**JUPYTER NOTEBOOK**

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. Jupyter Notebook integrates your code and its output into a single document which contains text, mathematical equations and visualisations that the code produces directly in the same page.

This step by step workflow promotes fast, iterative development since each output of your code displays right away. That’s why notebooks have become increasingly popular over the last several years, especially in Data Science.

This article is aimed for beginners keeping in mind to get started with Jupyter Notebooks. We’ll go through it all end to end : Installation, Basic Usage and how to create a data science project.

The Jupyter project is the successor to the earlier iPython Notebook, which was first published as a prototype in 2010. Although it is possible to use many different programming languages on Jupyter Notebook, this article will focus on Python as it is most common use case. (Among R users, **R Studios** tend to be a more popular choice).

The Jupyter Notebook is a combination of three languages – **Julia**, **Python** and **R.** To get most out of it, you must have a basic knowledge of any of the above languages; but Python knowledge is preferable. Jupyter notebook can also act as a flexible platform for getting to grips with pandas and even Python to understand it in clearer manner.

**Setting up a Jupyter Notebook**

The easiest way for a beginner to get started with Jupyter Notebooks is by installing **Anaconda**. Anaconda is most widely used Python distribution for data science and comes pre-loaded with all the most popular libraries and tools. Some of the biggest Python libraries wrapped up in Anaconda include **NumPy**, **pandas** and **Matplotlib**, though the full 1000+ list is exhaustive. This lets you hit the ground on running in your own fully stocked data science workshop without the hassle of managing countless installations or worrying about dependencies and OS-specific (read : Windows specific) installation issues.

To get Anaconda, simply:

* Download the latest version of Anaconda for Python 3 (ignore Python 2.7)
* Install Anaconda by following the instructions on the download page and/or in the executable.

If you are a more advanced user with Python already installed and prefer to manage your packages manually, you can just use pip:

|  |
| --- |
| pip3 install jupyter |

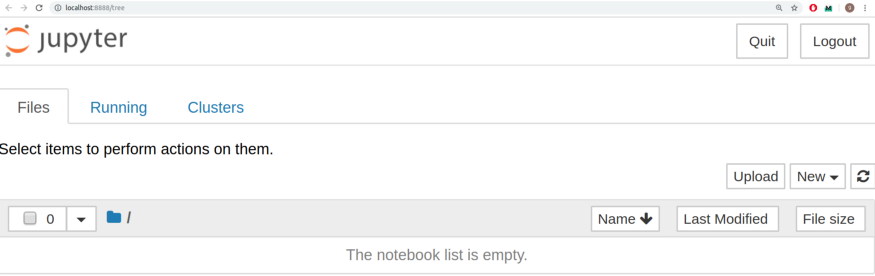
**Creating Your First Notebook**

In this section, we will learn to run and save notebooks, familiarize with their structures, and understand the interface. We’ll become acquainted with some core terminology that will steer you towards a practical understanding on how to use a jupyter notebook by yourself which steps through an example of data analysis and brings everything we learn here to life.

To get started, use your terminal to move into the folder you would like to work from using the **cd** command (Linux or Mac). Then, start up Jupyter something likewise the following command :

|  |
| --- |
| jupyter notebook |

This will start up a Jupyter server and your browser will open up a new tab to the following URL : <http://localhost:8888/tree>. It’ll look something like this :



On Windows, you can run Jupyter via the shortcut Anaconda adds to your start menu, which will open a new tab in your default web browser that should look something like the following :



This isn’t the Notebook yet; but don’t panic, it’s the Notebook dashboard, specifically designed for managing your Jupyter Notebooks. Be aware that the dashboards will give you access only to the files and sub-folders contained within Jupyter start-up directory; however, the directory can be changed.

**Running Jupyter Notebook with Anaconda**

**Getting started with JupyterLab**

Installation : JupyterLab can be installed using conda, pip, pipenv or docker

conda

If you use conda, you can install it with:

|  |
| --- |
| conda install -c conda-forge jupyterlab |

pip

If you use pip, you can install it with:

|  |
| --- |
| pip install jupyterlab |

If installing using pip install --user, you must add the user-level bin directory to your PATH environment variable in order to launch jupyterlab.

pipenv

If you use pipenv, you can install it as :

|  |
| --- |
| pipenv install jupyterlab  pipenv shell |

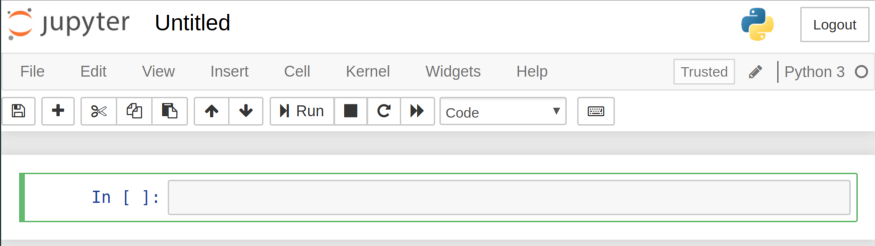
When using pipenv, in order to launch jupyterlab, you must activate the project’s virtualenv. For example, in the directory where pipenv’s pipfile and pipfile.lock live (i.e. where you ran the above commands):

Docker

If you have Docker installed, you can install and use JupyterLab by selecting one of the many ready-to-run Docker images maintained by the Jupyter Team. Follow the instructions in the Quick Start Guide to deploy the chosen Docker image.

**The Basics of Jupyter Notebooks**

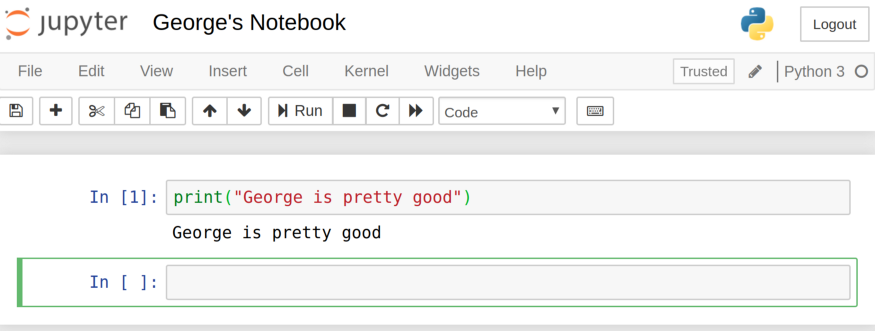
To create a notebook, click on the “new” menu in the top right and select “Python 3”. At this point your web-page will look similar to this :



You’ll notice that at the top of the page is the word *Untitled* next to the Jupyter icon – this is the title of your Notebook. Let’s change it to something a little more descriptive. Just move your mouse over the word *Untitled* and click on the text. You should now see an in-browser dialog where you can rename your Notebook; let’s say George’ Notebook.

Let’s start writing some code!

Notice how the first line of your Notebook is marked with an **In []** next to it. This keyword specifies that what you’re going to type is an input. Let’s try writing a simply print statement there. Recall that your print statement must have Python 3 syntax since this is a Python 3 Notebook. Once you write your print statement in the cell, press the ***Run***button.



**Running the Jupyter Notebook Without Anaconda**

* Contents

1. Basic Steps
2. Starting the Notebook Server
3. Introducing the Notebook Server Command Line Options

* How to open a specific Notebook?
* How to start Notebook using custom IP or port?
* How to start Notebook without opening a browser?
* How to get help about Notebook server options?

**Basic Steps :**

* Start the notebook server from the command line :

|  |
| --- |
| jupyter notebook |

* You should see the notebook open in your browser.

**Starting the Notebook Server :**

After you have installed the Jupyter Notebook on your computer, you are ready to run the notebook server. You can now start the notebook server from the command line (using Terminal on Mac/Linux, Command Prompt on Windows) by running :

|  |
| --- |
| jupyter notebook |

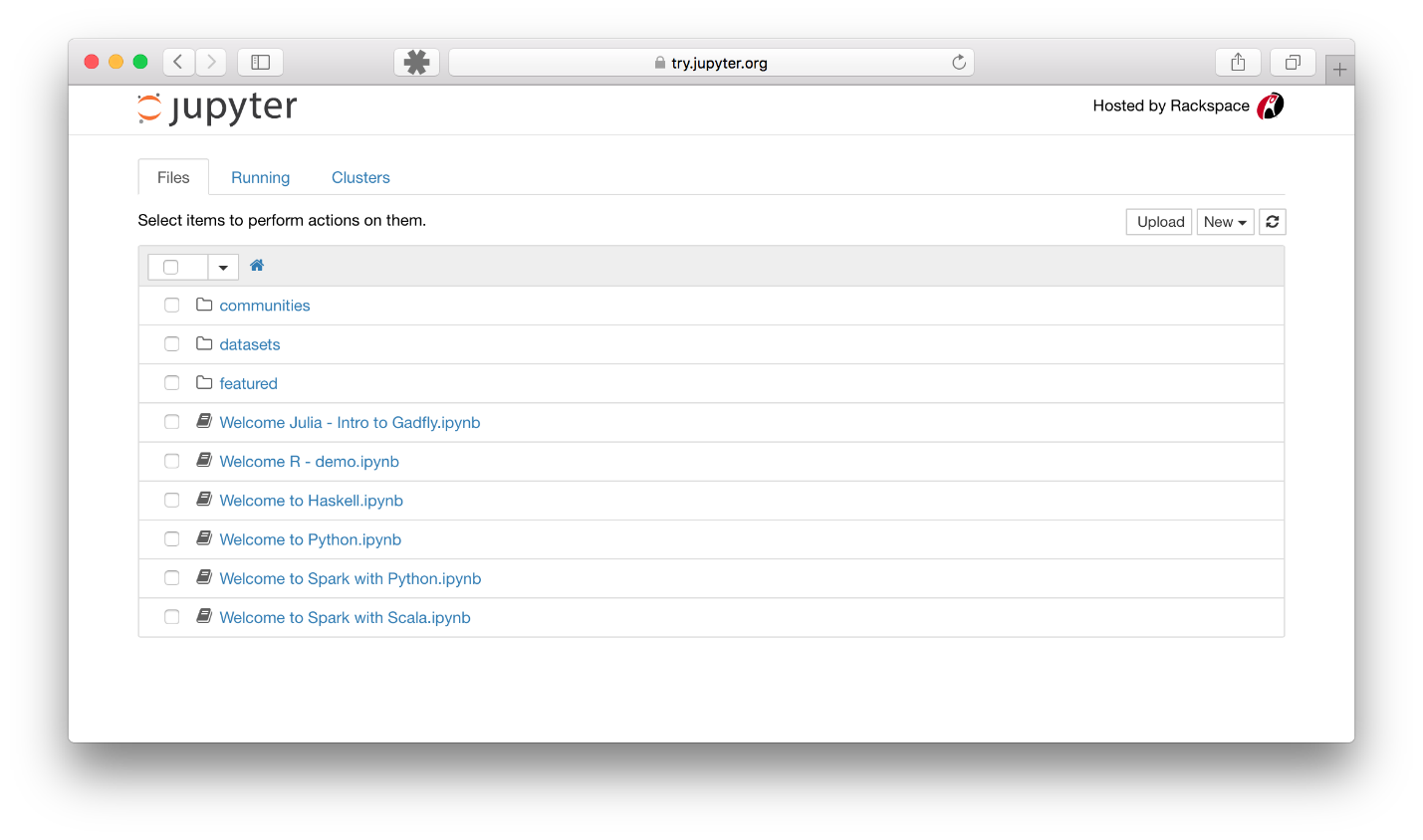
This will print some information about the notebook server in your terminal, including the URL of the web application (by default, <http://localhost:8888>):

|  |
| --- |
| $ jupyter notebook  [I 08:58:24.417 NotebookApp] Serving notebooks from local directory: /Users/catherine  [I 08:58:24.417 NotebookApp] 0 active kernels  [I 08:58:24.417 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/  [I 08:58:24.417 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation). |

It will then open your default web browser to this URL.

When the notebook opens in your browser, you’ll see the **Notebook Dashboard**, which will show a list of the notebooks, files and subdirectories in the directory where the notebook server was started. Most of the time, you will wish to start a notebook server in the highest level directory containing notebooks. Often this will be your home directory.

**Notebook Dashboard**



**Introducing the Notebook Server’s Command Line Options**

How to open a Specific Notebook?

The following code should open the given notebook in the currently running notebook server, starting one if necessary.

|  |
| --- |
| jupyter notebook notebook.ipynb |

How to start the Notebook using custom IP or port?

