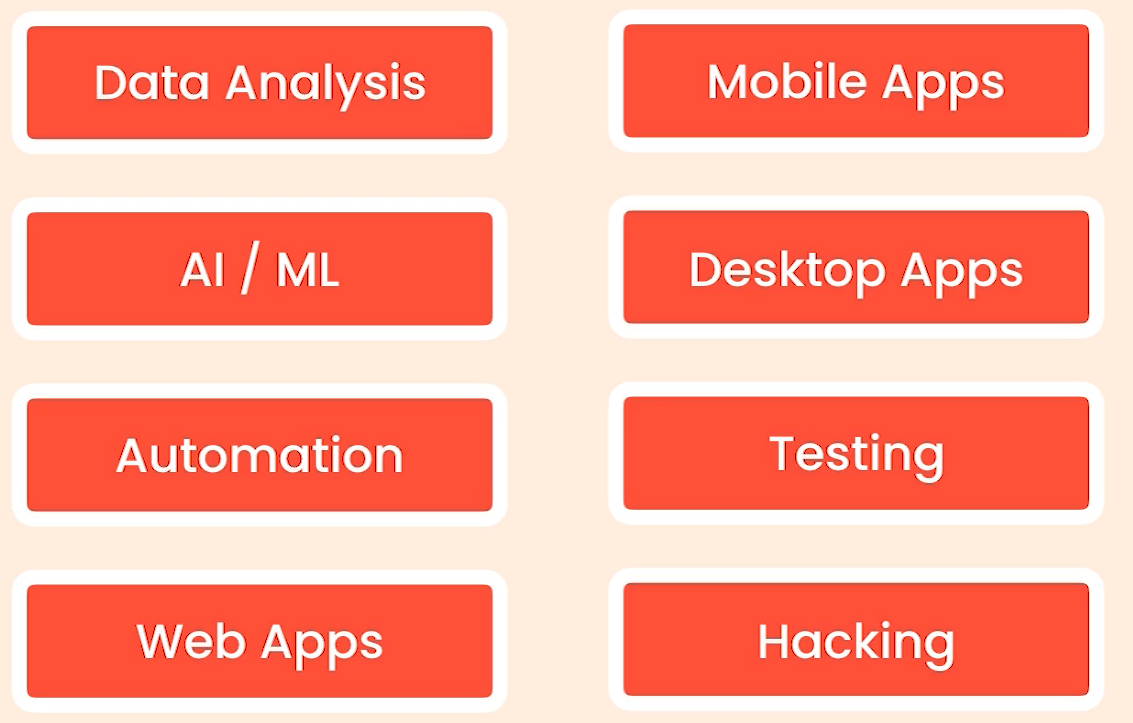
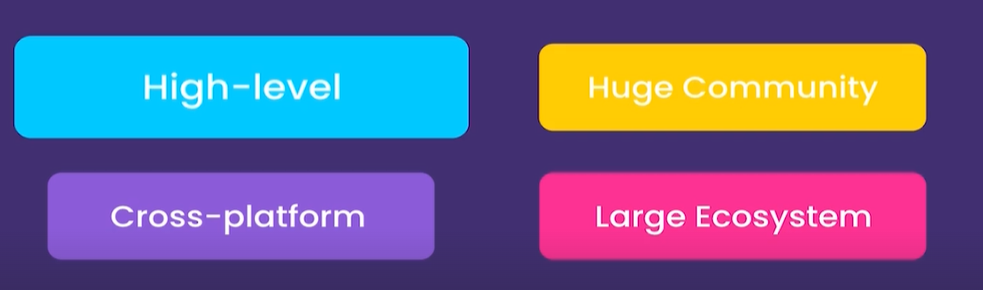


**What is python**:

With python we can do…



It is desirable because of these reasons…



🡪 Python is High level language so we do not have to worry about *complex tasks such as memory management* as we do in C++.

🡪 Due to being cross platform we can build and run Python apps Windows, Mac and Linux.

