

Scripting and testing

Introduction to Python scripting

An introduction to Python scripting

Python scripting is often used to automate a specific series of tasks, leading to a more efficient process. Python scripts can be used for:

- Facilitating data analysis and visualisation.
- Developing both desktop and web applications.
- The development of complex scientific and numeric applications.
- Machine learning and artificial intelligence.
- Business applications.



Running Python in Jupyter notebooks is not the only way to interact with code. We can run scripts through IDEs (Integrated Development Environments) such as **PyCharm** or **VS Code** or even through the **command line**. Through the rest of this train we will expand on the different tools used for compiling and executing Python scripts.

Tools used to compile and execute Python scripts



Command line

There are various reasons why we might wish to utilise Python in a command line.

These may include:

- We want to conduct quick **one-liner computations**.
- We want to execute Python scripts using **parsed arguments**.
- We want to run simple code.



Integrated Development Environment (IDE)

An **IDE** is an application that can be used to write code in a script format.

Scripts enable us to write much lengthier code, which can then be executed in a command terminal or imported into a Jupyter Notebook.

IDEs **facilitate this process**.
Examples of IDEs include **VS Code** and **PyCharm**.



Jupyter Notebook

Jupyter Notebook is an **open-source web application** that allows us to create and share documents that contain **code, equations, visualisations** and **narrative text**.

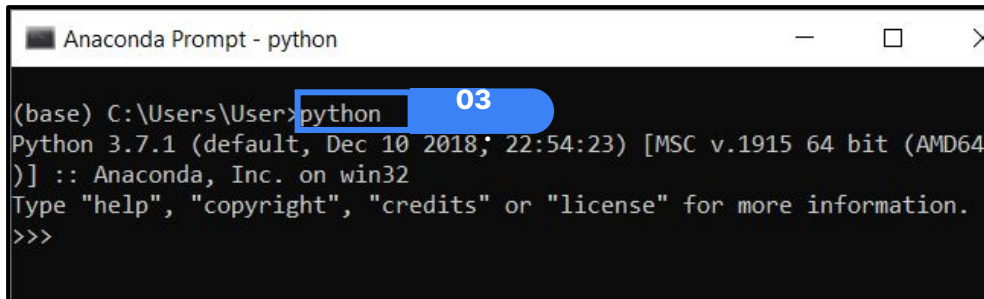
Applications include:

- Data exploration
- Machine learning
- Aiding in explanation and interactivity of code

Using Python in a command terminal on Windows

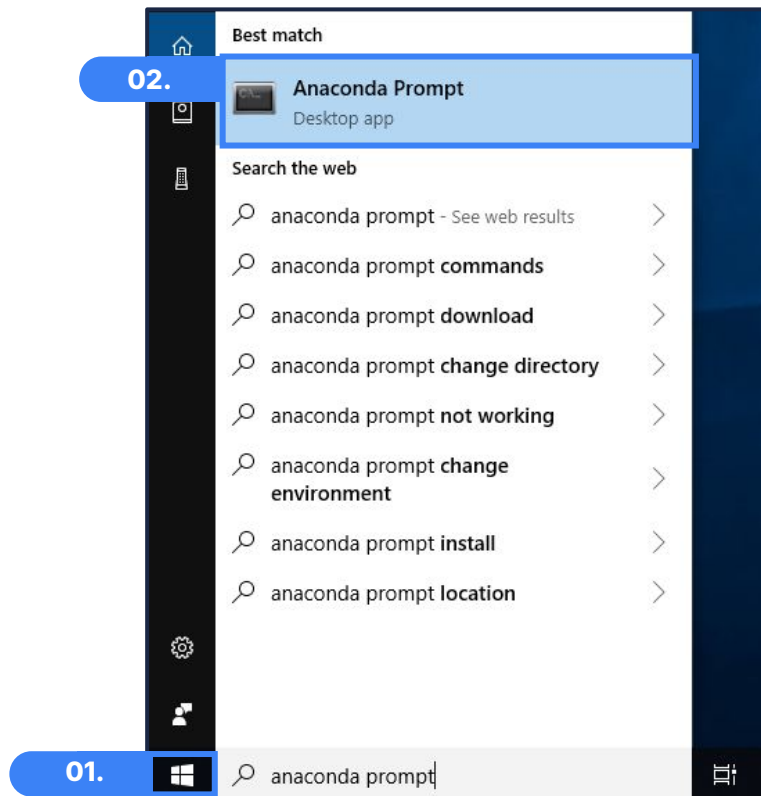
Let's see how we can use Python in a command terminal on Windows:

