# Installation and Instruction Guide for Python and its Dependencies

This installation guide is to help set you up with all the tools needed to carry out your own Python projects.

**Section 1**contains a glossary of terms.

**Section 2**describes the installation process for Anaconda, and therefore Python, and the setting up of a virtual environment.

**Section 3**briefly describes how to install and use Jupyter Notebooks.

**Section 4**describes how to install packages using either the Anaconda Prompt if you are on Windows or the command line if you are using Mac.

## Section 1 - Glossary

## Python

The programming language. It is very versatile supporting areas as diverse as data science and cheminformatics to website design and internet of things. Python has a very active community who develop ‘packages’ – software that is built to solve certain problems which can be imported into your own Python scripts.

#### Packages/Modules/libraries

Software that has a specific functionality. For example, the Numerical Python package (numpy) is a fundamental package for scientific computing. Numpy can be used to simply manipulate data, build arrays or run calculations. Packages are imported into any python script you make, you then use their functionality saving you the time of coding it yourself.

#### Anaconda

Anaconda can be thought of as a one stop shop for all things Python and data science. It’s almost an umbrella; by downloading Anaconda you also get Python, Jupyter notebooks and a whole host of other useful programmes. It is also a package manager and virtual environment manager.

#### Virtual environment

You may have multiple Python projects happening at once. A virtual environment is a named, isolated copy of Python that has its own files and packages. Anything done inside a virtual environment is specific to that environment and won’t affect any other projects. All the dependencies and packages you install will be sandboxed inside the environment you are using. You should therefore have a seperate virtual environment for each seperate project.

#### Jupyter Notebook

Comes with Anaconda. Jupyter (formerly IPython Notebook) is an open-source project that lets you easily combine markdown text and executable Python source code on one canvas called a notebook. They are both human readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which run the code to perform the data analysis.

#### Terminal/Command Line

A command-line interface processes commands to a computer program in the form of lines of text. The program which handles the interface is called a command-line interpreter or command-line processor. Put simply, you can type code (commands) into the terminal which will effect changes on your computer. It is often much faster, and much easier to understand, than using your mouse to perform similar tasks

#### Anaconda Prompt

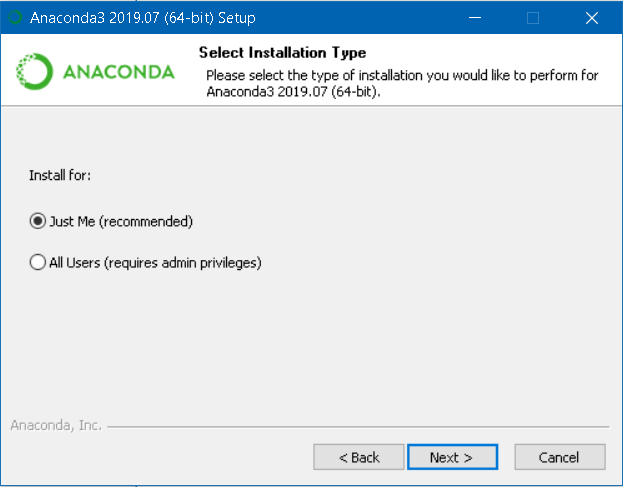
The Anaconda prompt brings all the functionality and benefits of the Terminal described above to Windows. It allows you to enter written commands (code) to effect changes specifially to programmes controlled by Anaconda. It makes installing libraries and setting up Virtual Environments very easy.

## Section 2 – Installing Python through the Anaconda Navigator

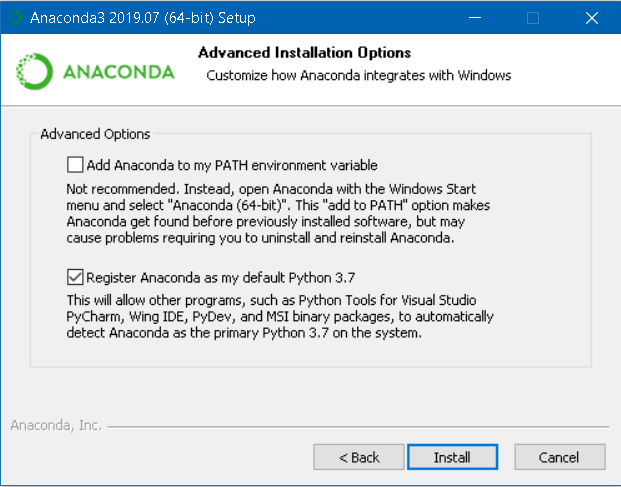
#### Installation

The easiest way to install Python is through the package and virtual environment manager Anaconda. Anaconda is free and open source and the Individual Edition can be installed from <https://www.anaconda.com/>.

