

Jupyter Notebook Introduction



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. The intuitive workflow promotes iterative and rapid development, making notebooks an increasingly popular choice at the heart of contemporary data science, analysis, and increasingly science at large. Jupyter Notebook is maintained by the people at [Project Jupyter \(http://jupyter.org/\)](http://jupyter.org/).

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Getting Up and Running With Jupyter Notebook

The Jupyter Notebook is not included with Python, so if you want to try it out, you will need to install Jupyter. It is assumed that you are using [Python 3 \(https://www.python.org/\)](https://www.python.org/).

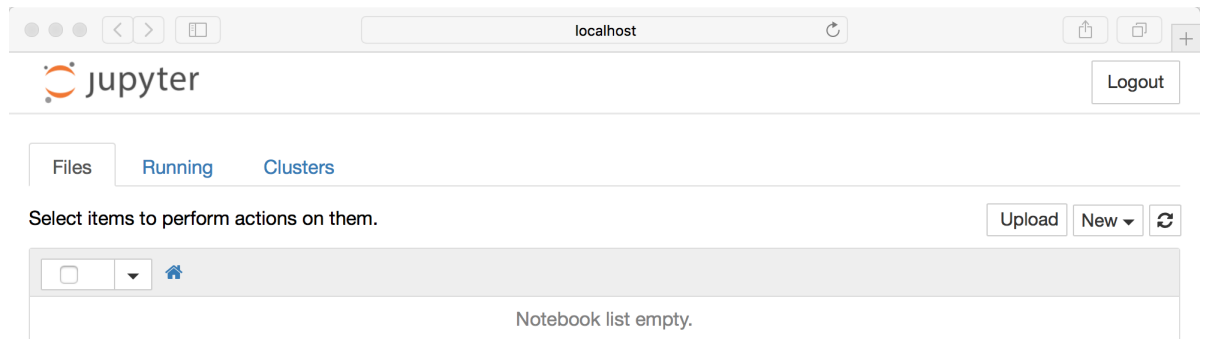
Installation

The most popular distribution of Python is [Anaconda \(https://www.anaconda.com/distribution/\)](https://www.anaconda.com/distribution/). Anaconda comes with many scientific libraries preinstalled, including the Jupyter Notebook, so you don't actually need to do anything other than install Anaconda itself.

Starting the Jupyter Notebook Server

Now that you have Jupyter installed, let's learn how to use it. On Windows, you can run Jupyter via the shortcut Anaconda apps from your start menu, which will open a new tab in your default web browser to the following URL: <http://localhost:8888/tree> (<http://localhost:8888/tree>)

Your browser should now look something like this:



(https://github.com/liuhoward/teaching/raw/master/big_data/assets/01_initial_notebook_screen.w)

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. Jupyter's Notebooks and dashboard are web apps, and Jupyter starts up a local Python server to serve these apps to your web browser, making it essentially platform independent and opening the door to easier sharing on the web. It is also possible to start the dashboard on any system via the command prompt (or terminal on Unix systems) by entering the command

```
$ jupyter notebook
```

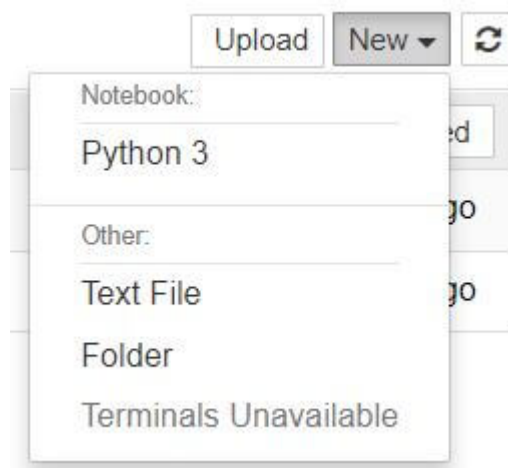
in this case, the current working directory will be the start-up directory.