🡪 Lastly it has huge community and large ecosystem of libraries, frameworks and tools(*Since it has been around 25 years*).

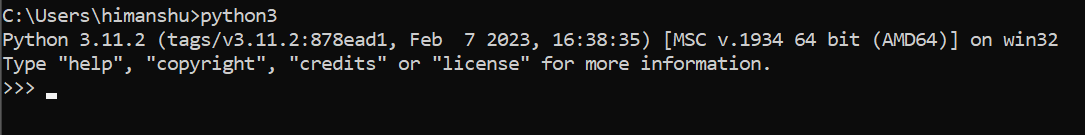
Note: There are two versions of python, python2(*legacy version of python*) and python3.

**Installing Python**:

Already Installed.

**Python Interpreter**:

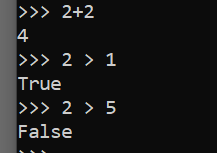
When we type python3 in the terminal after successful installation, we see an environment like this,



This is *python interpreter*, which is basically a program that executes python code.

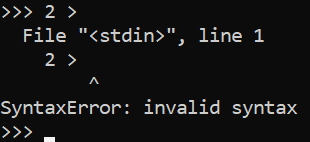
Note: We can type our python code in a file and give it to this interpreter or we can directly type our code here into this *interactive shell*.

We can write expressions(*a piece of code that produces a value*) here and see what value we get.



So far so good.

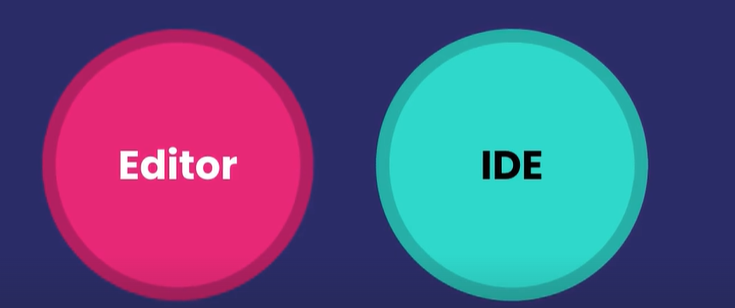
We get *syntax error* if we do not give any value after comparison operator.



Note: In programming, syntax means grammar, which is just like the concept of grammar in the languages we speak.

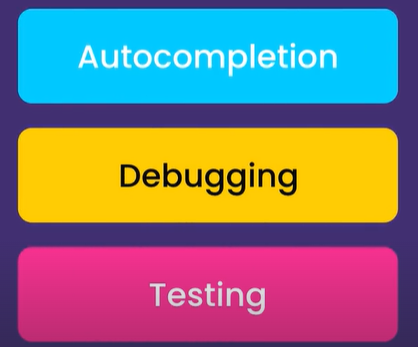
**Code Editors**:

When it comes to typing python code, we have two options.



We can either use code editor or IDE(*Integrated development environment*).

An IDE is basically a code editor with some fancy features like,



VSCode is a code editor.

**First Python Program**:

In Python we have many built in functions for performing various kind of tasks.

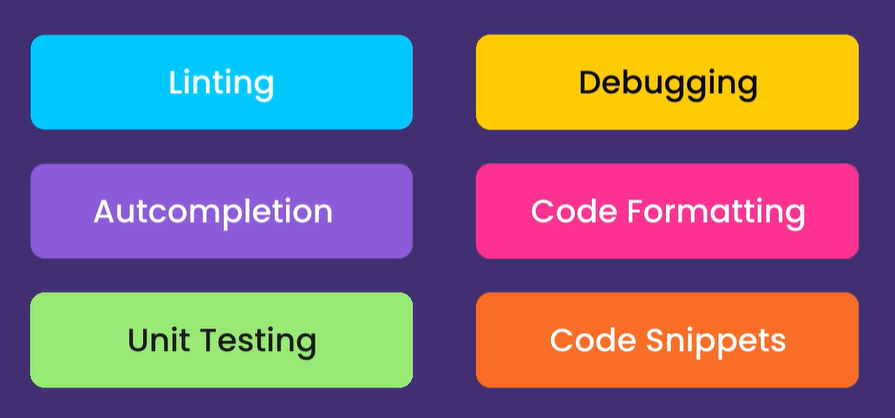
As a metaphor, take remote control of TV as an example. On this remote control we have a bunch of functions like turn off, turn on, change volume etc. These are the built in functions in your TV similar to build in functions of python.

print is one of the build in function and we can use it to print something on the screen.