01. Click the Start button.
02. From the menu, search for "**Anaconda prompt**" and select it.
03. Enter "**Python**" at the blinking cursor and hit **Enter**. This will start an instance of Python.



```
Anaconda Prompt - python

(base) C:\Users\User>python
Python 3.7.1 (default, Dec 10 2018; 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



Using Python in command terminal on Mac

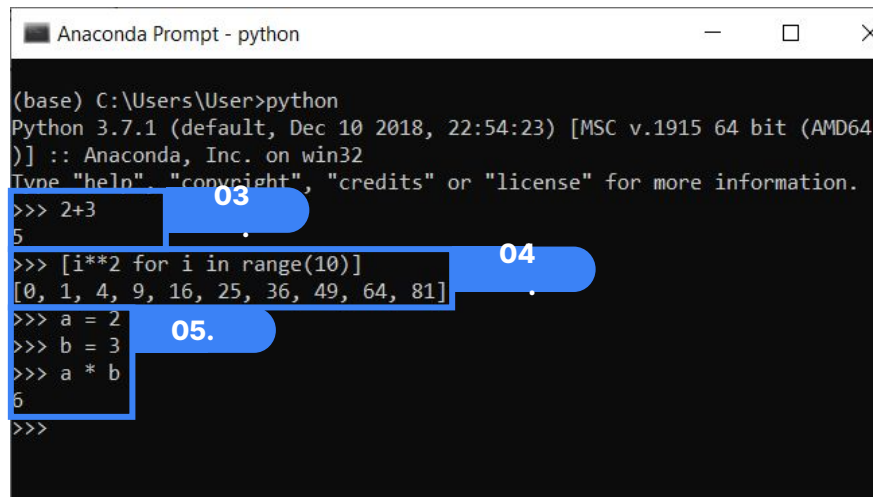
Let's see how we can use Python in a command terminal on Mac:

01. In Finder, navigate to **Applications > Utilities > Terminal.app**.
02. Start a Python instance by typing Python at the blinking cursor and hit **Enter**.

We can now run Python commands in the terminal. Since Python is a universal language, **the commands will be the same whether we are using a Mac or Windows O.S.**

We can input code at each line starting with `>>>`. The output of the code is displayed below this line.

03. Simple addition
04. List comprehension
05. Setting variables



```
Anaconda Prompt - python

(base) C:\Users\User>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.

