

Chapter 1

First steps

1.1 Python distributions

The canonical Python distribution can be found at <https://www.python.org>. It is also known as CPython, as it is implemented using the C programming language. Python 3.8.3 for Windows has a total download size of only 25MB. It comes with several powerful builtin modules. See <https://www.python.org/doc/essays/blurb/> for an official summary of what is Python.

For our scientific and engineering pursuits, we will need to use several external modules, like, *Numpy* for fast array based calculations, and *Matplotlib* for creating beautiful and meaningful graphs of our data. These packages and many others are hosted by the *Python Packaging Index*(PyPI), and they can be installed into Python using the built in package manager *pip*.

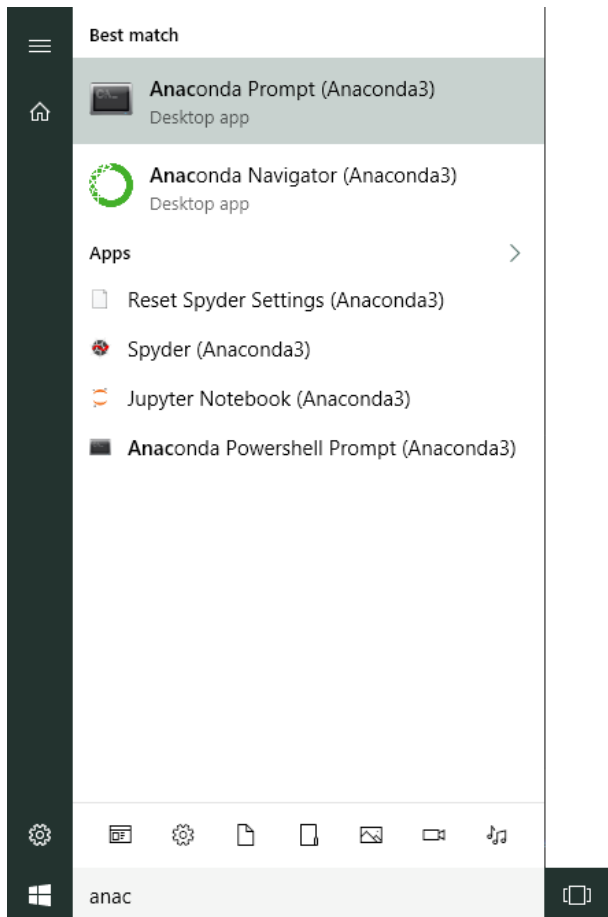
As a beginner, it is hard to find our way in the huge ecosystem that is Python. Fortunately, Anaconda exists. Anaconda is a distribution of Python that comes preinstalled with a large number of popular packages that are used for data science, AI, scientific analysis and computations, statistical modelling and analysis, and the list goes further. Once installed, it gives us a ready-made environment for us to play with.

1.2 Installation

We will look at how to install the latest version of the popular distribution of Python known as Anaconda. The main site is located at <https://www.anaconda.com> which will lead us to the download page at <https://www.anaconda.com/products/individual>. Scrolling to the bottom of the page we will see all the versions of Anaconda for all popular operating systems. We can see Anaconda offering both Python 2 and Python 3 versions. Python 2.7 is very old and has lost support. If we are using Windows 10, we want the 64-bit version. It is about 466MB. Download this installer.