Note: shortcut for opening and closing terminal in VS code is ctrl + ` . To run the program in terminal type python or python3 followed by name of file(python app.py).



**Python Extensions**:  
Add a new extension called Python(by Microsoft), with this we get below features,



*Linting* : Analyzing our code for potential errors

*Unit Testing*: Involves writing a bunch of tests for our code which we can also run these tests in an automated fashion to make sure our code is behaving correctly.

*Code Snippets*: Reusable code blocks that we can quickly generate and avoid typing everything by hand.

For Linting, we will install separate extension called PyLint(also by Microsoft).

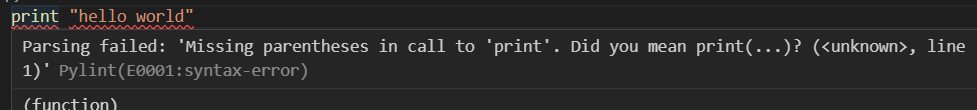
**Linting Python Code**:

Let us by writing some invalid code

print "hello world"

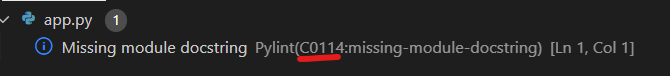
Here we are not using parentheses to contain the string after print(*it is a valid python2 code but invalid in python3*).

After saving the changes we see red underline and when we hover over it, we can see tool tip which is coming from PyLint with error message.

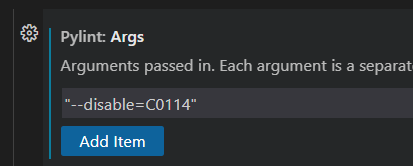


Note: You can disable a particular error message in PyLint by adding argument in PyLint settings.

Notice this error message



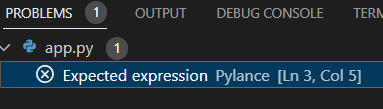
Disable this error message from PyLint settings



We do not get same error message after adding this argument.

Let us write an incomplete expression like this  and see what problem we get.

Note: Shortcut to open problem panel is **ctrl + shift + M**.



This problem panel lists all the issues in your code in one place.

So if we have an application with multiple files, PyLint will analyze all files and list them here.

Note: **ctrl + shift + P** opens command pallet where we can simply type name of the feature or command we want to access.