Make sure to install the correct version of Anaconda based on your operating system (Windows, Mac etc). You must also download it with Python 3.7, this is the most up to date version. When prompted to choose who to install it for, choose 'Just Me' which is the recommended option.



Leave the installation location as the default one and then register Anaconda as you default Python 3.7.



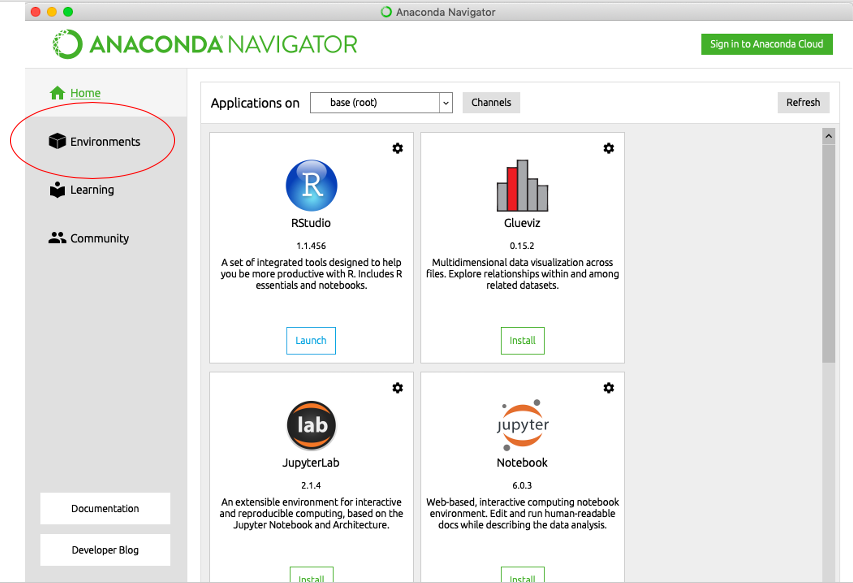
Build from source - Only applicable if you need a specific Python build

To build Python from source it can be downloaded from <https://www.python.org/>. This is the homepage for Python and as such contains documentations, user guides and an active community. Only install this way if you want a specific Python build.

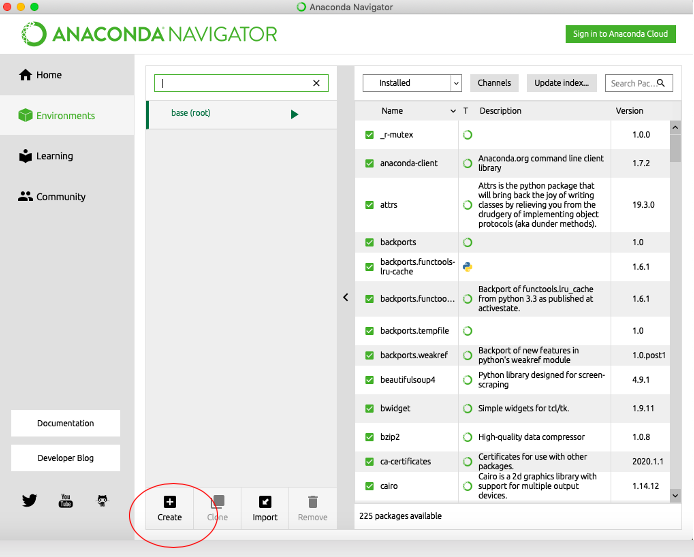
#### Using Anaconda

A virtual environment should now be created. This should be done for each new project that is started. Skip to section 4 for instructions on how to do this from the command line.

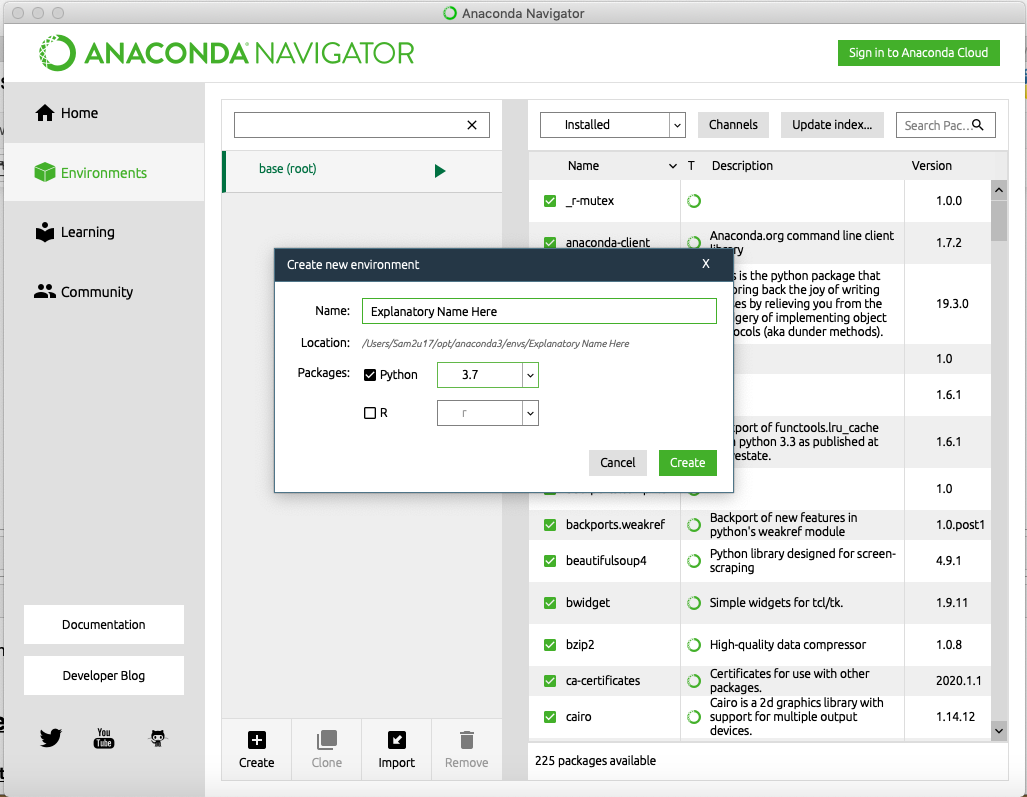
1. In the Anaconda navigator select the Environments tab from the menu options on the left. You will see a screen come up that has a list of environments and the packages they contain. On your first go there will only be the ‘base (root)’ environment.



1. At the bottom of the section that contains the environment names, choose ‘create’.



1. A pop-up box will appear, give your environment a descriptive name and choose Python 3.7 from the drop-down menu.



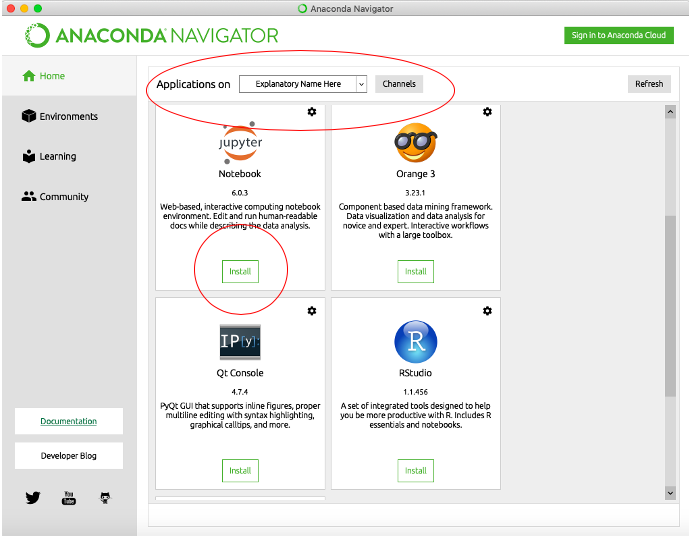
Congratulations, you have now created a virtual environment manually using the Anaconda environment manager!

## Section 3 – Installing and Using Jupyter Notebooks

#### Installing Jupyter Notebooks

Now that an environment has been made it is time install the tools needed to carry out the project. This section will describe how to quickly install Jupyter Notebooks using the Anaconda Navigator. For information on how to get the rest of the packages installed via the command line refer to **Section 4**.

Navigate back to the home screen using the menu on the left. From here select the text ‘Applications on’ followed by a drop-down menu. Make sure you select the environment you have just created and then install Jupyter Notebooks.



You can now run the environment in two ways:

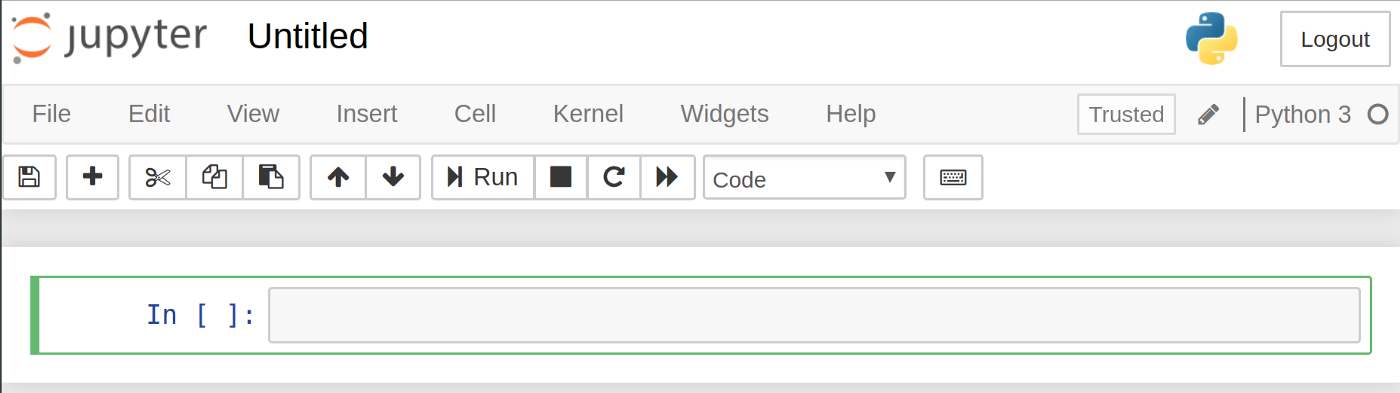
1) Launch directly from the Jupyter Notebook you have just installed

2) Navigate to the Environments tab. Press the play button next to your environment and select ‘Open with Jupyter Notebook’ from the drop-down menu

Congratulations! You now have an environment set up and can start programming in Jupyter Notebooks!

#### Using Jupyter Notebooks

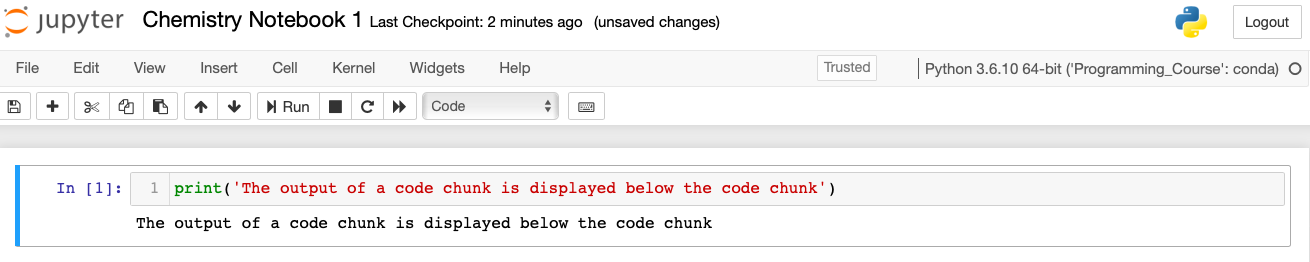
A Jupyter Notebook is tool that allows you to interactively develop and present Python projects. Jupyter Notebooks integrate markdown text, code and code output into a single document. This means that they contain the text, mathematical equations, and graphs that the code produces directly in the same page. In this way they promote a a fast iterative development as each section of code can be displayed immediately.



