing JIT compiler

- it generates machine code tailored to the data types that your program actually uses.
    - For example, if a part of your program uses different data types at different times, Psyco may generate a different version of machine code to support each different type combinations.

Frozen Binaries

- Using third-party tools it is possible to turn our Python programs into true executables known as frozen binaries in the Python world.
- These programs can be run without requiring a Python installation.
- Frozen binaries bundle together the byte code of your program files, along with the PVM and any Python support files your program needs into a single package and creates a single binary executable program that can easily be shipped to customers.

In [ ]:

```

```