00_introduction_notebooks

August 28, 2021

1 Introduction to python shell, porgrams and jupyter notebooks

Welcome to the python tutorial of start sience here! This tutorial will guide you through all the important topics in learning the python programming language.

Python is an interpreted, high-level, general purpose programming language.

- It's interpreted because python programs don't need to be compiled.
- It's a high-level programming language, because of its layer of abstraction. You don't need to write instructions which memory register to pass through which core of the CPU. This would also prohibit us from sharing code, between different processors.
- It's general purpose, because you can do almost anything with it. From math, websites, inventory management systems to computational chemistry.

1.1 The python shell

For our first python code we want to get to a python terminal. This can be done in many different ways:

1. If you are currenlty using a linux operating system, you are good to go. Just open a terminal and continue.
2. If you are on a Windows PC you can either install the Windows subsystem for linux
3. or download a python installer, install it and open a python shell by searching for "python" in your start menu.
4. If you don't want to do any of that or if you are on a mobile device, you can use the terminal provided by binder: https://kevinsawade.github.io/start-science-here/linux_tutorial.html#opening-a-terminal-on-binder

For all cases (except 2) we also want to enter `python` into our current shell to start the python shell. After that we are greeted by `>>>`, the python prompt. Which tells us, we can start to write python code. Try typing the following code line-by-line hitting return after every line and omitting the three chevrons `>>>` at the start. The lines without chevrons are the output of the previous line, so you can check whether the expected output is correct.

```
>>> 1 + 3
4
>>> a = 5
>>> a
5
>>> 1 + a
6
```

```
>>> print('Hello World!')
Hello World!
```

To leave the python shell type `quit()`:

```
quit()
```

1.2 A python program

Writing code that way is not very convenient. You have to retype quite a lot and making changes to something would have you re-typing quite a lot. That's why we put python code into file (normally with the `.py` file extension). If you tell python to run on a file it will execute the code in that file line-after-line. These files are basically complete python programs. Try creating a simple text file called `my_awesome_program.py` and put this line of code into it:

```
print('You are awesome!')
```

For linux/WSL/mybinder users

You can use either `nano` to create the file.

```
$ nano my_awesome_program.py
```

Write the code and save it by hitting (`Ctrl` and `X` to save and `y` to confirm). You can also put the contents into the file using `echo`:

```
$ echo "print('You are awesome!')" > my_awesome_program.py
$ # check contents with
$ cat my_awesome_program.py
```

For windows users

Create a new text file with the text editor of you choice. For this basic program the windos notepad (aka editor) will suffice. Make sure the file extension is `.py` and rename the file if not. You can now double-click the file, which will run it with python. **However,** after it has finished running the window will close automatically and you probably won't be able to read the output of the program. We need a windows shell to run the program. Open the Windows command prompt by hitting the `Windows Key` + `X` and navigate to the location of the file. If you saved the file to a different drive, than the standard `C:\` drive, you can change to that drive by calling simply:

```
> D:
```

Now you can navigate to the file. Use backslashes `\` and not slashes as path separators and use `cd ..` to go back one directory.

Run the program

Run your program on either of the command lines/shells by calling:

```
$ python my_awesome_program.py
```

on bash and

```
> python my_awesome_program.py
```

on Windows Command prompt. This usage of python is still not very useful when developing code. That's why we are using *jupyter-notebooks*.

1.3 Jupyter Notebooks

If you want to run the jupyter notebooks with the python tutorials locally, follow the instructions on [setting up python(https://kevinsawade.github.io/start-science-here/setting_up_python.html).

But for the python tutorials it will be easiest to head over to binder and run them there: https://mybinder.org/v2/gh/kevinsawade/start-science-here/HEAD

jupyter-notebooks contain a number of so-called cells. Cells can either be code cells, or markdown cells. This text is written in a markdown cell. In these cells text is easier to read and can be formatted *like* **these** ~~examples~~ here. You can make lists, tables and format your text nicely. These markdown cells accompany the code cells. In the code cells the magic happens. These cells contain code segments which can be executed on a cell-by-cell basis. All variables declared in the code cells are available as long as the notebook is running. This means that contrary to a code.py file a variable declared in a cell further down can be used in a cell further up, as long as the cell with the variable declaration is ran first.

How to execute code in jupyter notebooks

To execute a cell you can either use the "Run" button at menu bar on top, our you can use these key combinations:

- `Shift` + `Enter` to run a cell and move the cursor to the next cell.
- `Ctrl` + `Enter` to run a cell without moving the cursor to the next cell.

Further PROs for jupyter notebooks

- You can render graphs and images.
- You can also render molecule structures.
- You can add additional functionality like exercises, tables of contents for quick navigation of larger notebooks.
- You can describe your code via markdown cells.

1.4 Rendering a pdf of this notebook

This notebook can be renderd to html and pdf via these commands:

```
$ jupyter nbconvert --to pdf 00_introduction_notebooks.ipynb
$ jupyter nbconvert --to html 00_introduction_notebooks.ipynb
```