By default, the notebook server starts on port 8888. If port 8888 is unavailable or in use, the notebook server searches the next available port. You may also specify a port manually. In this example, we set the server’s port to 9999:

|  |
| --- |
| jupyter notebook --port 9999 |

How to start a Notebook server without opening a browser?

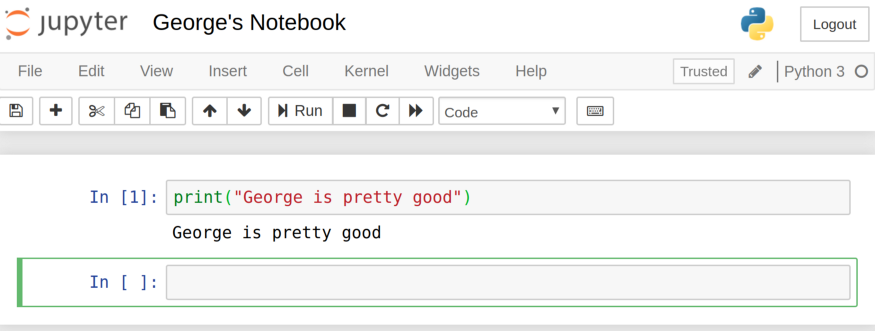
Start notebook server without opening a web browser:

|  |
| --- |
| jupyter notebook --no-browser |

How to get help about Notebook server options?

The notebook server provides help messages for other command line arguments using the --help flag:

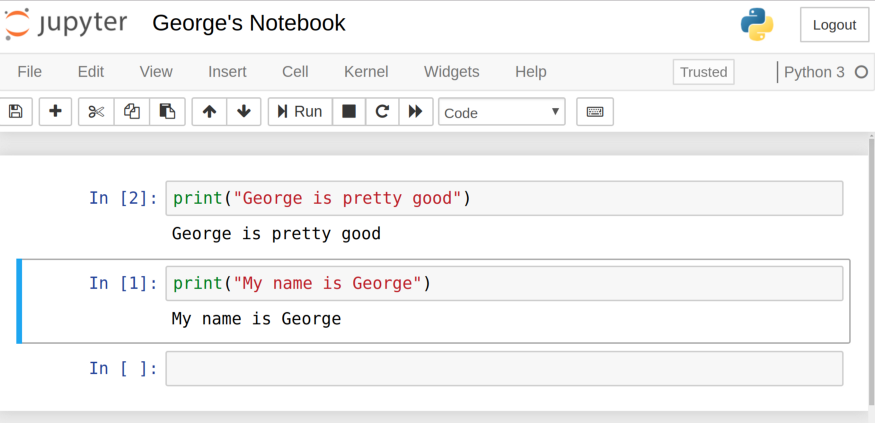
|  |
| --- |
| jupyter notebook --help |



Awesome! See how the output is printed directly on the notebook. This is how we can do an interactive project by seeing the output at each step of the process.

Also, notice when you ran the cell, the first line which had an **In []** next to it has now changed to **In [1]**. The number inside the square brackets indicates the order in which the cell is to be run; the first cell has a **[1]** because it was the first cell that we ran. We can run each cell individually at anytime and those numbers will change.

Let’s take an example of setting up 2 cells, each one with a different print statement. We’ll run the second print statement first following by the first print statement. Take a look at how the number inside the squared brackets changed as a result.



When you have multiple cells in your Notebook and you run the cells in order, you can share your variables and imports across cells. This makes it easy to separate out your code into related sections without any need to recreate variables at every cell. Just be sure that you run your cells in the proper order so that all your variables are created before usage.

**Glossary**

**command line :**

The terminal or console window where you type commands.

**Command Prompt :**

On Windows, this is the application where commands are typed into a window for execution.

**conda :**

The package manager for Anaconda.

**config :**

Refers to the configuration files and process.

**Notebook Dashboard :**

The notebook user interface which shows a list of the notebooks, files and subdirectories in the directory where the notebook server is started.

**pip :**

Python package manager.

**profiles :**

Not available in Jupyter. In IPython 3, profiles are collections of configuration and runtime files.

**REPL :**

read-eval-print-loop

**terminal :**

A window used to type in commands to be executed (Linux and OS).

**Widget** :

A user interface component, similar to a plugin, that allows customized input, such as a slider.