**Python Programming Language**

Python is a general-purpose programming language in a similar way to other programming languages like C++,C# , Java etc..

It has been around for some considerable time having been originally conceived back in the 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands.

The language is named after one of Guido’s favourite TV programs “Monty Pythons Flying Circus”.

As a language it has gained in interest over recent years, particularly within the commercial world, with many people wanting to learn the language. This increased interest in Python is driven by several different factors:

1. Its flexibility and simplicity which makes it easy to learn.
2. Its use by the Data Science community where it provides a more standard programming language than some rivals such as R.
3. Its ability to run on (almost) any operating system, but particularly the big three operating systems Windows, MacOS and Linux.
4. The availability of a wide range of libraries (modules) that can be used to extend the basic features of the language
5. It is free
6. There are 65000 libraries pertaining to any domain , which can be called and get the output
7. Python is popular because of 4 popular libraries like

Pandas - Data Analysis

numpy - numerical analysis - stats and maths related problem

matplotlib - plotting graph - histograms, barchart ,line plot , scatter plot

sklearn - for machine learning , no machines have capabilities to learn on its own , you create alogorthim , it understands

realtation between independednt and dependent data,

i have humidity/rainfall/temp and crop yield of last 20 yr

ML is all about predictive analysis based on historical data , if we book ola cab it will predict it will arrive

in next 10 mins , we predict based on historical data

sklearn has all algorithms for doing this.

Deep learnng is subset of ML

feature selection /column selection is not a role of data scientist, it is automatic

ML is good for 2 dimensional data - rows and columsn

DL is good for 3 dimensional data - images - RPG pixels

**Python Versions**

Currently there are two main versions of Python called Python 2 and Python 3.

• Python 2 was launched in October 2000 and has been, and still is, very widely used. • Python 3 was launched in December 2008 and is a major revision to the language that is not backward compatible.

The issue between the two versions can be highlighted by the simple print facility: • In Python 2 this is written as print 'Hello World' • In Python 3 this is written as print ('Hello World')

It may not look like much of a difference but the inclusion of the '()' marks a major change and means that any code written for one version of Python will probably not run on the other version. There are tools available, such as the 2–3 utility, that will (partially) automate translation from Python 2 to Python 3 but in general you are still left with significant work to do.

This then raises the question which version to use?

Although interest in Python 3 is steadily increasing there are many organisations that are still using Python 2. Choosing which version to use is a constant concern for many companies. However, the Python 2 end of life plan was initially announced back in 2015 and although it has been postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3, it is still living on borrowed time.

Python 3 is the future of the Python language and it is this version that has introduced many of the new and improved language and library features we will solely focuss on Python 3.

**Python Programming**

There are several different programming paradigms that a programming language may allow developers to code in, these are:

• Procedural Programming - in which a program is represented as a sequence of instructions that tell the computer what it should do explicitly. Procedures and/ or functions are used to provide structure to the program; with control structures such as if statements and loop constructs to manage which steps are executed and how many times. Languages typifying this approach include C and Pascal.

• Declarative Programming languages, such as Prolog, that allow developers to describe how a problem should be solved, with the language/environment determining how the solution should be implemented. SQL (a database query language) is one of the most common declarative languages that you are likely to encounter.

• Object Oriented Programming approaches that represent a system in terms of the objects that form that system. Each object can hold its own data (also known as state) as well as define behaviour that defines what the object can do. A computer program is formed from a set of these objects co-operating together. Languages such as Java and C# typify the object oriented approach.

• Functional Programming languages decompose a problem into a set of functions. Each function is independent of any external state, operating only on the inputs they received to generate their outputs. The programming language Haskell is an example of a functional programming language.

Some programming languages are considered to be hybrid languages; that is they allow developers to utilise a combination of difference approaches within the same program. Python is an example of a hybrid programming language as it allows you to write very procedural code, to use objects in an object oriented manner and to write functional programs.

​

​

**Python Libraries**

As well as the core language, there are many libraries available for Python.

These libraries extend the functionality of the language and make it much easier to develop applications. These libraries cover

1. numpy is a package for processing arrays of single or multidimensional type
2. pandas is a package for powerful data structures for data analysis, time series and statatics
3. matplotlib is a package for drawing electronic sircuits and 2D graphics
4. web frameworks such as Django/Flask,
5. Generation of Microsoft Excel files using the Python Excel library,
6. machine learning using libraries such as SKLearn and TensorFlow.