**Formatting Python Code**:

In Python community we have a bunch of documents called…

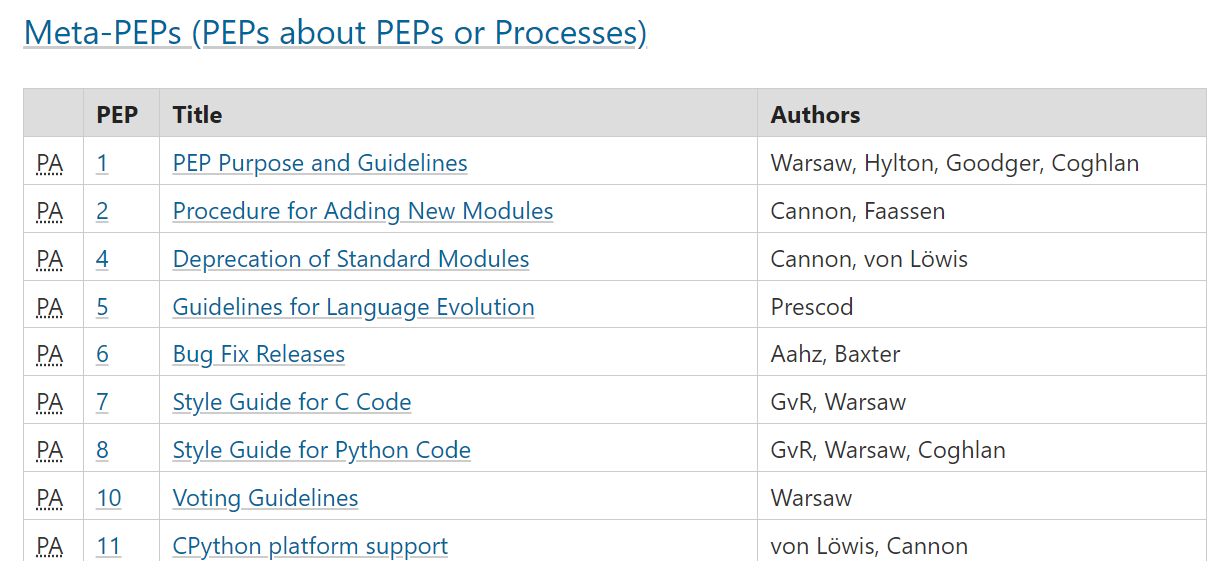


Or



If you search on google for *python PEPs*, we can see the list of all these peps under <https://peps.python.org/>

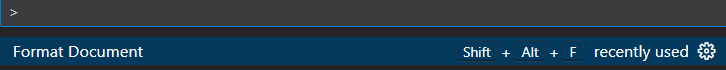
We can see each PEP has a number and a title, the one that is highly popular amongst python developers is PEP8



PEP8 is *style guide* for python code. A style guide is basically a document that defines a bunch of rules for formatting and styling our code.

*If you follow these conventions the code that you write end up being consistent with other people’s code*.

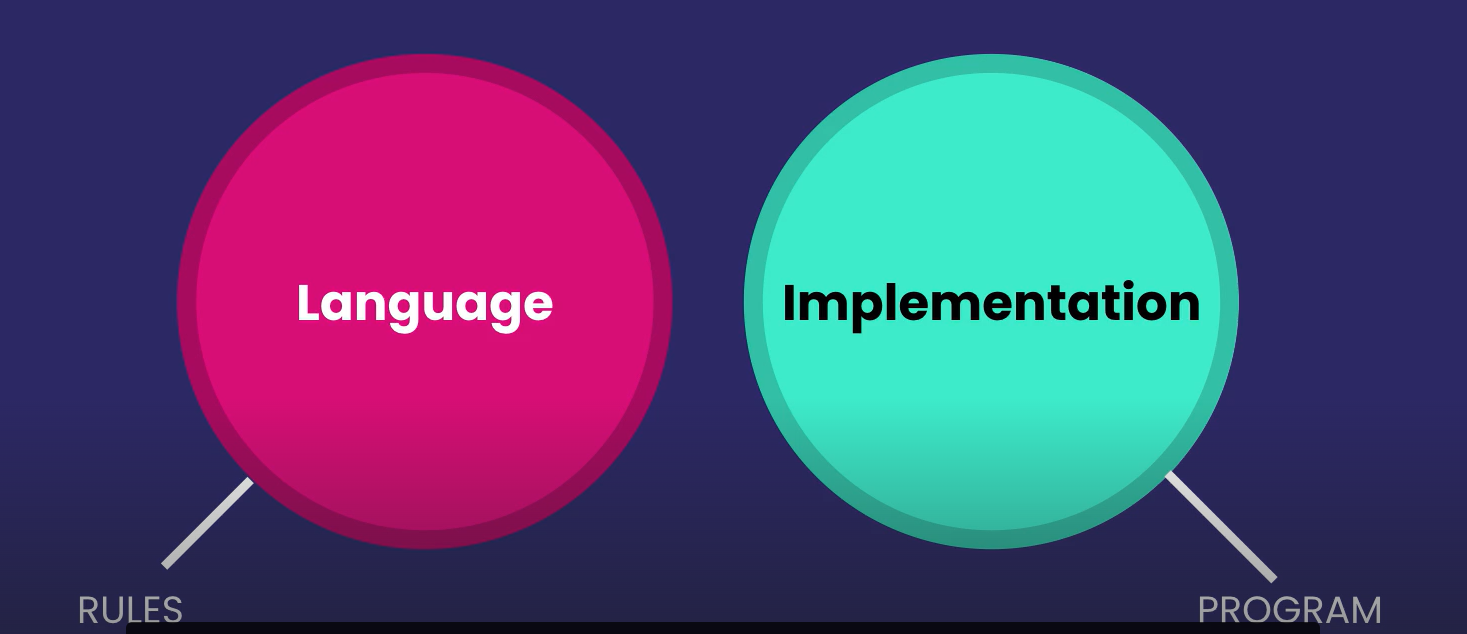
Note: To add a code formatter open command pallet and write format document.



