23-jupyter

November 29, 2020

1 Jupyter

Jupyter Notebook is an app that runs on the browser for editing and running *notebook documents*. According to jupyter's website:

> Notebook documents (or "notebooks", all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc.). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis,

Even though it runs on the browser, it is not on the internet!

Jupyter is part of the Anaconda package, so you probably have it already on your computer. Go to the Anacondau launcher, find jupyter, and click to open it [1]. This should open a new tab on your browser. (If this does not work, ask the course staff.)

The tab that just opened has the notebook's dashboard, and contains a list of files in a folder in your computer [2]. This folder is, most likley, your user folder. You can navigate your files from this folder down, by following other folders in the dashboard. On the upper right corner there is a "New" button with an arrow down. Clicking on it gives you some options:

- Python3: creates a new notebook
- Text file: creates a new text file (no formatting, only text, like a .txt or .csv file)
- Folder: creates a new folder
- Terminal: why don't you try clicking on it?

Note that notebooks, text files, and folders will be created on the directory that is on jupyter's dashboard. You can open that same directory on your file navigator (explorer on Windows and finder on MacOS) to see your new file or folder there.

You will need to have Jupyter installed for the final course project.

Check this notebook that explains some of the basic elements of jupyter notebook.

---

[1] If this opens a terminal (a window with a black screen), you might need to leave it open for jupyter to work.

[2] Your computer's files are organized in a tree-like structure, with one root folder (usually `C:` or `/`) and folders and files inside folders. When working with jupyter, python, and files, it is

important to know how this works and where your files are exactly!

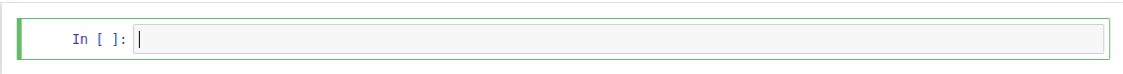
1.1 Why notebooks?

Jupyter notebook is a nice environment for data analysis because we can, in one place, play around with the code, see graphs and document our analysis. Moreover, we can share our findings easily in a nicely formated way without having to worry about importing/copying things from one software to the other. Notebooks can be exported to `pdf` and `html`, in case you want to share with people that do not have jupyter. People in the scientific community use it all the time, and so should you since you are on your path to becoming a scientist :)

You can check and get inspired by some nice examples of notebooks at here.

1.2 Your first notebook

Now that you understand the mechanics of jupyter, let's start creating your first notebook. Navigate to a folder where you want your notebook to be (or create a new folder), and click on New -> Python3. This will open a new notebook on a new browser tab. You should see one cell like this:



Cells are always prefixed by `In: [ ]` and they can contain code *or* explanatory text (but not both at the same time). You can change the type of a cell from the toolbar:

The dropdown menu indicates the type of the current cell. You can click on it to change the type. If you want explanatory text, choose "Markdown".

1.2.1 Markdown

Markdown is a markup language for writing formatted text that can be read by any generic text editor. You can think about it as a way of typing text as you would type in a word document, but with specific commands for making text italic, bold, a header or adding an image or table. This means that you can do all of those things without taking your hands off the keyboard.

Some examples of formatting you may find useful are:

```
# Section heading
## Subsection heading
### Subsubsection heading
*italic*
**bold**
[link](http://the.address)
- list item
  - sublist item
1. numbered list item
```

Writing this in a cell may look very weird at first, but once the cell is "ran", the text is rendered nicely.

There is a nice cheatsheet with the most common markdown commands. If you want some kind of formatting that is not there, you can google things like "What is the markdown syntax for . . . ?". You can also ask the course staff, but know that there is no actual "course" for markdown. It is something people usually pick up as they need.

Tip: You can look at how the markdown syntax looks like in any of the lecture notebooks (including this one) by opening it on jupyter and double-clicking on the text.

1.2.2 Cells

A cell has two *modes*, indicated by the color of the frame around it:

- Green frame indicates *edit* mode. This is when you can type stuff (python code or markdown) in the cells.
- Blue frame indicates *command* mode. The cell cannot be edited and if you try to type stuff weird things may happen. (Like Spyder, jupyter has a bunch of keyboard shortcuts, and trying to type something in command mode might activate some of these shortcuts.)

You can switch modes by clicking in/out the textbox of the cell, or by pressing *esc* and *enter*.

Once you have typed what you want in a cell, you can run it by going to the top-menu and clicking on Cell -> Run Cells, or by hitting `ctrl + enter`. Alternatively, you can click `shift + enter` and focus moves to the next cell (or creates a new one if it does not exist).

If you need to create a new cell, you can either hit `shitf + enter` on the last cell, or click the + button on the toolbar.

Note that **python code in cells in the same notebook behaves as if it was in the same python file**. That means that if you imported something on the first cell (and ran it!), this is available for every subsequent cell. That also holds true of global variables and functions. An annoying consequence of this is that sometimes the code in the cells become "out of sync", specially if they are modified and ran in a random order, which may cause your code to behave unexpectedly. To clear all memory of past runs, you can click on the top-menu Kernel -> Restart. If you want to restart and run everything from scratch (which you should do often), click on Kernel -> Restart & Run All.

[ ]: