week1_jupyter_intro

November 26, 2018

1 Quick intro to Jupyter Notebook

originally by Stephan Rasp, modified by Baird

Jupyter notebooks are a fantastic tool for data analysis and code development. This notebook explains the essentials to get you up and running.

---

2 Installation

We will assume that you are using Anaconda. To install Jupyter type:

```
conda install jupyter
```

2.0.1 Notebook extensions

You can add extended functionalities by installing notebook extensions:

```
conda install -c conda-forge jupyter_contrib_nbextensions
```

This will add a tab to your dashboard, from where you can select the extension. These include: * Collapsible heading extensions * Table of contents extensions * Cell runtime information, * … and much more

2.0.2 Notebook themes

For changing the look of the notebook, you want to run `pip install jupyterthemes`. This will allow you to select different themes. For more information visit: https://github.com/dunovank/jupyter-themes

---

3 Basic usage

To start a Jupyter notebook, type:

```
jupyter notebook
```

in the command line. If you do this locally (e.g., on your laptop) this should automatically open up a browser window with the dashboard. We'll cover running a notebook remotely below.

On the dashboard you can see all files in your current directory. (Note that you can click on any file and edit it.) Jupyter notebook files end in `.ipynb`. To create a new notebook, click on `New` and select a Python kernel. Now a new tab will appear with your empty notebook.

3.0.1 Cells

Jupyter notebooks have cells such as this one, and there are different types. The two most useful ones are code cells and markdown cells.
This here is a markdown cell, which allows us to write formatted text using the "markdown" language. For a quick introduction, see here: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Note that in markdown you can also type Latex:
$x = \frac{a}{b}$

To switch the cell type, either use the toolbar above or press `y` for code or `m` for markdown in command mode, which we will get to now.

3.0.2 Modes

To navigate Jupyter notebooks there are two modes: 1. Command mode, which you can enter by pressing `esc` (cells have a BLUE border) * Lets you arrow up/down across cells, delete them, etc.

2. Edit mode which you can enter by pressing `enter/return` (cells have a GREEN border)

- Lets you edit the text within a cell

3.0.3 Keyboard shortcuts!

If you get confused about all the keys just press `h` in command mode (may need to press `esc` to do this).
Of course, you can also use the mouse and the toolbar.

3.0.4 Executing a cell

To execute a cell, press `shift + enter`. This will run the code or render the text and jump to the next cell.
If you would like to insert a cell below, type `alt + enter`.

Now let's start by actually writing some code.

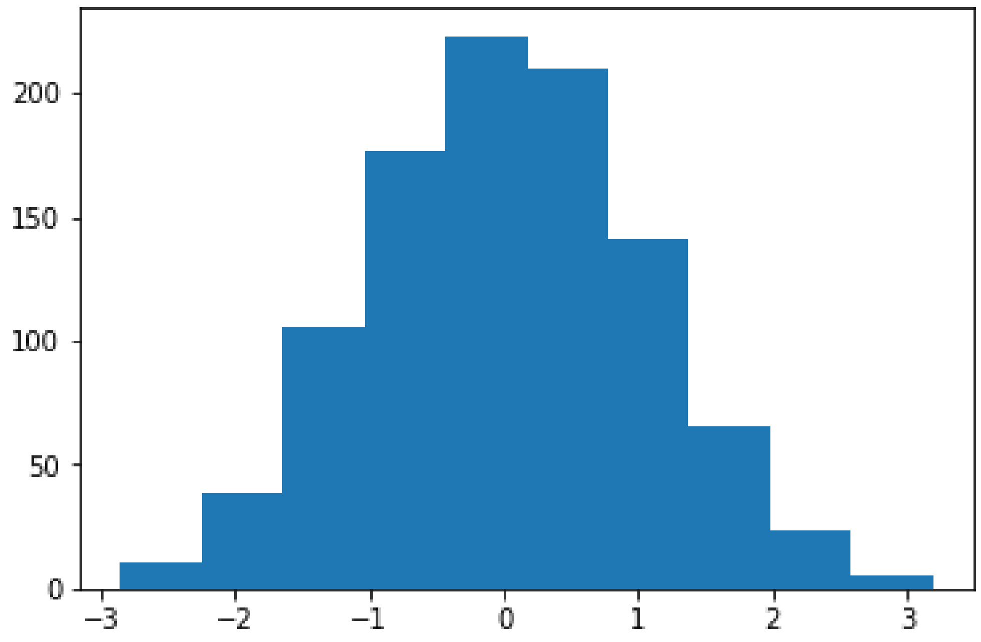
```
In [1]: # Import modules just as you would in a regular Python script
        import numpy as np
        import matplotlib.pyplot as plt

In [2]: # Assign variables
        a = 10

In [3]: # The value of the last object in the cell will be displayed
        b = 2
        a

Out[3]: 10

In [4]: a + b

Out[4]: 12
```

