

## Chapter 2: Installing Python and packages

After reading this chapter, readers will be able to:

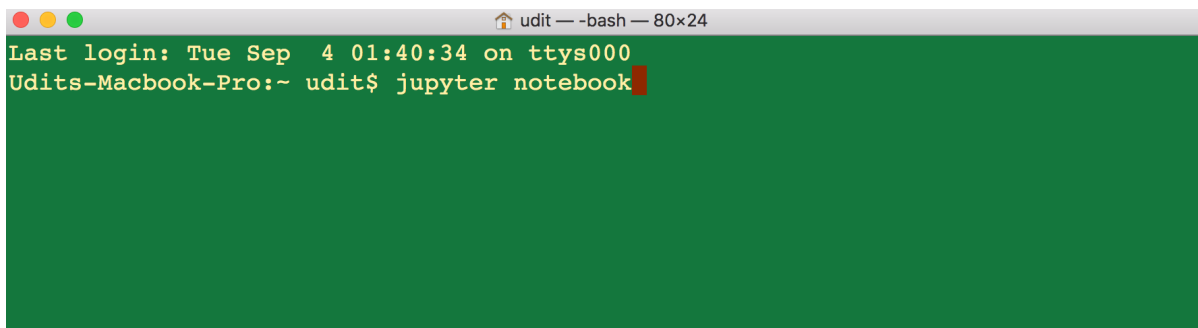
1. Install Python via Anaconda on MacOS and Windows.
2. Launch Jupyter Notebook and Python kernel.
3. Understand the difference between modules and packages.
4. Install Python packages.
5. Troubleshoot package installation errors

If you have not installed Python via anaconda, it is strongly recommended to install Python 3.6 via Anaconda: <https://www.anaconda.com/download/> (<https://www.anaconda.com/download/>). Anaconda is a free and open source distribution of the Python programming language for data science and machine learning related applications. It is great tool to simplify package installation and management. We will discuss about packages in detail as we move forward. Another advantage of installing Python via Anaconda is that it installs a multi-utility Integrated Development Environment (IDE) called Jupyter Notebook that provides comprehensive facilities to to programmers for writing and running codes.

### 2.1 Launching Jupyter Notebook

#### a) MacOS/Linux/UNIX users

Once the software successfully installs, press Command (⌘) and Spacebar in MacOS, type Terminal and press return key. This launches the command line in MacOS. In command line, type **jupyter notebook** and press enter.

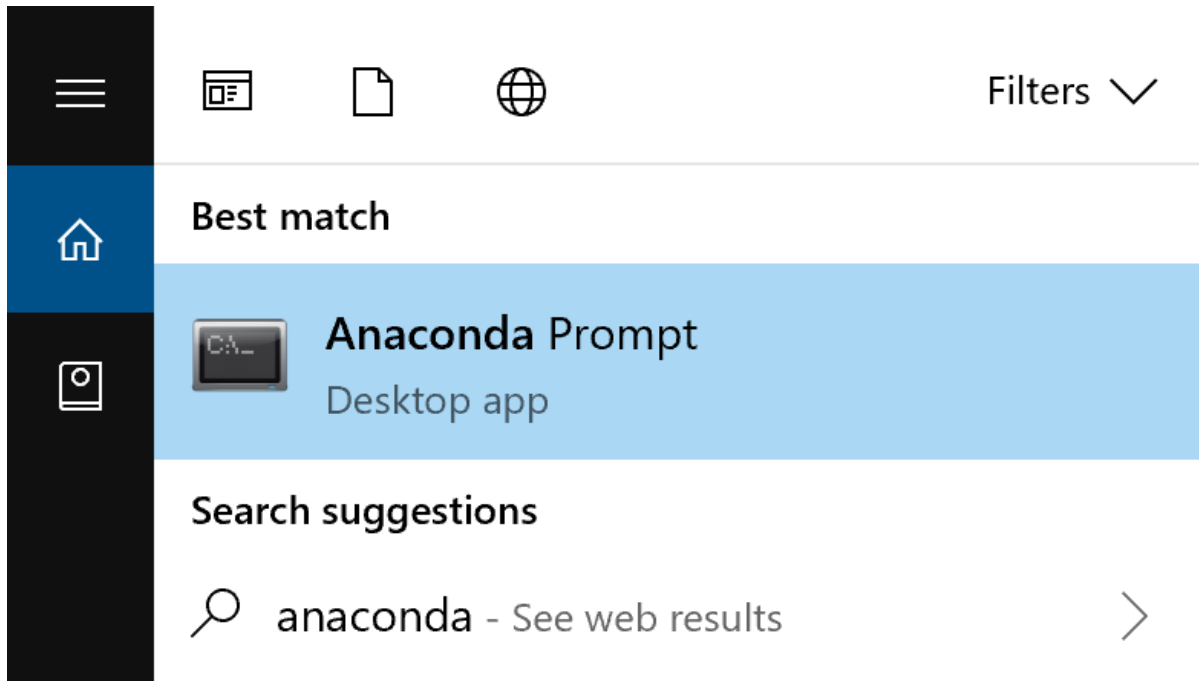


```
udit — -bash — 80x24
Last login: Tue Sep  4 01:40:34 on ttys000
Udits-Macbook-Pro:~ udit$ jupyter notebook
```

This will launch the Jupyter Notebook IDE in your default web browser. Please note that although Notebook launches in web browser, it does not require active Internet connection to function. Do not close Terminal when you are working on jupyter notebook.

## b) Windows users

Launch Anaconda Prompt from Start Menu.



In Anaconda prompt, type **jupyter notebook** and press enter. This will launch the Jupyter Notebook IDE in your default web browser. Please note that although Notebook launches in web browser, it does not require active Internet connection to function. Do not close the prompt when you are working on jupyter notebook.

## 2.2 Launching Kernel (both MacOS and Windows)

To launch a Python Kernel, click **New** on the right side and select **python3** from the dropdown list (See below). This will launch the kernel where you can write and run your code. Note that Python kernel is a program that runs and introspects the user's code.



Now, we are all set to write and run the code! However, before start writing any meaningful codes, it is important to understand the pivotal role of **Packages** and **modules** in Python.

## 2.3 Packages and modules

**Module:** In Python, **module** is a piece of software that has a specific functionality. For example, while analyzing financial time series of a certain company (say stock price in last one year), you are required to compute average stock price in a given week. To do the **averaging** operation, you will require a **module** that has set of commands to calculate average of numbers.

**Package:** On the other hand, a **package** is collection of **Python modules**. A **package** directory (or folder) always contains a file with extension `__init__.py`. If this file is present, Python recognises this folder as a special folder.

Let us go back to our Financial time-series example. In addition to performing averages, you want to calculate variance (or standard deviation), trends, seasonality, etc. Hence, you will create a **package** that will contain various **modules** to compute average, trends, seasonality, etc. Figure below shows how sample **package** looks like. **.py** files are individual **modules** within the package.

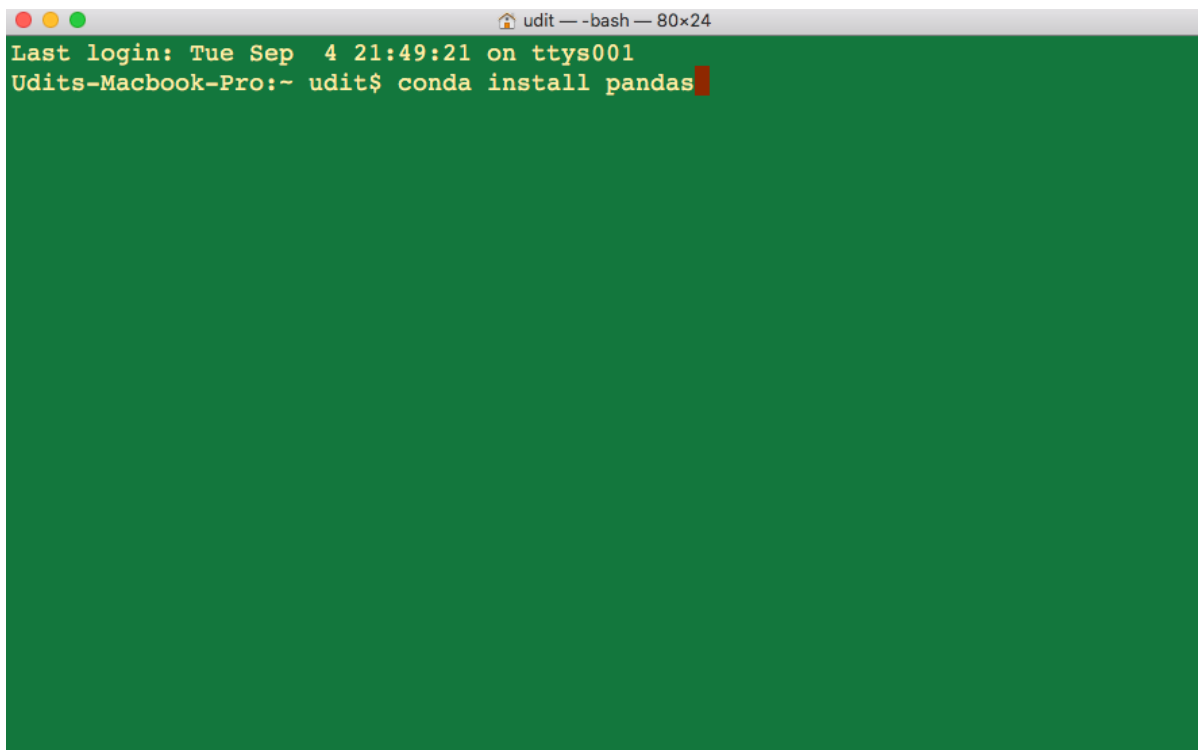
Name	^	Date Modified	Size	Kind
__init__.py		Today at 10:07 PM	Zero bytes	Python
__pycache__		Today at 10:09 PM	--	Folder
average.py		Today at 10:07 PM	Zero bytes	Python
seasonality.py		Today at 10:07 PM	Zero bytes	Python
trend.py		Today at 10:07 PM	Zero bytes	Python

