

1.3 Jupyter – Running


March 23, 2020

1 Running Code in Jupyter

This notebook is based on [Lectures on scientific computing with Python](#) by J.R. Johansson.

First and foremost, the Jupyter Notebook is an interactive environment for **writing and running code**. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the IPython kernel, therefor runs Python code.

1.1 Code cells

Code cells allow you to enter and run code. Run a code cell by pressing the  button in the bottom-right panel, or **Control+Enter** on your hardware keyboard.

```
[1]: a = 10
```



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



10

There are a couple of keyboard shortcuts for running code:

- **Control+Enter** run the current cell and enters command mode.
- **Shift+Enter** runs the current cell and moves selection to the one below.
- **Option+Enter** runs the current cell and inserts a new one below.

1.2 Managing the kernel

Code is run in a separate process called the **kernel**, which can be interrupted or restarted. You can see kernel indicator in the top-right corner reporting current kernel state:  means kernel is **ready** to execute code, and  means kernel is currently **busy**. Clicking kernel indicator will open **kernel menu**, where you can reconnect, interrupt or restart kernel.

Try running the following cell — kernel indicator will switch from  to , i.e. reporting kernel as “busy”. This means that you won’t be able to run any new cells until current execution finishes, or until kernel is interrupted. You can then go to kernel menu and select “Interrupt”.

```
[3]: import time
time.sleep(10)
```

If kernel dies you will be prompted to restart it.

1.3 Run

Pressing “Run” button () runs the code inside the cell

1.4 Restarting kernels

Kernel maintains the state of a notebook’s computations. You can reset this state by restarting the kernel. This is done by going to the kernel menu by tapping the kernel indicator in the top-right corner and selecting “Restart”.

1.4.1 sys.stdout and sys.stderr

The `stdout` and `stderr` streams are displayed as text in the output area.

```
[4]: print("hi, stdout")
```

hi, stdout

```
[5]: import sys
     print('hi, stderr', file=sys.stderr)
```

hi, stderr

1.5 Output is asynchronous

All output is displayed asynchronously as it is generated in the kernel. If you execute the next cell, you will see the output one piece at a time, not all at the end.

```
[6]: import time, sys
     for i in range(8):
         print(i)
         time.sleep(0.5)
```

0
1
2
3
4
5
6
7