>>> 2+3
5
>>> [i**2 for i in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> a = 2
>>> b = 3
>>> a * b
6
>>>
```

What are Python scripts?

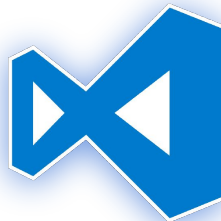
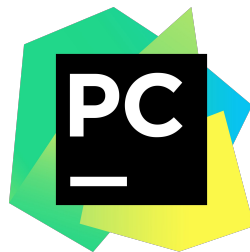
Python scripts allow us to write much longer sections of code that can then be executed in a command line or imported to a Jupyter notebook.

Though we can use a **Notepad or a word processor** to write our Python scripts, it's much more advisable to **use an IDE**. These come with a number of features that make **writing and executing scripts far easier**.

There are a number of IDEs available for free, including:

- PyCharm (community edition is free)
- VS Code
- Spyder

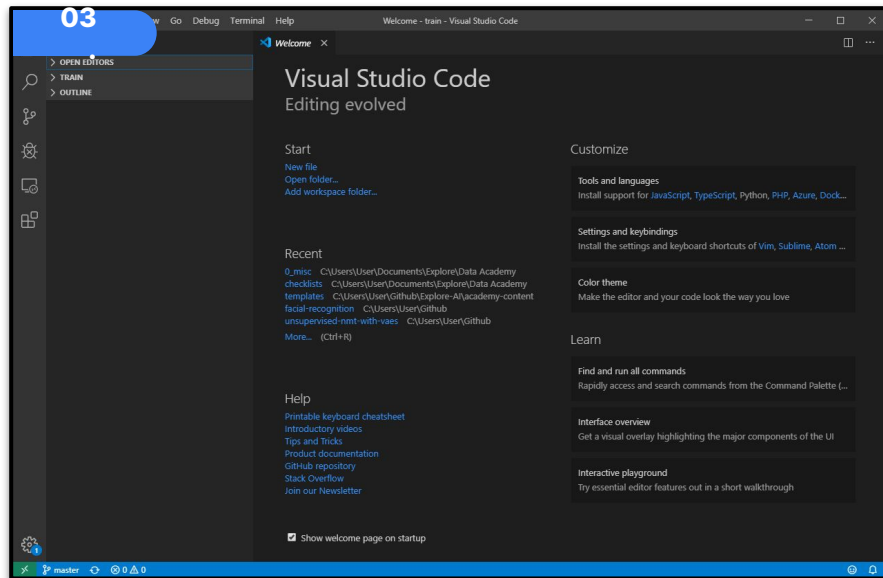
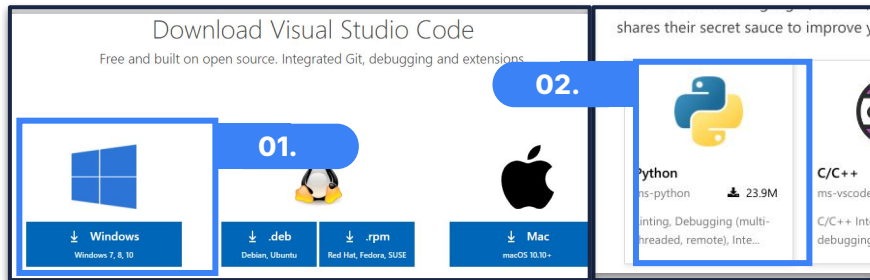
None of these are perfect. Each comes with its own advantages and disadvantages. As a growing data scientist, we each have our own preferences. For the purpose of this tutorial, we'll be using **VS Code**.



Installing Visual Studio Code on Windows

Follow the links and instructions below to install VS Code on Windows.

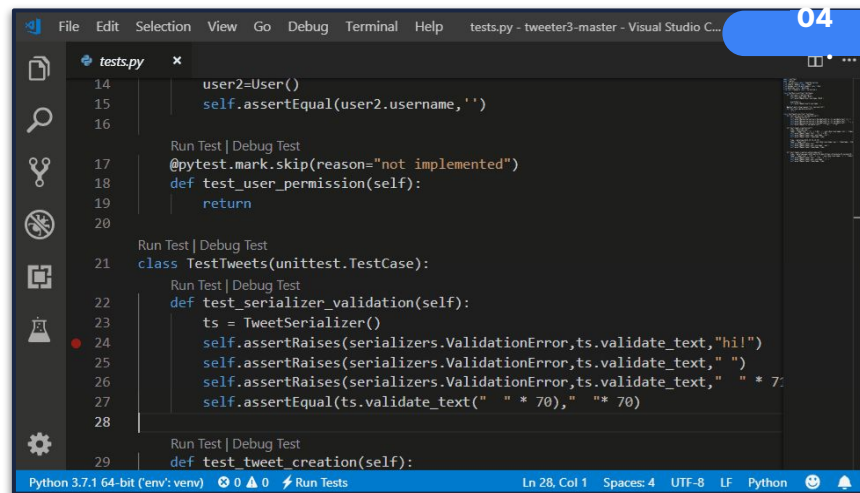
01. Install the Windows version of VS Code.
02. Download and install the Python extension for VS Code.
03. Open VS Code. Click 'New file' or add an existing workspace.
04. Start coding!



Installing Visual Studio Code on Mac

Follow the links and instructions below to install VS Code on Windows.

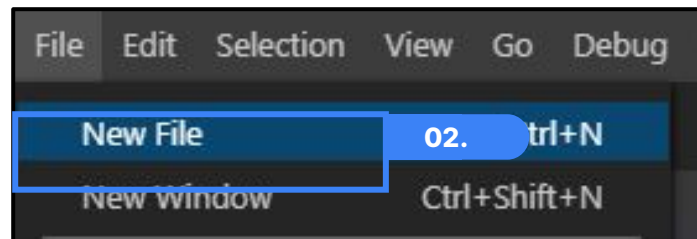
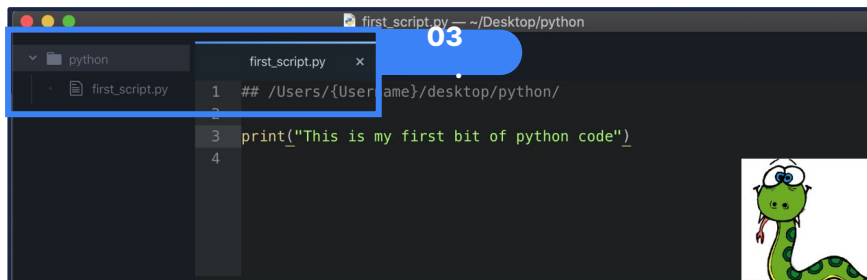
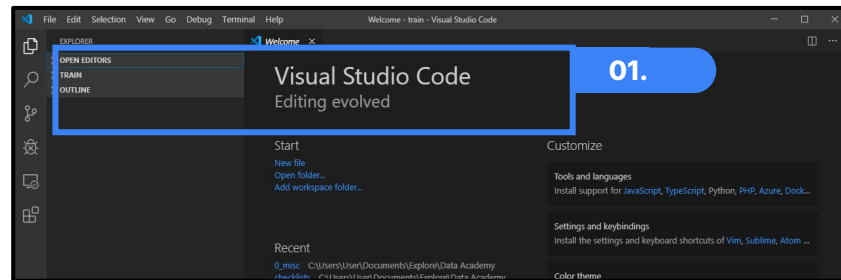
01. Install the **Mac version of VS Code**.
02. Locate the downloaded archive file. Move the visual studio code.app file to the application folder.
03. Download and install the **Python extension for VS Code**. Be sure to take note of the Python requirements!
04. Open VS Code. Click 'New file' or add an existing workspace.
05. Start coding!



Writing our first Python script

We're now ready to write our first Python script. We can use a text editor or IDE to write our script. Let's have a look at the process of writing a script.

- 01.** Open a **text editor or IDE** of choice; for this train we will be using VS Code.
- 02.** Navigate to **File > New File**.
- 03.** To be able to execute our new script we will have to save our file with a **.py extension**.



Executing a Python script on Windows

To execute our Python script:

01. Open a **new instance** of an anaconda prompt.

02. Navigate to the path where we saved our script.

To do this:

- Take note of the **path where the file is saved** in VS Code.
- Convert the '>' symbols to '\' symbols.
- Therefore **C:>Users>User>Documents>** becomes **C:\Users\User\Documents**.

03. Next, we type the path from step 2 into our **anaconda prompt** as **cd <your path here>** which in our case becomes **cd C:\Users\User\Documents**.

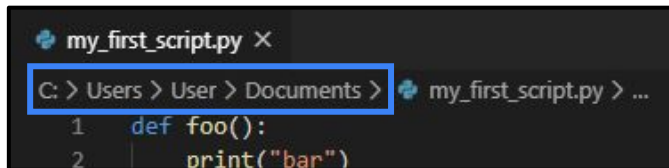
04. We can now execute our script by running **python my_first_script.py**. The output should be the printed word **'bar'**.



Anaconda Prompt