The default notebook is displaye as above. It is currently 'Untitled' as we haven't given it a title yet. To name it something descriptive, hover over the current name and click. We'll name ours 'Chemistry Notebook 1'.

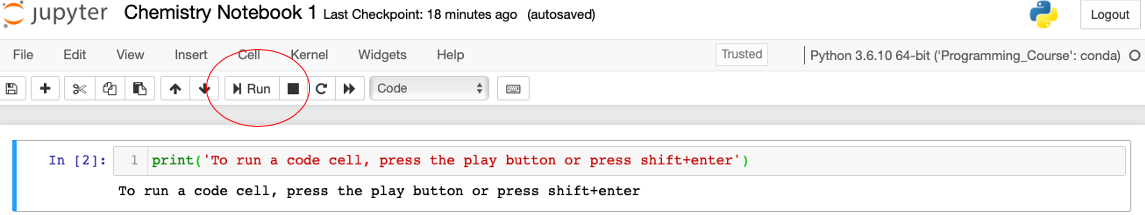


The In[ ] specifies where you type your input. The output of the code is displayed directly below it as demonstrated below.



The In[ ] now has a 1 in it to indicate the order in which the cells have been ran. So as this was our first cell it becomes In [1], however if we were to go on and run 3 more cells before re-running this cell it would become In[5], as it was now run fifth.

To run a code chunk you may either press the play button, circled in the picture below, or you can press 'shift+enter'. This will execute any code in a code cell, or it will print to screen writing in markdown cell.



Try running the code cell below. Notice how the number appears in the In[ ] to let you know that it has been run and in what order.

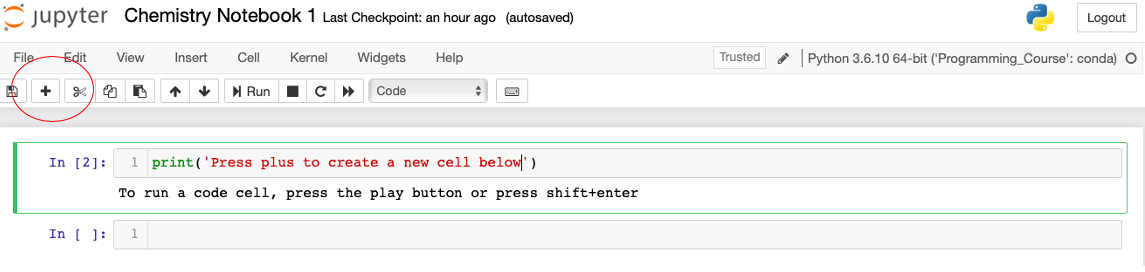
In [5]:

print('Run this code chunk to see how it works for yourself!')

Run this code chunk in your own Jupyter notebook to see how it works for yourself!

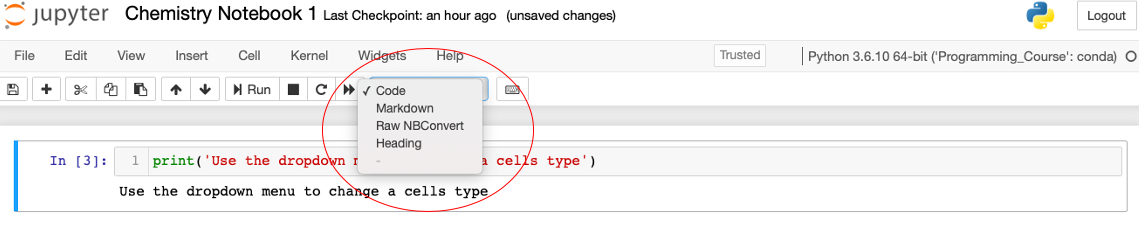
So far we have covered how to give your notebook a title and how to run pieces of code. However, we haven't covered one of Jupyters biggest advantages: its markdown capabilities. The markdown sections contain all the text you're currently reading aswell as all the pictures. These sections allow for code, or results, to be properly explained allowing for a whole report to be written in one place.

The first thing we need to do to create our markdown cell is to create a new cell. Do this by pressing the plus button in the ribbon at the top, as shown below.

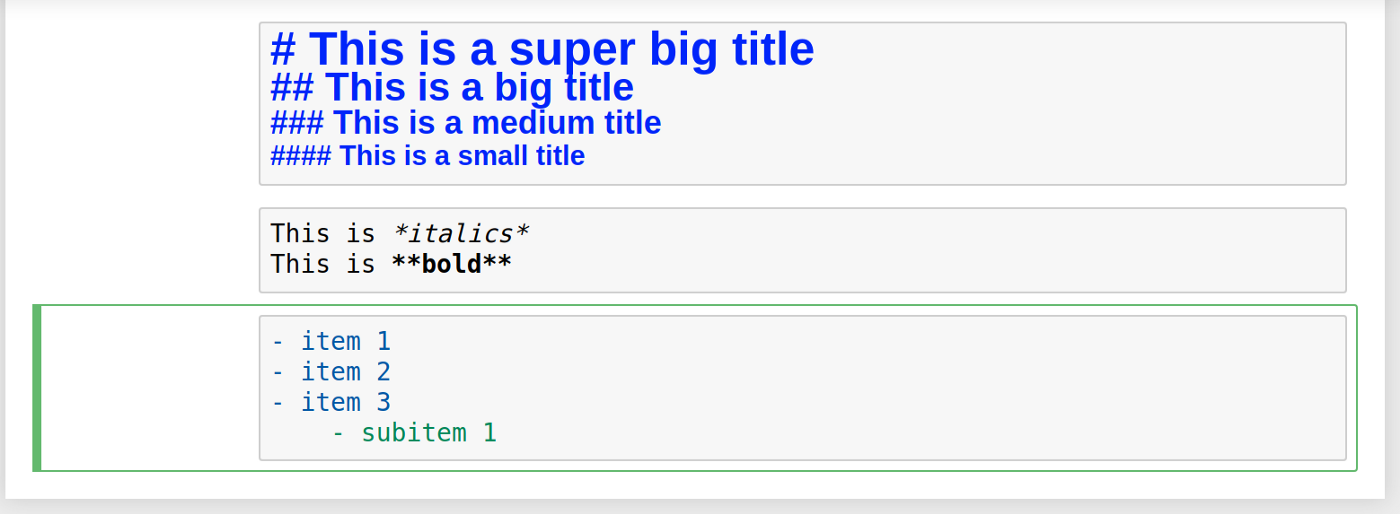


Try adding your own code chunk here in your own Jupyter Notebook!

We now need to convert the cell we've just created from a code cell to a markdown one. This can be done by entering the cell and pressing 'esc+m' or by using the dropdown menu at the top of the screen.



Once you’ve made a mardown cell there are a few shorcuts that will be useful. Titles can be made by using the # key, the number of #'s you use determine the size of the heading. Stars are used for italics and bold font.



As you have probably seen there are plenty of other tabs in the notebooks as well as a ton more functionality. we'll describe what a few of the menus do here and then leave the rest for you to discover by yourselves as and when you need them.

**File**: This allows you to create, copy, rename, and save your notebooks to file. The most notable item in the File menu is the Download as drop down menu which lets you download your notebook in a variety of formats including pdf, html, and slides — perfect for creating a presentation!

**Edit**: This menu lets you do the good’ol can cut, copy, and paste of code. You can also reorder cells here, perhaps if you’re creating a notebook for an interactive presentation and want to show your audience things in a certain order.

**View**: This menu lets you play around with things like displaying line numbers and modifying the toolbar. The best feature in this menu is definitely the Cell Toolbar where you can add tags, notes, and attachments to each cell. You can even select the formatting you would want for this cell if you turned the notebook into a slide show!

**Insert**: This is just for inserting cells above or below the currently selected cell. The Cell menu is where you go to run your cells in a specific order or change the cell type.

**Help**: This menu gives you direct access to important documentation. You’ll be able to learn about all the Jupyter Notebook shortcuts to speed up your workflow. You also get convenient links to the documentation of some of the most important Python libraries including Numpy, Scipy, Matplotlib, and Pandas!

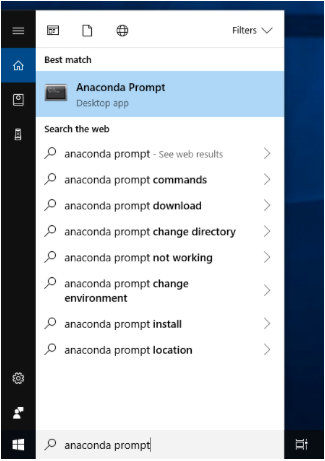
## Section 4 – Using Anaconda via the Command Line

The Anaconda navigator can be used to create environments and install packages manually, however not all modules are available through anaconda and it is often more efficient to create an environment using the command line instead.