You do not have to work chronologically. You can always edit cells above.
But pay attention:
This means that the notebook might throw an error when executed in chronological order after restarting the kernel.

Get function docs
Next, create a basic plot and also note a super-handy functionality of the notebook:
* When using a function and pressing `shift + tab` when the cursor is inside the function parenthesis, it will bring up the list of arguments.
Pressing it twice will bring up the doc string.

Similarly, executing `?function_name` will bring up the doc string in a separate window. `??function_name` will bring up the source code.

```
In [5]: plt.hist(np.random.normal(size=1000))
        plt.show()
```



```
In [6]: ?plt.hist
```

```
In [7]: ??plt.hist
```

4 Sharing a notebook

4.0.1 GitHub

If the notebook is in a GitHub repository (which it probably should be!), GitHub will simply render the notebook for you like this: https://github.com/raspstephan/ESS-Python-Tutorial/blob/master/notebooks/jupyter-intro.ipynb

4.0.2 NBViewer

Another option is to copy the GitHub URL to http://nbviewer.jupyter.org/.

4.0.3 As HTML or PDF

You can also export the notebook to HTML or PDF by typing `jupyter nbconvert --to FORMAT notebook.ipynb` where `FORMAT` can be `pdf`, `html` or many others.

More information here: https://ipython.org/ipython-doc/3/notebook/nbconvert.html

---

5 Remote setup

How to run jupyter notebook remotely

5.0.1 Port tunnel

If you want to run Jupyter on a remote computer (e.g. greenplanet), you need to create a port tunnel in order to open the notebook locally in the browser. The command to do this is:

```
ssh -L localhost:8888:localhost:8888 user@gplogin1.ps.uci.edu
```

This will create a tunnel for port 8888. If you do your actual computations on a *different* node, you can do the same thing to create a tunnel to that node:

```
ssh -L localhost:8888:localhost:8888 c-3-39
```

Now locally in your browser, go to `localhost:8888`. Most probably it will ask you for a token. You will find the token in the terminal where Jupyter is running, where it will look something like this:

```
http://localhost:8888/?token=940f1a436d91bde5ef15e3640dd66bd7d3f724a68513c99f
```

Just copy and paste the token after = and you should be good to go.

5.0.2 Screen

Since remote connections die sometimes, screen is very handy. It lets you run processes remotely, which will continue even after your connection has timed out.

To start a screen session, type `screen`. To execute screen commands, you need to press `ctrl + a` followed by a command. `c` will create a new window. `n` and `p` will go to the next and previous window. `k` will kill the window. Finally, `d` will detach from the window.

To log back into a screen session after detaching or having been disconnected, type `screen -r`.

Here is more info: http://aperiodic.net/screen/quick_reference

5.0.3 Mounting

Finally, to edit remote files in your preferred editor (PyCharm, VSC, etc.), you might want to mount a remote directory. For Linux/Mac you can use sshfs. On the Mac you have to install OSXFuse.

Here is a link outlining the process: https://blogs.harvard.edu/acts/2013/11/08/the-newbie-how-to-set-up-sshfs-on-mac-os-x/

Then you can mount a remote directory: `sshfs USERNAME@gplogin1.ps.uci.edu:/export/home/USERNAME/ greenplanet_home/`