```
(base) C:\Users\User>
```

01.



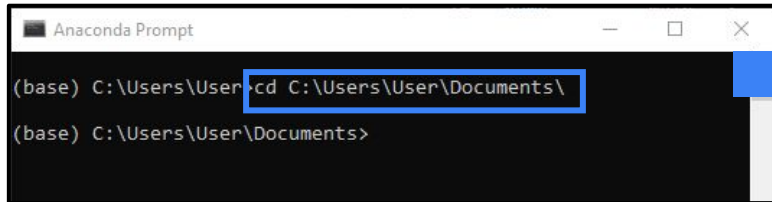
my_first_script.py X

```
C: > Users > User > Documents > my_first_script.py > ...
```

```
1 def foo():
```

```
2     print("bar")
```

02.

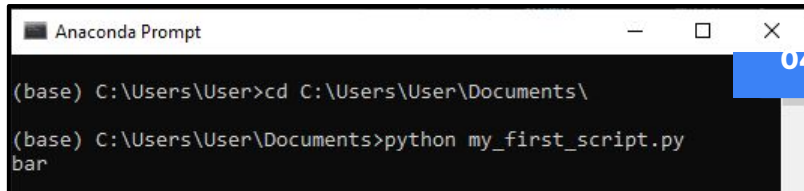


Anaconda Prompt