It will prompt you to install a formatter for python files (like autopep8). Install it and when you save the changes your code will be automatically formatted as per pep8 conventions.

**Python Implementations**:

When we talk about python, we mean 2 separate things that are closely related. Python *Language* and a particular *Implementation*.

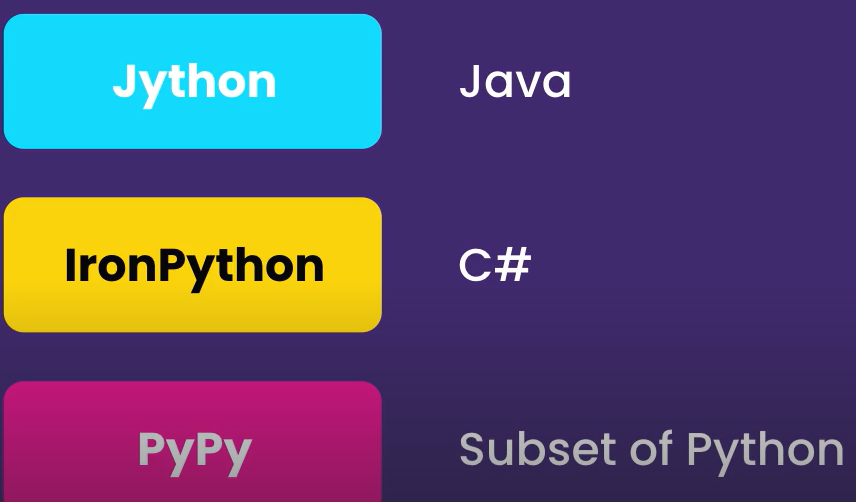


Python as a language is just a specification that defines a set of rules and parameters for writing python code.

**WHILE…**

A python implementation is basically a program that understand those rules and can execute python code

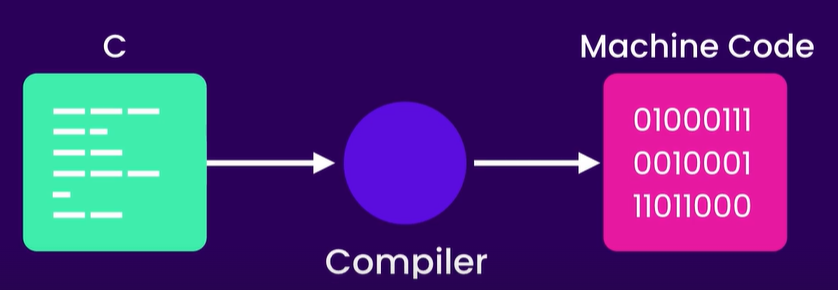
When we download Python.exe file from python.org website, that is the *default implementation of Python* called CPython(since it is a program written in C). Few other implementations are,



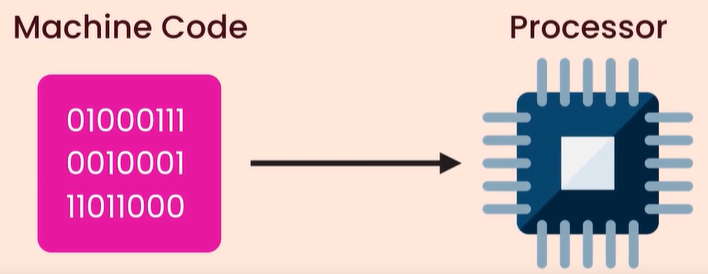
As new features are added in Python language, they are first supported by CPython and then they will gradually come to other implementations.

