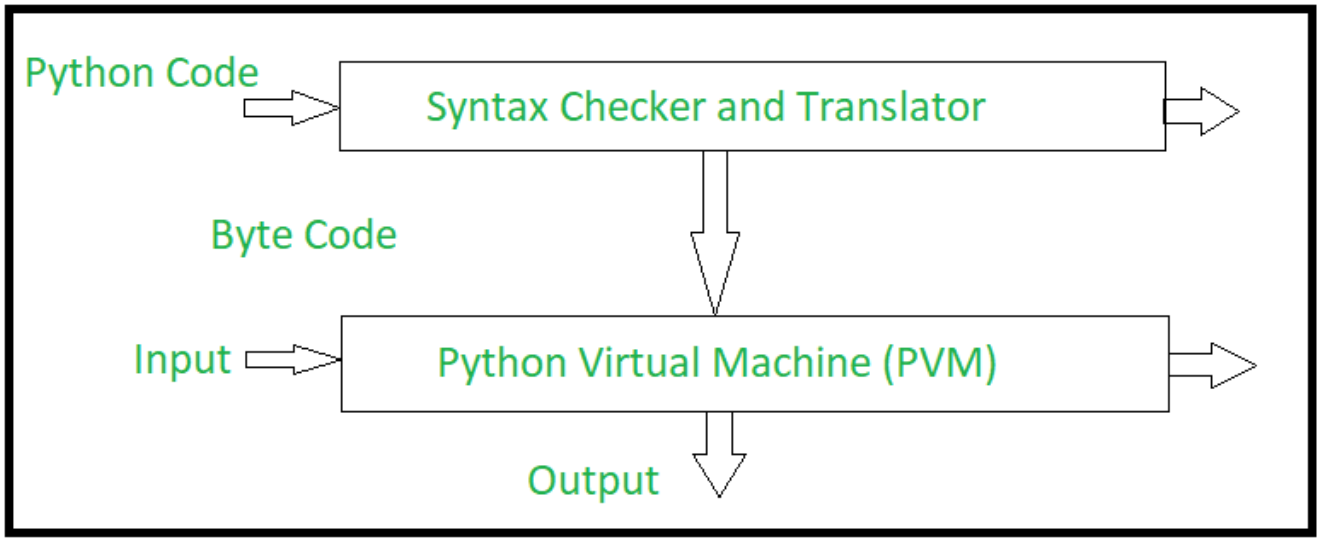
Build Process of Python Application

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

- Python doesn't convert its code into machine code, something that hardware can understand.
- It actually converts it into something called byte code.
- So within python, compilation happens, but it's just not into a machine language.
- It is into byte code and this byte code can't be understood by CPU.
- So we need actually an interpreter called the python virtual machine.
- The python virtual machine executes the byte codes.



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

1. It is compiled to bytecode.
2. Then it is routed to virtual machine.

Python Compiler :

- When we execute a source code, Python compiles it into a byte code. Compilation is atranslation step, and the byte code is a low-level platform-independent representation of source code.
- Note that the Python byte code is not binary machine code (e.g., instructions for an Intel chip).
- Actually, Python translate each statement of the source code into byte code instructions by decomposing them into individual steps.
- The byte code translation is performed to speed execution.
  Byte code can be run much more quickly than the original source code statements.
- It has.pyc extension and it will be written if it can write to our machine.

- So, next time we run the same program, Python will load the .pyc file and skip the compilation step unless it's been changed.
- Python automatically checks the timestamps of source and byte code files to know when it must recompile.
- If we re save the source code, byte code is automatically created again the next time the program is run.
- If Python cannot write the byte code files to our machine, our program still works.
- The byte code is generated in memory and simply discarded on program exit.
- But because .pyc files speed startup time, we may want to make sure it has been written for larger programs.
- When a Python executes a program, Python reads the .py into memory, and parses it in order to get a bytecode, then goes on to execute.
- For each module that is imported by the program, Python first checks to see whether there is a precompiled bytecode version, in a .pyo or .pyc, that has a timestamp which corresponds to its .py file.
- Python uses the bytecode version if any. Otherwise, it parses the module's .py file, saves it into a .pyc file, and uses the bytecode it just created.
- Byte code files are also one way of shipping Python codes.
- Python will still run a program if all it can find are.pyc files, even if the original .py source files are not there.

Python Virtual Machine (PVM)

- Once our program has been compiled into byte code, it is shipped off for execution to Python Virtual Machine (PVM).
- The PVM is not a separate program. It need not be installed by itself.
- Actually, the PVM is just a big loop that iterates through our byte code instruction, 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.
- It's the component that truly runs our scripts. Technically it's just the last step of what is called the **Python interpreter.**