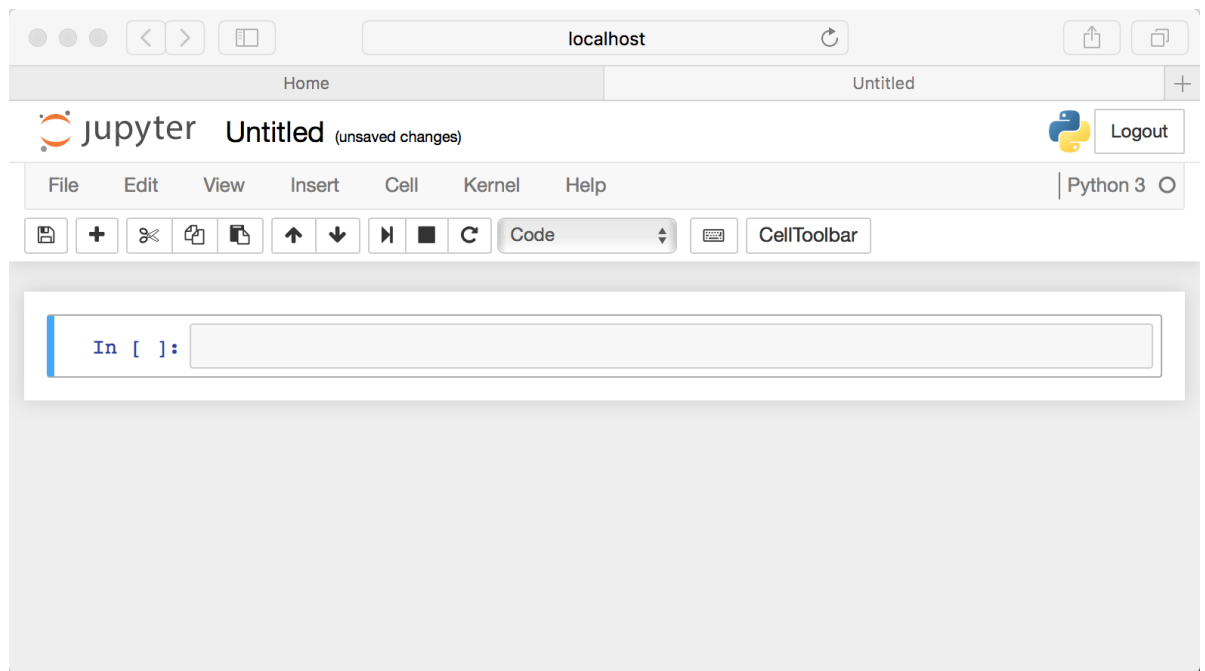
Creating a Notebook

Now that you know how to start a Notebook server, you should probably learn how to create an actually Notebook document.

All you need to do is click on the *New* button (upper right), and it will open up a list of choices. For simplicity's sake, let's choose Python 3.



Your web page should now look like this:



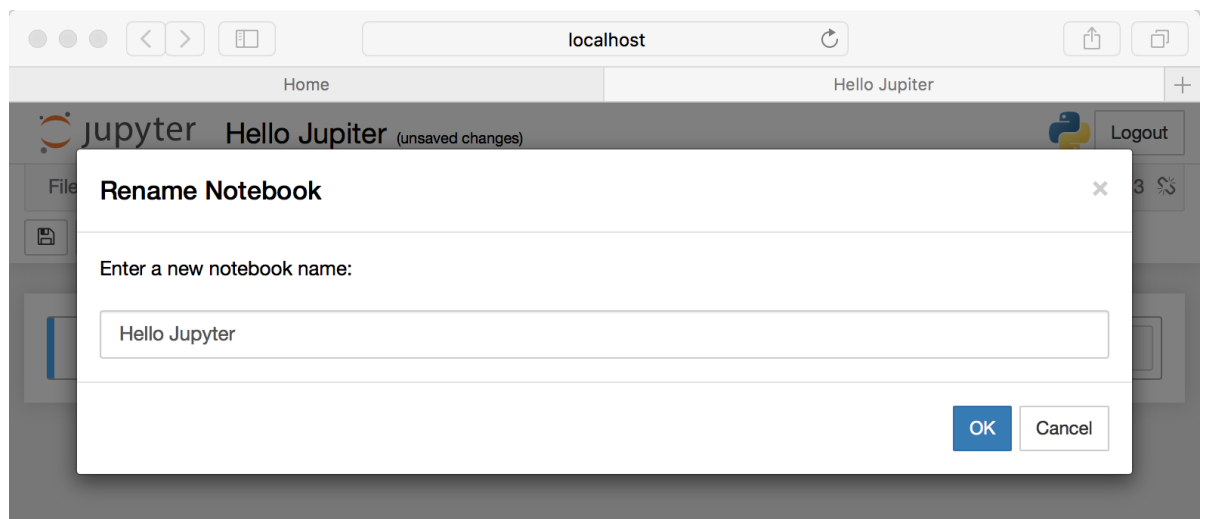
(https://github.com/liuhoward/teaching/raw/master/big_data/assets/02_new_notebook.webp)

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. 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) (<https://en.wikipedia.org/wiki/JSON>). You can edit this yourself — if you know what you are doing! — by selecting “Edit > Edit Notebook Metadata” from the menu bar in the notebook.