If you are using a windows machine you will need to bring up the Anaconda Prompt, if you are using Mac you need to bring up the Terminal.

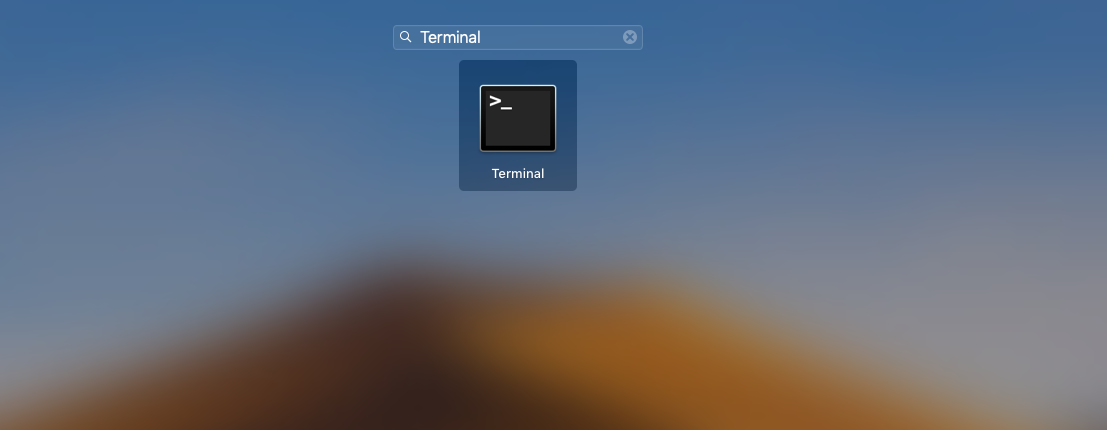
#### Opening Anaconda Command on Windows

To get the Anaconda Prompt open in Windows, open the start menu and type 'Annaconda Prompt'. This will bring up a window for, surprisingly, the Anaconda Prompt. From here you can input instructions directly to the computer.



#### Opening Terminal on Mac

To get the Terminal open on mac, go onto Launchpad and type 'Terminal'. Again, this will allow you to input instructions directly into the computer.



#### Managing Environments

The commands for creating and loading environments are exactly the same in both Windows and Mac. The first thing we need to do is create a new environment and set the version of Python that we want the environment to run. This can be achieved with one command.

picture

Breaking this down:

1) We first tell Anaconda that we are creating a new environment with the 'conda create' command.   
2) Secondly, we tell Anaconda that we want to name the environment with the '--name' command   
3) We name the environment, in this case we have chosen 'my\_first\_env'   
4) Set the version of Python we wish to use in the environment, in this case Python 3.6

Once this command has been executed, Conda will ask you if you want to proceed. Type Y and press enter.

You can now check you have created the environment successfully, or you can check other environments you have previously made in the following way:

picture

You should now see the environment you have just created listed in the output.

#### Launching Environments

We now need to run the environment that has just been made. To do this type:

picture

'conda activate' lets Anaconda know that we want to run an environment. The part that comes after, in this case 'my\_first\_env' is simply the name of the environment that needs to be run.

#### Closing Environments

To deactivate an environment we do the opposite of the command above:

picture

Installing Packages

Now that the environment is set up, it needs to be populated with useful packages and libraries. This is exactly like when we installed Jupyter Notebooks in section 3, however instead of doing it through the Anaconda Navigator we will do it through the command line.

There are two main ways of installing packages, via conda or via pip. Pip is simply another package manager for Python, a bit like Anaconda, though it doesnt come with all the bells and whistles. It is good practise to always try and install a package via conda, and then only if it is not available this way, then use pip.

The commands are very similar - 'conda install' and 'pip install'

Here, we demonstrate how to install Numpy (numerical Python) via conda and pip.

picture

picture

And it’s as simple as that. Press enter when prompted and any package that you need will be downloaded into your environment!