Fortunately, we need not write packages for most of the tasks that we want to do via Python. Since Python is an open-source language (See chapter 1), developers across the globe contribute packages to do various tasks ranging from averaging to spatial analysis. We can install these packages via Anaconda and **import** these into Python environment. A few packages that we will use often includes **pandas** for data analysis , **numpy**: to handle arrays, and **os**: to interact with operating system via Python.

## 2.4 Installing Packages: requires Internet connection

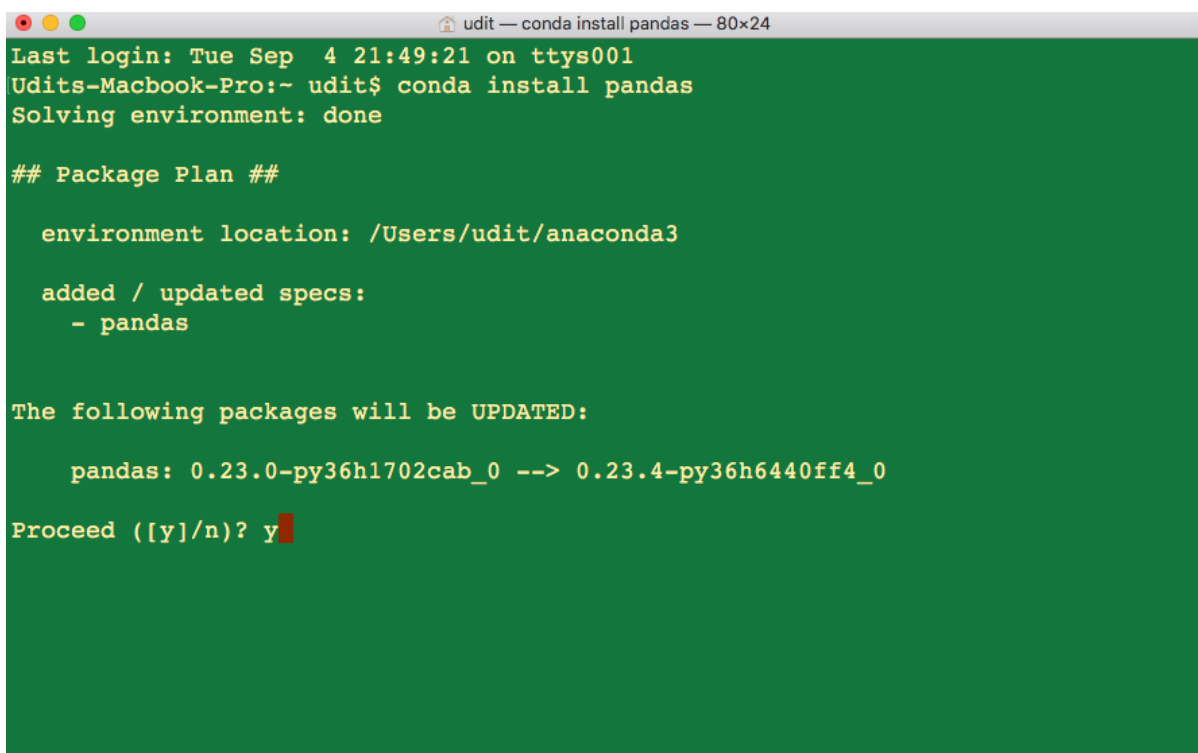
### a) MacOS/Linux/UNIX users

Press Command (⌘) and Spacebar in MacOS, type Terminal and press return key. If your Terminal is already running an instance of Python, you can press (⌘)+N to launch a new instance of terminal. In command line, type **conda install package\_name** and press enter. For example to install **pandas**, type **conda install pandas** and press enter.

A terminal window titled 'udit - bash - 80x24' with a green background. The text shows 'Last login: Tue Sep 4 21:49:21 on ttys001' and the command 'Udits-Macbook-Pro:~ udit\$ conda install pandas' being entered at the prompt. A red cursor is visible at the end of the command.

```
udit — bash — 80x24
Last login: Tue Sep 4 21:49:21 on ttys001
Udits-Macbook-Pro:~ udit$ conda install pandas
```

Follow the prompt and package will be installed. (In this case, press [Y] followed by enter)

A terminal window titled 'udit — conda install pandas — 80x24' with a green background. It shows the output of the 'conda install pandas' command. The text includes 'Solving environment: done', a package plan summary, and a confirmation prompt 'Proceed ([y]/n)? y' with a red cursor at the end.

```
udit — conda install pandas — 80x24
Last login: Tue Sep 4 21:49:21 on ttys001
Udits-Macbook-Pro:~ udit$ conda install pandas
Solving environment: done

## Package Plan ##

  environment location: /Users/udit/anaconda3

  added / updated specs:
    - pandas

The following packages will be UPDATED:

  pandas: 0.23.0-py36h1702cab_0 --> 0.23.4-py36h6440ff4_0

Proceed ([y]/n)? y
```

## b) Windows users

Launch Anaconda Prompt from Start Menu. In Anaconda Prompt, type ***conda install package\_name*** and press enter. For example to install ***pandas***, type ***conda install pandas*** and press enter. This will install the required package. You can install various packages by