Naming

You will notice that at the top of the page is the word *Untitled*. This is the title for the page and the name of your Notebook. Since that isn't a very descriptive name, let's change it!

Just move your mouse over the word *Untitled* and click on the text. You should now see an in-browser dialog titled *Rename Notebook*. Let's rename this one to *Hello Jupyter*.



(https://github.com/liuhoward/teaching/raw/master/big_data/assets/03_hello_jupyter.webp)

If 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 logical chunks without needing to reimport libraries or recreate variables or functions in every cell.

When you run a cell, you will notice that there are some square braces next to the word *In* to the left of the cell. The square braces will auto fill with a number that indicates the order that you ran the cells. For example, if you open a fresh Notebook and run the first cell at the top of the Notebook, the square braces will fill with the number *1*.

The Menus

The Jupyter Notebook has several menus that you can use to interact with your Notebook. The menu runs along the top of the Notebook just like menus do in other applications. Here is a list of the current menus:

- *File*
- *Edit*
- *View*
- *Insert*
- *Cell*
- *Kernel*
- *Widgets*
- *Help*

Let's go over the menus one by one. This article won't go into detail for every single option in every menu, but it will focus on the items that are unique to the Notebook application.

The first menu is the *File* menu. In it, you can create a new Notebook or open a preexisting one. This is also where you would go to rename a Notebook. I think the most interesting menu item is the *Save and Checkpoint* option. This allows you to create checkpoints that you can roll back to if you need to.

Next is the *Edit* menu. Here you can cut, copy, and paste cells. This is also where you would go if you wanted to delete, split, or merge a cell. You can reorder cells here too.

Note that some of the items in this menu are greyed out. The reason for this is that they do not apply to the currently selected cell. For example, a code cell cannot have an image inserted into it, but a Markdown cell can. If you see a greyed out menu item, try changing the cell's type and see if the item becomes available to use.

The *View* menu is useful for toggling the visibility of the header and toolbar. You can also toggle *Line Numbers* within cells on or off. This is also where you would go if you want to mess about with the cell's toolbar.

The *Insert* menu is just for inserting cells above or below the currently selected cell.

The *Cell* menu allows you to run one cell, a group of cells, or all the cells. You can also go here to change a cell's type, although I personally find the toolbar to be more intuitive for that.

The other handy feature in this menu is the ability to clear a cell's output. If you are planning to share your Notebook with others, you will probably want to clear the output first so that the next person can run the cells themselves.

The *Kernel* cell is for working with the kernel that is running in the background. Here you can restart the kernel, reconnect to it, shut it down, or even change which kernel your Notebook is using.

You probably won't be working with the Kernel all that often, but there are times when you are debugging a Notebook that you will find you need to restart the Kernel. When that happens, this is where you would go.

The *Widgets* menu is for saving and clearing widget state. Widgets are basically JavaScript widgets that you can add to your cells to make dynamic content using Python (or another Kernel).

Finally you have the *Help* menu, which is where you go to learn about the Notebook's keyboard shortcuts, a user interface tour, and lots of reference material.

Starting Terminals and Other Things

Jupyter Notebook also allows you to start more than just Notebooks. You can also create a text file, a folder, or a Terminal in your browser. Go back to the home page that opened when you first started the Jupyter server at `http://localhost:8888/tree`. Go to the *New* button and choose one of the other options.

The Terminal is probably the most interesting of the bunch, as it is running your operating systems terminal in the browser. This allows you to run bash, Powershell, and so on in your browser and run any shell command that you might need to there.

Viewing What's Running

Also on the home page of your Jupyter server (`http://localhost:8888/tree`) are two other tabs: *Running* and *Clusters*.