**How Python code is executed**:

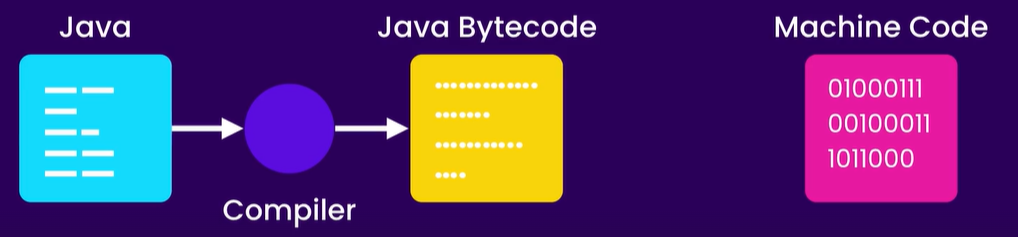
The programming languages we use like C, C#, Python, Java, these are all simple text based languages that we humans understand while computers only understand machine code.



So if you have some code written in C, we should convert it into machine code by using a C compiler.

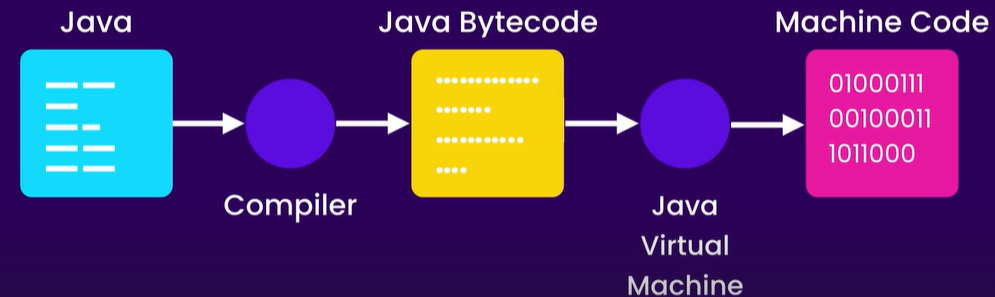


However this machine code is specific to the type of CPU of a computer. So if you compile a C program on a windows machine, we cannot execute it on a MAC, since windows and mac have different machine code just like how people from different countries speak different languages.



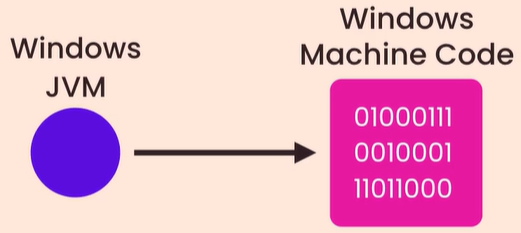
Java came to solve this problem. Java compiler does not compile java code into machine code, instead it compiles it into a portable language called Java Bytecode which is not specific to a hardware platform like Windows or Mac.

Now we still need to convert Java Bytecode to machine code. So Java also comes with a program called Java Virtual Machine.



When we run a Java program, JVM kicks in, It loads a Java Bytecode and then at run time, it will convert each instruction to machine code.