following the same procedure. It is noted that packages are required to be installed only once and these remain on your system until you uninstall these or uninstall Anaconda from your system. Once packages are installed, these are ready to be imported into python.

```
(base) C:\Users\udit>conda install pandas
```

## 2.5 Importing packages

Launch Python 3 via Jupyter Notebook (See Section 2.1 and 2.2). In the Notebook, type import pandas and press Shift+return. Note that when you press (shift+return), square bracket changes to an asterix (implying that code is running). Once code has run, asterix changes to number (implying that code has been executed). You can import various packages that you intend to use. For example, in this example code, two packages, pandas and numpy are imported.

In [3]:

```
import pandas
import numpy
```

## 2.6 Troubleshooting errors while importing packages

When you try to import a package that is either not installed or is misspelled, python returns an error. For example, let us try to import a package named xarray and see what prompt we get:

```
In [1]: import xarray

-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-1-45e55b6321f5> in <module>()
----> 1 import xarray

ModuleNotFoundError: No module named 'xarray'
```

**ModuleNotFoundError: No module named 'xarray'** precisely tell us that xarray package is missing and we get ModuleNotFoundError. To undo this, we go to Terminal (or Anaconda Prompt in Windows), launch a new instance ((⌘)+N), and type **conda install xarray**. Follow the prompts and intallation should be complete.

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(python_manual) Udits-Macbook-Pro:python_tutorials udit$ conda install xarray
```

Try import xarray again, and code should run without any error!

In [4]:

```
import xarray
print(xarray.__version__) #Printing version of xarray
```

0.10.8

## 2.7 Conclusions and next steps

Now that we have installed Python and learnt how to install packages, we are all set to dive deeper into data structures in python. We will discuss variables, list, dictionaries, tuples, sets, and strings in next chapter with the help of examples. If you do not understand certain jargons such as Pandas, numpy, and xarray, do not worry at this point as we will cover these extensively as we move forward!