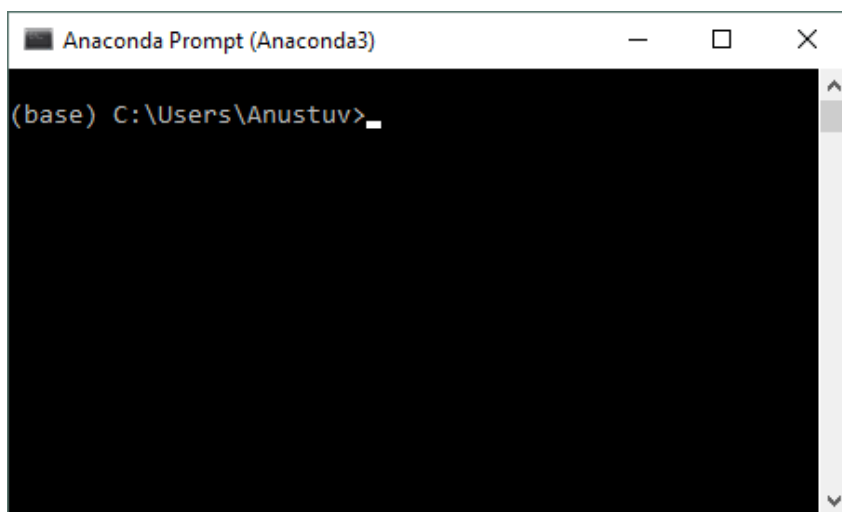
After downloading is finished we can just run the setup and install it like any other software or game. I would recommend following [this tutorial](#) for a guided setup process. It's a little old, but very little should have changed between then and now.

Once Anaconda is installed in our system, we can simply press the Win key start typing anaconda. The windows search will show us something like,



The main executable that Anaconda provides is the Anaconda Navigator. It manages Python environments and versions for us, helps us install and manage packages in them using a graphical interface. It is very useful for the beginner, but it is slower than doing these things from the command line. We will prefer the command line approach from the beginning as it is simpler and allows a much quicker workflow. We will run the Anaconda Prompt, as shown in the above image.

When we run the Anaconda Prompt, we should see something like this,



1.3 Usage

In the command line,

```
(base) C:\Users\Anustuv>_
```

The `(base)` indicates that the default Python environment provided by Anaconda is active.

The `C:\Users\Anustuv>` shows the current working directory.

There are two primary ways to run programs in Python. First is the interactive Python interpreter, which is a *REPL*, short for Read-Print-Eval-Loop. Second is to run Python script files, let us look at the various ways to execute Python programs.

1.3.1 The default interpreter

Running Python here is as simple as calling `python`, and we will see the interactive Python interpreter. Let us run the customary "Hello world" program at the terminal.

```
(base) C:\Users\Anustuv>
(base) C:\Users\Anustuv>python
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64
  bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more
  information.
>>> print("Hello world")
Hello World
>>> _
```

This application is the default interactive console for python. We can use Python interactively in this interface. The `>>>` indicates the Python prompt. We can type any python command, or statement, or expression, and Python will evaluate it and output the result. Try various arithmetic expressions,

```
>>> print("Hello world")
Hello World
>>> 3 + 4
7
>>> 7 - 2
5
>>> a = 5
>>> a ** 2
25
>>> exit()

(base) C:\Users\Anustuv>_
```

We will call the `exit()` function to exit the interpreter and return control to the OS.

1.3.2 The IPython interpreter

First, run the `ipython` application at the prompt. This is an improved interpreter that gives various features over the Python interpreter. For all intents and purposes, this is a Python interpreter.

```
(base) C:\Users\Anustuv>ipython
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64
  bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for
  help.

In [1]: _
```

1.3.3 Running scripts and IDLE

The interactive interpreters are a good environment to try out various ideas and small snippets of code. However, we will most likely want to run large algorithms that require many lines of code. To do this we can write our Python program in a text file with the `.py` extension.

We can use any text editor or *IDE* (Integrated Development Environment) software to create our Python script files. Here, let us use the inbuilt Python text editor known as *IDLE*. To create a new script file called `helloworld.py` run at the terminal

```
(base) C:\Users\Anustuv>idle helloworld.py
```

A simple text editor opens up, type `print("Hello World")`, save the file, then exit. The cursor comes back to the command line. Running this script is as simple as,

```
(base) C:\Users\Anustuv>python helloworld.py
Hello World

(base) C:\Users\Anustuv>_
```

Here we must also mention that, IDLE isn't just a text editor. Run the `idle` command without a filename suffix, and a windows based Python interpreter opens up. In this environment we can both create script files and run them, and interactively compute on those results.

```
Python 3.7.5 Shell
Python 3.7.5 (default, Oct 31 2019, 15:18:51)
[MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "licens
e()" for more information.
>>>
== RESTART: C:\Users\Anustuv\OurNotebooks\sine
example.py ==
Sine of 120 = 0.8660254037844387
>>>
```

```
sineexample.py - C:\Users\Anustuv\OurNotebooks\sineexample.py (3.7.5)
from math import sin, radians
angle = 120
rad = radians(angle)
print("Sine of", angle, "=", sin(rad))
```

1.3.4 Jupyter notebooks

We very briefly invoked both the standard interpreter and the ipython interpreter. There exists another interpreter, known as the Jupyter Notebook. According to jupyter.com,

"The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more."

Before we launch jupyter lets make a new folder where our notebook files will be saved, and change our working directory to that path, like so,

```
(base) C:\Users\Anustuv>mkdir OurNotebooks
```

```
(base) C:\Users\Anustuv>cd OurNotebooks
```

From this path we run the `jupyter notebook` command, and that starts the Jupyter local web server.

```
(base) C:\Users\Anustuv\OurNotebooks>jupyter notebook
[I 13:28:45.608 NotebookApp] Serving notebooks from local
  directory: C:\Users\Anustuv\OurNotebooks
[I 13:28:45.610 NotebookApp] The Jupyter Notebook is running
  at:
[I 13:28:45.611 NotebookApp] http://localhost:8888/?token=
  da457d6ccb4f732621d37826e44ea7f6c9e021d05db3cb98
[I 13:28:45.611 NotebookApp] or http://127.0.0.1:8888/?token=
  da457d6ccb4f732621d37826e44ea7f6c9e021d05db3cb98
[I 13:28:45.611 NotebookApp] Use Control-C to stop this server
  and shut down all kernels (twice to skip confirmation).
[C 13:28:45.756 NotebookApp]
```

To access the notebook, open this file in a browser:

```
file:///C:/Users/Anustuv/AppData/Roaming/jupyter/runtime/
  nbserver-13208-open.html
```

Or copy and paste one of these URLs:

```
http://localhost:8888/?token=
  da457d6ccb4f732621d37826e44ea7f6c9e021d05db3cb98
or http://127.0.0.1:8888/?token=
  da457d6ccb4f732621d37826e44ea7f6c9e021d05db3cb98
```

When the Jupyter webserver runs, our internet browser should automatically launch the jupyter service in a new tab, if it doesnt, then we can paste the address shown in the above message in the browser address bar, and that should open the jupyter main page. This is what it should look like,

Select items to perform actions on them.

Upload

New ▾



<input type="checkbox"/> 0 ▾	📁 /	Name ▾	Last Modified	File size
<input type="checkbox"/>	📄 helloworld.py		32 minutes ago	22 B
<input type="checkbox"/>	📄 sineexample.py		22 minutes ago	108 B

We can open a new jupyter notebook from the New menu,

Select items to perform actions on them.

Upload

New ▾



<input type="checkbox"/> 0 ▾	📁 /	Name ▾	Last Modified	File size
<input type="checkbox"/>	📄 helloworld.py			
<input type="checkbox"/>	📄 sineexample.py			

Notebook:

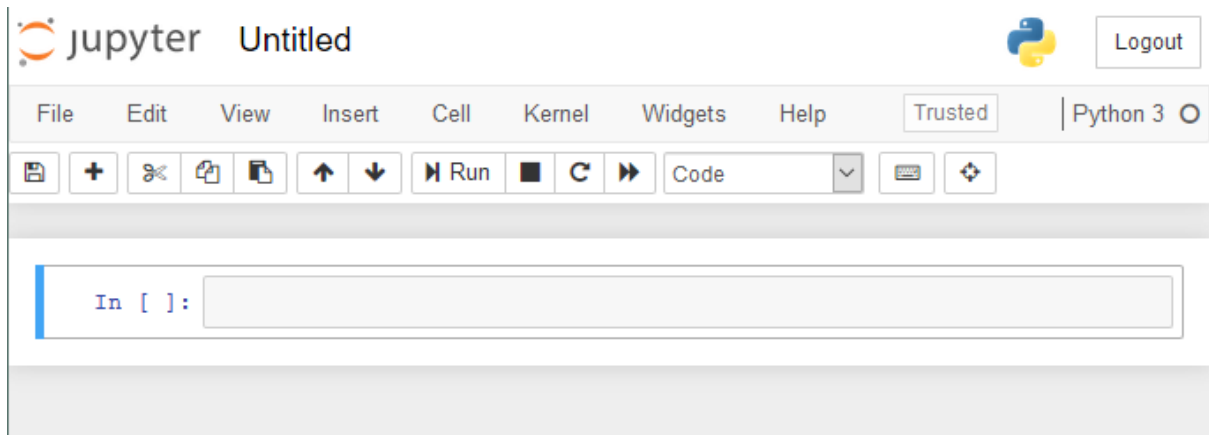
- Python 3

Other:

- Text File
- Folder
- Terminal

Create a new notebook

And that will open a notebook in a new tab.



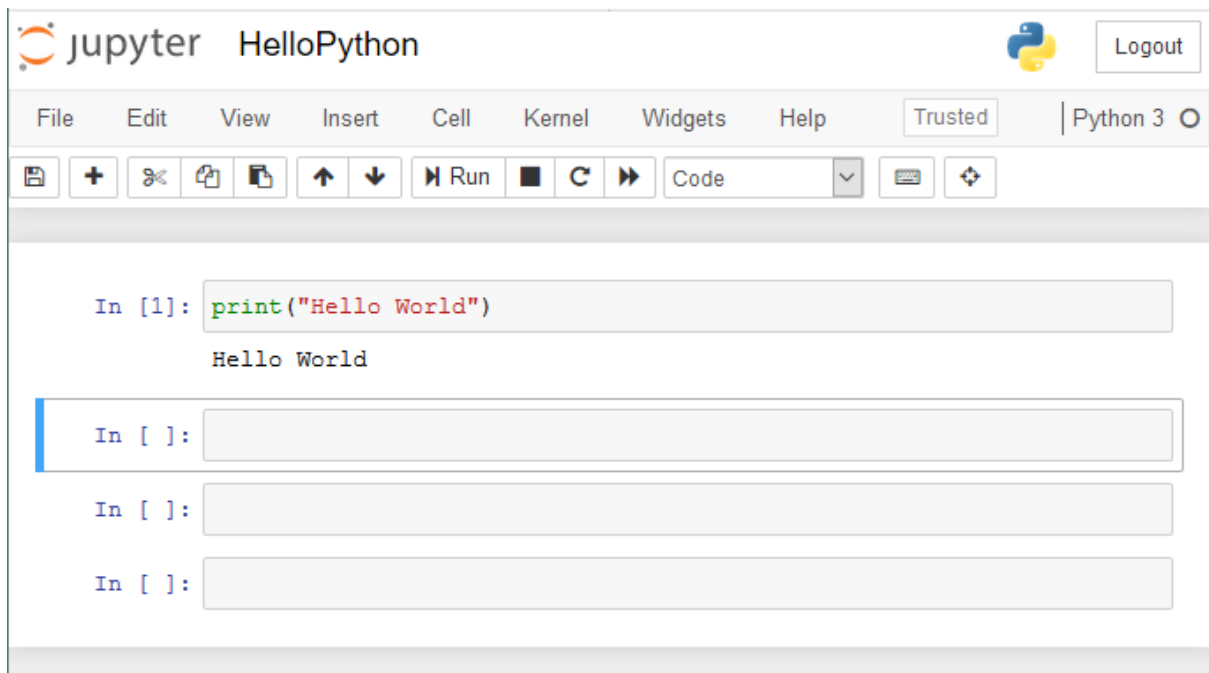
If you managed to get till here, Congratulations! You are now ready to play around with Python.

1.4 Navigating the notebook

In the above figure, you can see the Jupyter logo at the top left corner, beside it is the word "Untitled". That is the document name. Click on it and change it to something useful, "HelloPython" for example. Observe that in the Jupyter main page, this change is reflected.

Press the `B` key a few times. A number of input cells appear. We can navigate the selection of a cell using the `Up` and `Down` keys. Pressing the `Enter` key, the selected cell frame turns green and the cursor appears. Now we are in the *edit mode*. Press the `Esc` key, and the frame turns blue again, this is the *command mode*.

Navigate to the first cell and get into edit mode. Let's enter some code. At this point we don't know much code beyond something like, `print("Short text.")`, so just do that. Once done, press `Ctrl+Enter`, and the current cell gets executed. Get back into the edit mode and change the string to something else. Now press `Shift+Enter`. The cell gets executed again and we see the output. This time, the selection moves to the next cell as well. These are two ways of executing any cell. You should have something like this,




In the command mode, pressing H brings up a helpful window showing us all the available keyboard shortcuts. Examples of some useful shortcuts in the command mode are,

Keystroke	Function
A and B	Insert new cell above and below respectively.
C	copy a cell and its contents.
X	cut a cell and its contents.
V	paste the cells in the clipboard below the current cell.
D, D	Pressing D twice deletes the current cell. Twice, so we dont delete it accidentally.
P	brings up the <i>command palette</i> . It shows a searchable menu of every action that jupyter can make, along with their available shortcuts.
Y, M, and R	respectively convert the current cell to a normal code cell, a markdown cell, or a raw cell.

Markdown is a simple scheme to format text. We can use markdown cells to tell stories about our code and the outputs and plots that we generate. It can be used to display headings, paragraphs, images, and well formatted mathematical equations. Markdown is very simple and can be learnt within minutes.

In the edit mode, particularly useful are,

Keystroke	Function
Tab	inserts indention, or <i>code completion</i> menu if available.
Ctrl+/ Ctrl+] and Ctrl+[comments or uncomments a line. Useful to temporarily disable a line of code. indents or dedents a line or selection of lines. This is useful if we want to put an existing procedure under a block, or bring it outside a block.
Ctrl+X and Ctrl+C	behave like normal, but without a text selection, it cuts or copies the entire current line.

Few more things: The menu bar and the toolbar on the top also allow us to do these things using mouse interaction. Noteworthy are the three buttons . In order, they are used to stop the running Python kernel if our code is stuck or taking too long, to restart the Python kernel, and to restart the kernel and run all the cells in the notebook.

The current notebook is intermittently saved to disk by the jupyter server so that our changes are never lost. We can also save the notebook in various different formats from the *File>Download as* menu.

When our work is done and we need to close everything, we can start by closing the browser tabs of our notebooks and main page. Then we go to the terminal that is serving our jupyter app, and press `Ctrl + C`, the server will terminate with messages like,

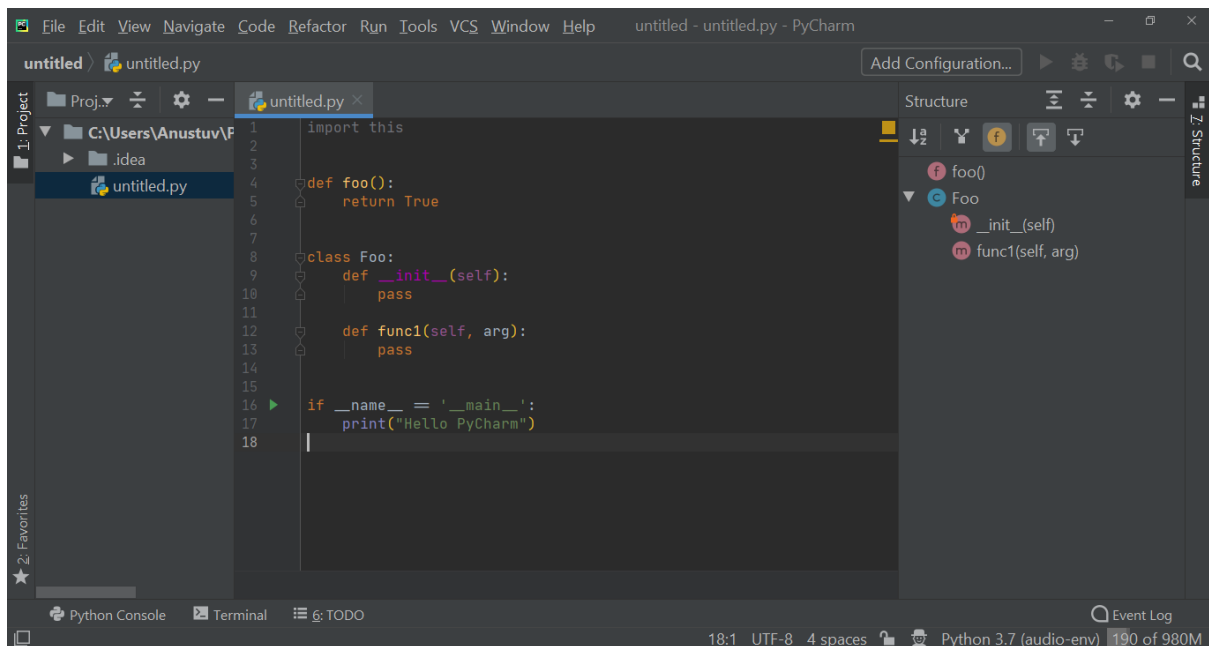
```
[I 10:39:01.303 NotebookApp] Interrupted...
[I 10:39:01.678 NotebookApp] Shutting down 1 kernel
[I 10:39:06.057 NotebookApp] Kernel shutdown: 1832a1ce-6b91
-455b-87b0-836952828345

(apenv1) C:\Users\Anustuv\OurNotebooks>_
```

and the control is returned to the operating system.

1.5 PyCharm

This article wouldn't be complete without mentioning PyCharm, specially the Community Edition, freely available to download at <https://www.jetbrains.com/pycharm/>. It is a Python IDE made by JetBrains. Following is an example of what PyCharm looks like. Don't worry about the displayed code.



PyCharm is extremely feature-rich and extendable with freely available plugins. It provides features like intelligent code assistant, built in developer tools, and some scientific tools. It makes the process of debugging and maintaining large projects easy, it integrates with external

tools like version control systems, and much more. It would take a few chapters to satisfactorily describe an effective workflow in PyCharm and its many tools and features. We will gradually start using PyCharm in this course as our projects get bigger.

1.6 Summary

In this article we learnt about two major distributions of the Python programming language and how to get it installed in our system. We learnt about the standard interpreter, the IPython interactive interpreter, and IDLE. We also learnt that an amazing IDE called PyCharm exists that can make our large projects easier to handle.

We also learnt how to do simple things like create a folder and navigate to that folder using the MS-DOS command line. If you are using a Linux or a Mac operating system, everything should pretty much still work the same, with very few changes.

We also observed some preliminary Python code in action. We learnt that `print()` function can be used to display text and numbers on the screen. We also did some simple calculations at the interpreter, and also learnt that `**` is the mathematical raise to the power operator.

We went in depth into the Jupyter notebook environment and its features, and learnt to navigate it in an agile manner. Knowing these preliminary things, we are now ready at the start line. All we now need are problems to solve along with some imagination and creativity.

Caveat

We saw how to install Python and use it in various ways in a Windows OS and the DOS command line. The entire process is nearly identical in other operating systems like Linux or Mac OS.

There are many other methods to use and program in Python. Various IDEs like Spyder, Atom, Eclipse and others provide smooth workflows in Python. Python can also be used online, for instance Google Colab, or Kaggle.

This is just the beginning, just to get started. Everything lies ahead!