**Python Execution**

Python is not a precompiled language in the way that some other languages you may have come across are (such as C++). Instead it is what is known as an interpreted language (although even this is not quite accurate). An interpreted language is one that does not require a separate compilation phase to convert the human readable format into something that can be executed by a computer. Instead the plain text version is fed into another program (generally referred to as the interpreter) which then executes the program for you. Python actually uses an intermediate model in that it actually converts the plain text English style Python program into an intermediate 'pseudo' machine code format and it is this intermediate format that is executed

sourcecode(hello.py) ->compile using python compiler (hello.pyc)->run using PVM (machine code) ->final display on computer

​

​

**Python Execution model**

1. Python program with extension .py is saved.
2. compile the code , so that compiler converts the python program into another code called byte code (Byte code represents a fixed set of instructions that represents all operations like artimetic operations,comparison operations, memory related operations etc which run on any operating system and hardware and byte instructions are system independent or platform independent)
3. The compile code is present in .pyc file , called as python compiled code
4. Next step will be to run the program , since byte code can not be directly executed by the computer, as it can be only binary code consisting of 0s and 1s can be understandable by the machine and it is called as machine code , so we need to convert byte code into machine code for this purpose PVM(Python Virtual Machine)
5. PVM uses an interpreter which understands the byte code and converts it into machine code , these instructions are then executed by the processor and results are displayed
6. An interpreter translates the program source code line by line , hence it is slow ,to rectify this problem some flavours of python , compiler is added to the PVM which converts the byte code into machine code , and this compiler is called as JIT(just in compiler) to improve the speed. This JIT is not available in all the favours of python , it is available in PyPy.
7. Normally when we compile the source code , we will not be able to see the compiled output(.pyc files) as this is done internally and the final output is displayed
8. if we observe we can not find either the directory or the .pyc files .but if we wanted to create it then we can do the following command

c:> python -m py\_compile hello.py

where -m option represents module and the module name is py\_compile .this module will create .pyc file

1. The compiler creates a seperate directory in the current directory by the name **pycache** , where it stores .pyc file.
2. and the file name will be like hello.cpython-34.pyc , where cpython indicates that we are using the python compiler that was created using C.
3. This pyc files which contains bytecode will be converted to machine code by PVM c:> python hello.cpython-34.pyc which will produce the output.
4. .pyc files can be directly distributed to the user who can directly run these files using PVM and view the output.

If we want to see the byte code instructions that were created internally by python compiler before they are executed bu PVM then we can use dis module c:>python -m dis hello.py

The way in which the Python interpreter processes a Python program is broken down into several steps. The steps shown here are illustrative (and simplified) but the general idea is correct.

1. First the program is checked to make sure that it is valid Python. That is a check is made that the program follows all the rules of the language and that each of the commands and operations etc. is understood by the Python environment.
2. It then translates the plain text, English like commands, into a more concise intermediate format that is easier to execute on a computer. Python can store this intermediate version in a file which is named after the original file but with a '.pyc' extension instead of a '.py' extension (the 'c' in the extension indicates it contains the compiled version of the code).
3. The compiled intermediate version is then executed by the interpreter. When this program is rerun, the Python interpreter checks to see if a '.pyc' file is present. If no changes have been made to the source file since the '.pyc' was created, then the interpreter can skip steps 1 and 2 and immediately run the '.pyc' version of the program.

One interesting aspect of Python’s usage is that it can be (and often is) used in an interactive fashion (via the REPL), with individual commands being entered and executed one at a time, with context information being built up. This can be useful in debugging situations.

**Running Python Programs**

There are several ways in which you can run a Python program, including

• Interactively using the Python interpreter

• Stored in a file and run using the Python command

• Run as a script file specifying the Python interpreter to use within the script file

• From within a Python IDE (Integrated Development Environment) such as PyCharm.

**Interactively Using the Python Interpreter**

It is quite common to find that people will use Python in interactive mode. This uses the Python REPL (named after Read Evaluate Print Loop style of operation). Using the REPL, Python statements and expressions can be typed into the Python prompt and will then be executed directly. The values of variables will be remembered and may be used later in the session.

To run the Python REPL, Python must have been installed onto the computer system you are using. Once installed you can open a Command Prompt window (Windows) and type python into the prompt.

we interactively type in several Python commands and the Python interpreter 'Read' what we have typed in, 'Evaluated' it (worked out what it should do), 'Printed' the result and then 'Looped' back ready for further input.

e.g • Printed out the string 'Hello World'.

• Added 5 and 4 together and got the result 9.

• Stored the string 'John' in a variable called name.

• Printed out the contents of the variable name.

To leave the interactive shell (the REPL) and go back to the console (the system shell), press Ctrl-Z and then Enter on Windows, Alternatively, you could also run the Python command exit() or quit().

**Running a Python File**

We can of course store the Python commands into a file. This creates a program file that can then be run as an argument to the python command. For example, given a file containing the following file (called hello.py) with the 4 commands in it: To run the hello.py program on a PC using Windows we can use the python command followed by the name of the file

**Using Python in an IDE**

We can also use an IDE such as PyCharm to writing and execute our Python program. The same program is shown using PyCharm below: The simple set of commands are again listed in a file called hello.py. However, the program has been run from within the IDE and the output is shown in an output console at the bottom of the display.