```
(base) C:\Users\User>cd C:\Users\User\Documents\
```

```
(base) C:\Users\User\Documents>
```

03.



Anaconda Prompt

```
(base) C:\Users\User>cd C:\Users\User\Documents\
```

```
(base) C:\Users\User\Documents>python my_first_script.py
```

```
bar
```

04.

Executing a Python script on Windows

To execute our Python script:

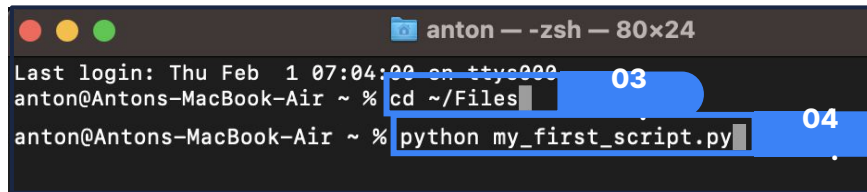
01. Open a new Terminal window.

02. Copy the path to the script's **.py** file.

A shortcut: Right-click on the file while holding down the **Option** key and select "**Copy ... as Pathname**" to copy the path to the clipboard.

03. In the Terminal window, type **cd** and then paste the path that we copied in step 2.

04. We can now execute the script by typing **python my_first_script.py** and hitting **Enter**. The output should be the printed word 'bar'.



Importing a Python script into a Jupyter notebook

Let's say we've written a very useful Python script that we'd like to run in a Jupyter notebook.

We can run it by doing the following:

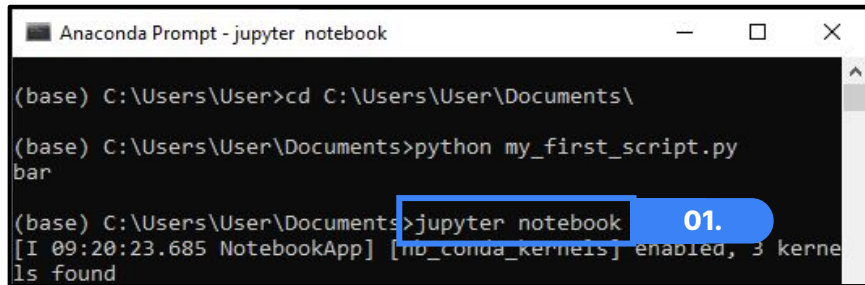
01. Open a command prompt in the same directory as the script and type **Jupyter notebook**, then hit Enter. This will open an instance of Jupyter in our browser.

02. Start a new notebook. In the first cell of the new notebook, type **import my_first_script** and execute the cell by hitting "Shift+Enter".

```
In [1]: import my_first_script
```

03. We can now access the **foo function** in this script by typing **my_first_script.foo()**. Executing this in a new cell will print the word 'bar'.

```
In [2]: my_first_script.foo()  
  
bar
```



The screenshot shows an Anaconda Prompt window titled "Anaconda Prompt - jupyter notebook". The command prompt shows the following sequence of commands and output:

```
(base) C:\Users\User>cd C:\Users\User\Documents\  
(base) C:\Users\User\Documents>python my_first_script.py  
bar  
(base) C:\Users\User\Documents>jupyter notebook 01.  
[I 09:20:23.685 NotebookApp] [no_conda_kernels] enabled, 3 kernels found
```

A blue box highlights the command `jupyter notebook` and the number `01.` next to it.

Tools used to compile and execute Python script



Jupyter Notebook

Medium scale coding operations. About 100 to 200 lines of code as longer-length notebooks become unmanageable.

Can import Python scripts.

Not recommended for a production pipeline.

Useful for presenting and exploring data.

Can execute cells in any order.

Does not take parsed arguments.



Visual Studio Code (IDE)

Large-scale coding operations.
Can cover thousands of lines of code.

Can import other Python scripts.

Can form the backbone of a production pipeline.

Useful for writing large amounts of code.

Always executes from top to bottom.

Can work with parsed arguments.



Command line

Small-scale coding operations.
Usually only a couple of lines.

Can import Python scripts and execute Python scripts.

Not used in production outside of debugging and monitoring.

Useful for executing Python scripts and debugging.

Always executes from top to bottom.

Can parse arguments to a script.

Extra reading

Extending Python scripts with arguments:

Argument parsing

Argument parsing in Python

Python scripting:

Running Python scripts

Python scripts

Visual Code Tips and Tricks:

Visual Code tips and tricks

Python scripts vs modules:

Python modules vs scripts

Reusing code

