

**What is Jupyter Notebook?**

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects.

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

Using Notebooks is now a major part of the data science workflow at companies across the globe. If your goal is to work with data, using a Notebook will speed up your workflow and make it easier to communicate and share your results.

**Creating Your First Notebook**

In this section, we’re going to learn to run and save notebooks, familiarize ourselves with their structure, and understand the interface. We’ll become intimate with some core terminology that will steer you towards a practical understanding of how to use Jupyter Notebooks by yourself and set us up for the next section, which walks through an example data analysis and brings everything we learn here to life.

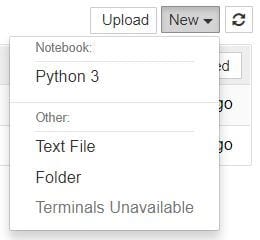
**Running Jupyter**

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 screenshot.



This is the Notebook Dashboard, specifically designed for managing your Jupyter Notebooks. Think of it as the launchpad for exploring, editing and creating your notebooks.

Be aware that the dashboard will give you access only to the files and sub-folders contained within Jupyter’s start-up directory (i.e., where Jupyter or Anaconda is installed). However, the start-up directory [can be changed](https://stackoverflow.com/q/35254852/604687).



With Jupyter Notebook open in your browser, you may have noticed that the URL for the dashboard is something like https://localhost:8888/tree. Localhost is not a website, but indicates that the content is being served from your *local* machine: your own computer.

Browse to the folder in which you would like to create your first notebook, click the “New” drop-down button in the top-right and select “Python 3”:

Each notebook uses its own tab because you can open multiple notebooks simultaneously.

If you switch back to the dashboard, you will see the new file Untitled.ipynb and you should see some green text that tells you your notebook is running.

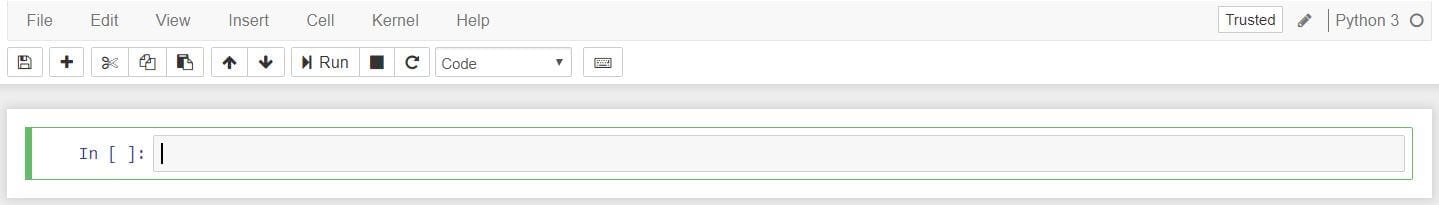
**What is an ipynb File?**

The short answer: each .ipynb file is one notebook, so each time you create a new notebook, a new  .ipynb file will be created.

The longer answer: Each .ipynb file is a text file that describes the contents of your notebook in a format called [JSON](https://en.wikipedia.org/wiki/JSON). Each cell and its contents, including image attachments that have been converted into strings of text, is listed therein along with some [metadata](https://ipython.org/ipython-doc/3/notebook/nbformat.html#metadata).

**The Notebook Interface**

Now that you have an open notebook in front of you, its interface will hopefully not look entirely alien. After all, Jupyter is essentially just an advanced word processor.



There are two fairly prominent terms that you should notice, which are probably new to you: *cells* and *kernels* are key both to understanding Jupyter and to what makes it more than just a word processor. Fortunately, these concepts are not difficult to understand.

* A **kernel** is a “computational engine” that executes the code contained in a notebook document.
* A **cell** is a container for text to be displayed in the notebook or code to be executed by the notebook’s kernel.

**Cells**

Cells form the body of a notebook. In the screenshot of a new notebook in the section above, that box with the green outline is an empty cell. There are two main cell types that we will cover:

* A **code cell** contains code to be executed in the kernel. When the code is run, the notebook displays the output below the code cell that generated it.
* A **Markdown cell** contains text formatted using [Markdown](https://www.markdownguide.org/basic-syntax/) and displays its output in-place when the Markdown cell is run.

The first cell in a new notebook is always a code cell.

Let’s test it out with a classic hello world example: Type print('Hello World!') into the cell and click the run button Notebook Run Button in the toolbar above or press Ctrl + Enter.

The result should look like this:

print('Hello World!')

Hello World!

When we run the cell, its output is displayed below and the label to its left will have changed from In [ ] to In [1].

The output of a code cell also forms part of the document. You can always tell the difference between code and Markdown cells because code cells have that label on the left and Markdown cells do not.

The “In” part of the label is simply short for “Input,” while the label number indicates *when* the cell was executed on the kernel — in this case the cell was executed first.

Run the cell again and the label will change to In [2] because now the cell was the second to be run on the kernel.

**Kernels**

Behind every notebook runs a kernel. When you run a code cell, that code is executed within the kernel. Any output is returned back to the cell to be displayed. The kernel’s state persists over time and between cells — it pertains to the document as a whole and not individual cells.

For example, if you import libraries or declare variables in one cell, they will be available in another. Let’s try this out to get a feel for it.

And if we ever wish to reset things, there are several incredibly useful options from the Kernel menu:

* Restart: restarts the kernel, thus clearing all the variables etc that were defined.
* Restart & Clear Output: same as above but will also wipe the output displayed below your code cells.
* Restart & Run All: same as above but will also run all your cells in order from first to last.

If your kernel is ever stuck on a computation and you wish to stop it, you can choose the Interrupt option.