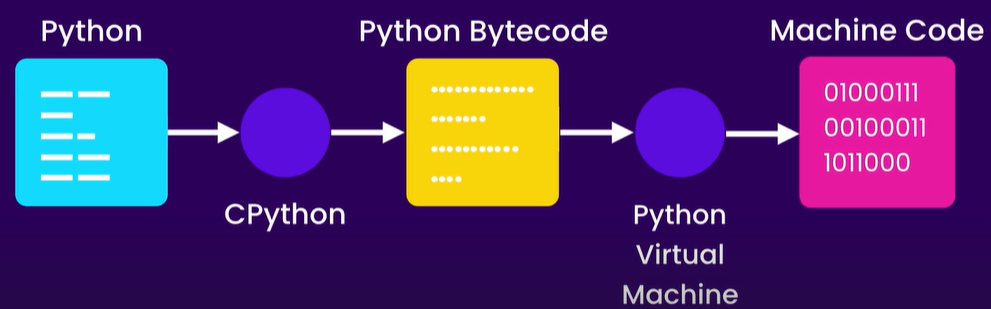
With this model we can run Java Bytecode on any platform that has JVM.



We have JVM implementation for windows, mac and so on.

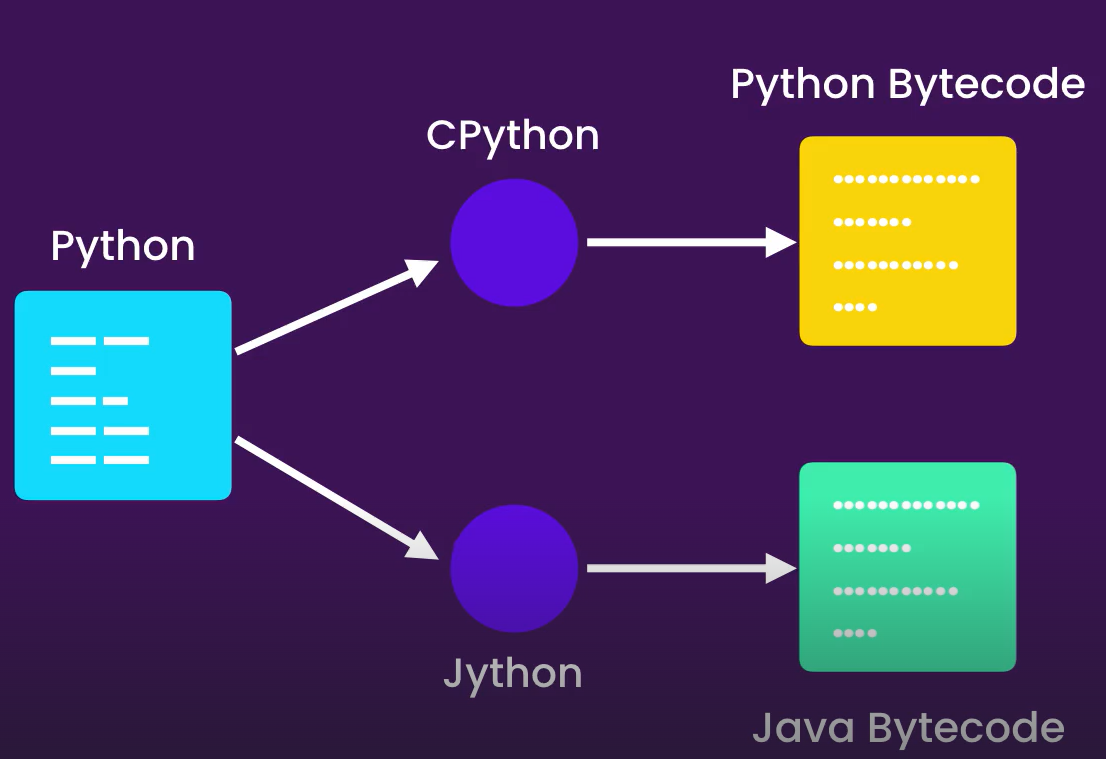
So the JVM implementation of windows knows how to convert Java Bytecode in to machine code that a windows machine can understand.

C# and Python have also taken the same route so they are platform independent.



“*When we* ***run a python program using CPython****, first it will* ***compile Python code into Python Bytecode*** *then it will* ***pass that Bytecode to Python virtual machine*** *which will in turn* ***convert it into machine code and execute it***”.

*How Jython works*:



I hope diagram is self – explanatory at this point.