The *Running* tab will tell you which Notebooks and Terminals you are currently running. This is useful for when you want to shut down your server but you need to make sure that you have saved all your data. Fortunately, Notebooks auto-save pretty frequently, so you rarely lose data. But it's good to be able to see what's running when you need to.

The other nice thing about this tab is that you can go through your running applications and shut them down there.

Cell Types

There are two main cell types that we will cover:

- A **code cell** contains code to be executed in the kernel and displays its output below.

- A **Markdown cell** contains text formatted using Markdown and displays its output in-place when it is run.

A Notebook's cell defaults to using code whenever you first create one, and that cell uses the kernel that you chose when you started your Notebook. In this case, you started yours with Python 3 as your kernel, so that means you can write Python code in your code cells.

Let's try adding the following code to that cell:

```
print('Hello Jupyter!')
```

Running a cell means that you will execute the cell's contents. To execute a cell, you can just select the cell and click the *Run* button that is in the row of buttons along the top. It's towards the middle. If you prefer using your keyboard, you can just press `Shift + Enter` or `Ctrl + Enter`.

When I ran the cell, its output will have been displayed below and the label to its left will have changed from `In []` to `In [1]`. 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.

```
In [ ]: print('Hello Jupyter')
```

In general, the output of a cell comes from any text data specifically printed during the cells execution, as well as the value of the last line in the cell, be it a lone variable, a function call, or something else. From the menu bar, click *Insert* and select *Insert Cell Below* to create a new code cell underneath your first and try out the following code:

```
def say_hello(recipient):  
    return 'Hello, {}'.format(recipient)  
say_hello('Tim')
```

```
In [ ]: def say_hello(recipient):  
        return 'Hello, {}'.format(recipient)  
say_hello('Tim')
```

Keyboard shortcuts

One final thing you may have observed when running your cells is that their border turned blue, whereas it was green while you were editing. There is always one “active” cell highlighted with a border whose colour denotes its current mode, where green means “edit mode” and blue is “command mode.”

So far we have seen how to run a cell with `Ctrl + Enter`, but there are plenty more. Keyboard shortcuts are a very popular aspect of the Jupyter environment because they facilitate a speedy cell-based workflow. Many of these are actions you can carry out on the active cell when it's in command mode.

Below, you'll find a list of some of Jupyter's keyboard shortcuts. You're not expected to pick them up immediately, but the list should give you a good idea of what's possible.

- Toggle between edit and command mode with `Esc` and `Enter`, respectively.
- Once in command mode:
 - Scroll up and down your cells with your `Up` and `Down` keys.
 - Press `A` or `B` to insert a new cell above or below the active cell.
 - `M` will transform the active cell to a Markdown cell.
 - `Y` will set the active cell to a code cell.
 - `D + D` (`D` twice) will delete the active cell.
 - `Z` will undo cell deletion.
 - Hold `Shift` and press `Up` or `Down` to select multiple cells at once.
 - With multiple cells selected, `Shift + M` will merge your selection.
- `Ctrl + Shift + -`, in edit mode, will split the active cell at the cursor.
- You can also click and `Shift + Click` in the margin to the left of your cells to select them.

Go ahead and try these out in your own notebook. Once you've had a play, create a new Markdown cell and we'll learn how to format the text in our notebooks.

Markdown

[Markdown \(https://www.markdownguide.org/\)](https://www.markdownguide.org/) is a lightweight, easy to learn markup language for formatting plain text. Its syntax has a one-to-one correspondance with HTML tags, so some prior knowledge here would be helpful but is definitely not a prerequisite. Remember that this article was written in a Jupyter notebook, so all of the narrative text and images you have seen so far was achieved in Markdown. Let's cover the basics with a quick example.

This is a level 1 heading

This is a level 2 heading

This is some plain text that forms a paragraph.

Add emphasis via *****bold***** and **__bold__**, or **italic** and *_italic_*.

Paragraphs must be separated by an empty line.

* Sometimes we want to include lists.

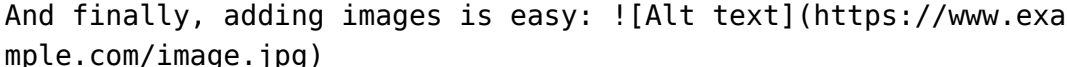
* Which can be indented.

