00. Using the Jupyter notebook

December 26, 2021

1 Using the Jupyter notebook

The **Jupyter notebook** allows you to write notebooks similar to e.g. Mathematica, SageMath, etc. The advantage of doing this is that you can include text, code, and plots in the same document. This makes it ideal for example to include code, notes, and plots about projects and sharing it with others.

1.1 Starting up

The normal way to start up the IPython notebook is:

```
jupyter notebook
```

Once you do this, your web browser should open and go to a page showing a list of folders.

1.2 First steps

[ ]:

Click on **New** on the right then select **Python** (if there are several, select **Python [default]**), which will start a new document. You can change the name of the document by clicking on the **Untitled** name at the top and entering a new name. Make sure you then save the document (make sure that you save regularly as you might lose content if you close the browser window).

At first glance, a notebook looks like a fairly typical application - it has a menubar (File, Edit, View, etc.) and a tool bar with icons. Below this, you will see an empty cell, in which you can type any Python code. You can write several lines of code, and once it is ready to run, you can press **shift-enter** and it will get executed:

[ ]:
```
a = 1
print(a)
```

You can then click on that cell, change the Python code, and press **shift-enter** again to re-execute the code. Once you have executed a cell once, a new cell will appear below. You can again enter some code, then press shift-enter to execute it.

1.3 Plotting

To make plots, enter any Matplotlib commands (which we'll learn about in the plotting section of the workshop), and just press shift-enter - note that all commands for a plot should be entered in one cell, you cannot split it up over multiple cells:

[ ]:
```
%matplotlib notebook
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.xlabel("x")
plt.ylabel("y")
```

As before, you can always go back and edit the cell to re-make the plot. If you want to save it, make sure you include `plt.savefig(filename)` as the last command, where `filename` is the name of the plot, such as `my_plot.png`.

1.4 Text

To enter actual text (non-code) in the notebook, click on a cell, and in the drop-down menu in the toolbar, select 'Markdown'. This is a specific type of syntax for writing text. You can just write text normally and press shift-enter to *render* it:

```
This is some plain text
```

To edit it, double click on the cell. You can also enter section headings using the following syntax:

```
This is a title
===============

This is a sub-title
-------------------
```

which will look like:

2 This is a title

2.1 This is a sub-title

Finally, if you are familiar with LaTeX, you can enter equations using:

```
$$E = m c^2$$
```

on a separate line, or:

```
The equation $p=\frac{h}{\lambda}$ is very important
```

to include it in a sentence. This will look like:

$$E = mc^2$$

The equation $p = \frac{h}{\lambda}$ is very important

2.2 Splitting/deleting/moving cells

You can split, delete, and move cells by going to 'Edit' and selecting the appropriate command. Some of the commands are also available in the toolbar - put your mouse over the icon and wait for a second, and it will tell you what it does.

2.3 Sharing your notebook

Once you are happy with your notebook, you can share it with other people. The file containing your notebook is the `.ipynb` file that is in the directory in which you started the notebook and has the correct name. You can send this file directly to other people, and they will be able to view it in their Jupyter notebook application.

If you add a notebook to GitHub, it will also be rendered there, so this is another way of sharing notebooks. To quickly share a notebook, you can just go to http://gist.github.com and drag and drop the notebook onto the page, then click on **Create Private Gist** or **Create Public Gist** (a private one is still really public, but only people with the link will be able to find it).

2.4 Important notes

A few important notes about using the notebook: code *can* be executed in an order different from top to bottom, but note that if you do this, variables will not be reset. So for example if you type:

```
[ ]: a = 1
```

then go higher up and type:

```
[ ]: print(a)
```

it will give the value you previously set. To make sure that your code works from top to bottom, go to the **Cell** menu item and go to **All Output** -> **Clear** then in the **Cell** menu, select **Run All**.

In addition, even if you remove a cell, then variables set in that cell still exist unless you restart the notebook. If you want to restart a notebook, you can select **Kernel** -> **Restart**. This removes any variables from memory, and you have to start running the notebook from the start.

We interact with python by typing into *cells* in the notebook. By default, a cell is a *code* cell, which means that you can enter any valid python code into it and run it. Another important type of cell is a *markdown* cell. This lets you put text, with different formatting (italics, bold, etc) that describes what the notebook is doing.

You can change the cell type via the menu at the top, or using the shortcuts:

- ctrl-m m : mark down cell
- ctrl-m y : code cell

Some useful short-cuts:

- shift+enter = run cell and jump to the next (creating a new cell if there is no other new one)
- ctrl+enter = run cell-in place
- alt+enter = run cell and insert a new one below

ctrl+m h lists other commands

Important: when you work through a notebook, everything you did in previous cells is still in memory and *known* by python, so you can refer to functions and variables that were previously defined. Even if you go up to the top of a notebook and insert a cell, all the information done earlier in your notebook session is still defined – it doesn't matter where physically you are in the notebook. If you want to reset things, you can use the options under the *Kernel* menu.

Quick Exercise:

Create a new cell below this one. Make sure that it is a *code* cell, and enter the following code and run it:

```
print("Hello, World")
```

[ ]:

`print()` is a *function* in python that takes arguments (in the `()`) and outputs to the screen. You can print multiple quantities at once like:

[ ]:
```
print(1, 2, 3)
```

[ ]: