Python Fundamentals

Python-Orientation:

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development,
- software development,
- mathematics,

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.
- Python Syntax compared to other programming languages
- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Fundamentals

Interpreter vs Compiler:

What is a Compiler?

- A compiler is software that takes the entirety of your source code and converts it into a form that can be executed by your operating system, usually called a "binary" or "executable".
- Passing the entirety of your source code through the compiler has multiple benefits
    - your entire program is pre-converted into whatever format your system needs to execute
    - optimizations can be applied during the compilation process
    - the compiler can detect issues with your code before you try and run it

Some compiled languages are:

- C++
- Java
- Go

What is an Interpreter

- An interpreter is similar to a compiler in that it converts source code into machine code, but it does so line by line instead of all at once.
- Using an interpreter brings multiple advantages
    - Interpreters are specific to their operating system, so as long as you have access to the software you can run your applications. This makes applications created via interpreted languages much more portable than applications created in compiled languages
    - Interpreted languages are typically easier to debug due to the interpreter going line-by-line in the compilation/execution process. It is easier to track where things are happening

Some interpreted languages are:

- Python
- JavaScript
- Ruby
- "Compiler vs Interpreter" describes two different approaches to how programming languages go from source code to executable code
- Compiled languages tend to be better when execution speed is important.

Python Fundamentals

- Interpreted languages tend to be more flexible in their deployment options, but due to compilation happenings line-by-line at runtime their execution tends to be slower than compiled language applications

REPL Jupyter:

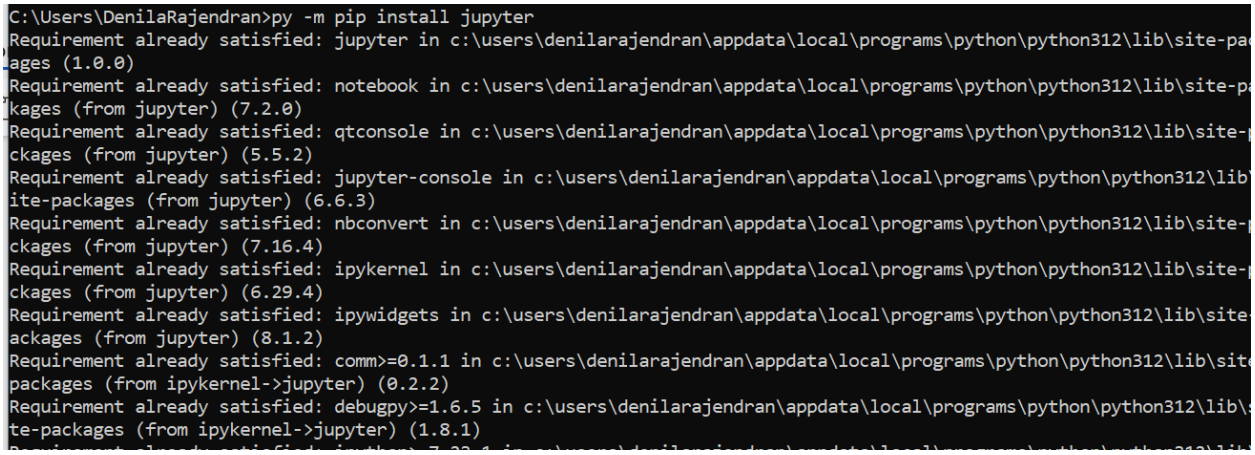
Link: https://jupyter.org/try-jupyter/consoles/?path=notebooks%2FIntro.ipynb

Shift+ Enter (Run the commands in the shell)

Link: https://jupyter.org/try-jupyter/lab/index.html

In command line:

Py –m pip install jupyter



Link to install jupyter:
https://jupyter.org/install