 * sub item

1. Lists can also be numbered.

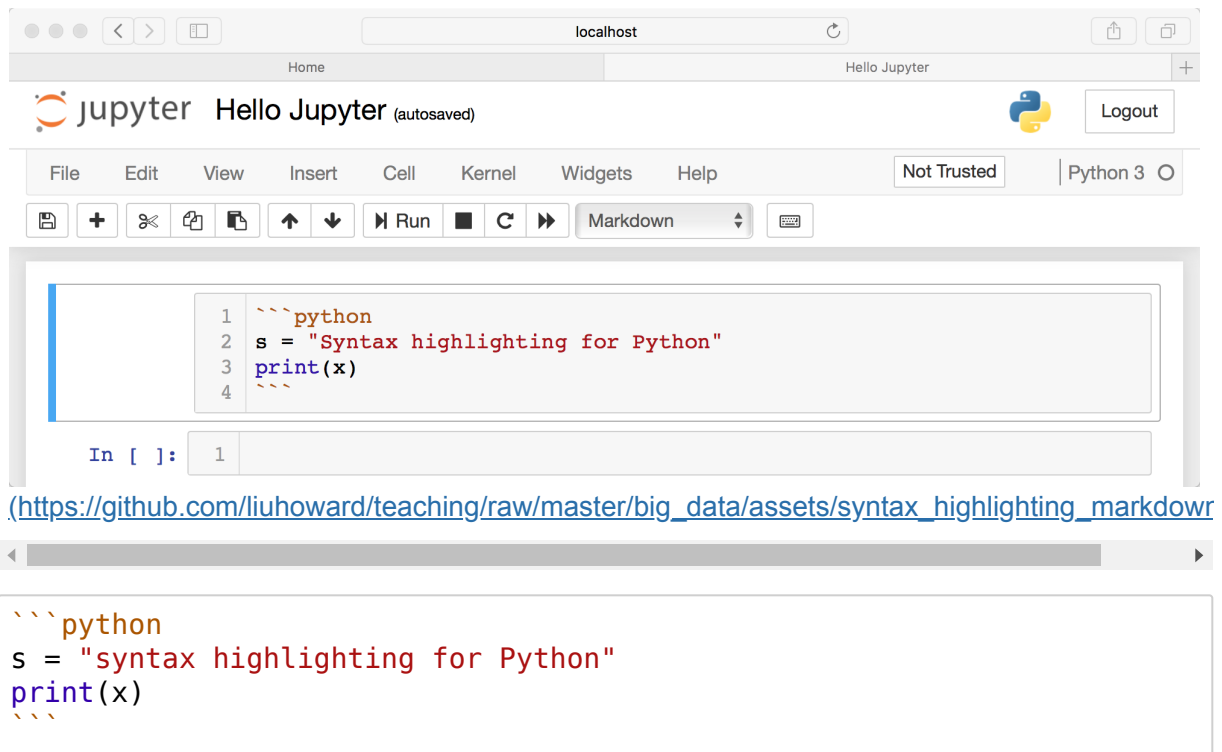
2. For ordered lists.

[It is possible to include hyperlinks](https://www.example.com)

And finally, adding images is easy: 

Code and Syntax Highlighting

If you want to insert a code example that you don't want your end user to actually run, you can use Markdown to insert it. For inline code highlighting, just surround the code with backticks. If you want to insert a block of code, you can use triple backticks and also specify the programming language:



Exporting Notebooks

When you are working with Jupyter Notebooks, you will find that you need to share your results with non-technical people. You can export your currently running Notebook by going to the *File* menu and choosing the *Download as* option to convert or export your Notebook into one of the following formats:

- HTML
- LaTeX
- PDF
- RevealJS
- Markdown
- ReStructured Text
- Executable script

Also note that *Download as* option also depends on [Pandoc \(https://pandoc.org/\)](https://pandoc.org/) and TeX to be able to export to all the formats above. If you don't have one or more of these, some of the export types may not work. You can also use Browser built-in *Print* to export your notebook as PDF.

Reference:

1. <https://realpython.com/jupyter-notebook-introduction/> (<https://realpython.com/jupyter-notebook-introduction/>)
2. <https://www.dataquest.io/blog/jupyter-notebook-tutorial/> (<https://www.dataquest.io/blog/jupyter-notebook-tutorial/>)
3. <https://github.com/takluyver/jupyter-intro-xfel-jan17> (<https://github.com/takluyver/jupyter-intro-xfel-jan17>)

