Work to be done:

Code Pack 01

A. Hello Python

1. Run python on cmd:

```
print(Hello world)

quit()
```

2. Run

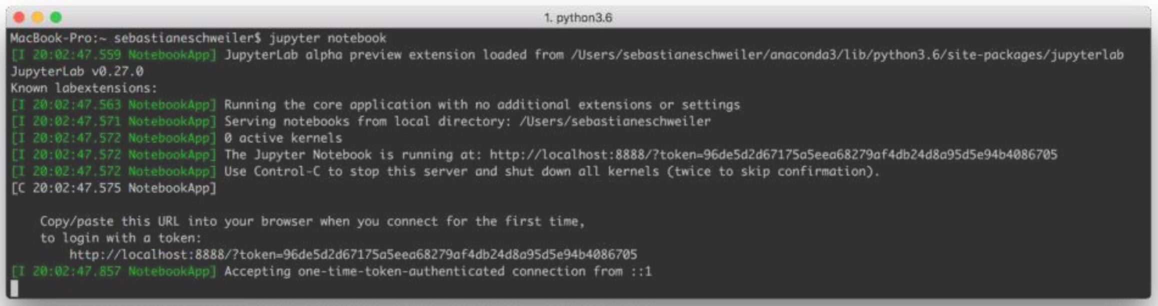
```
type helloworld.py

python helloworld.py
```

B. Travelling to Jupyter

Start Jupyter Notebook by using the following command:

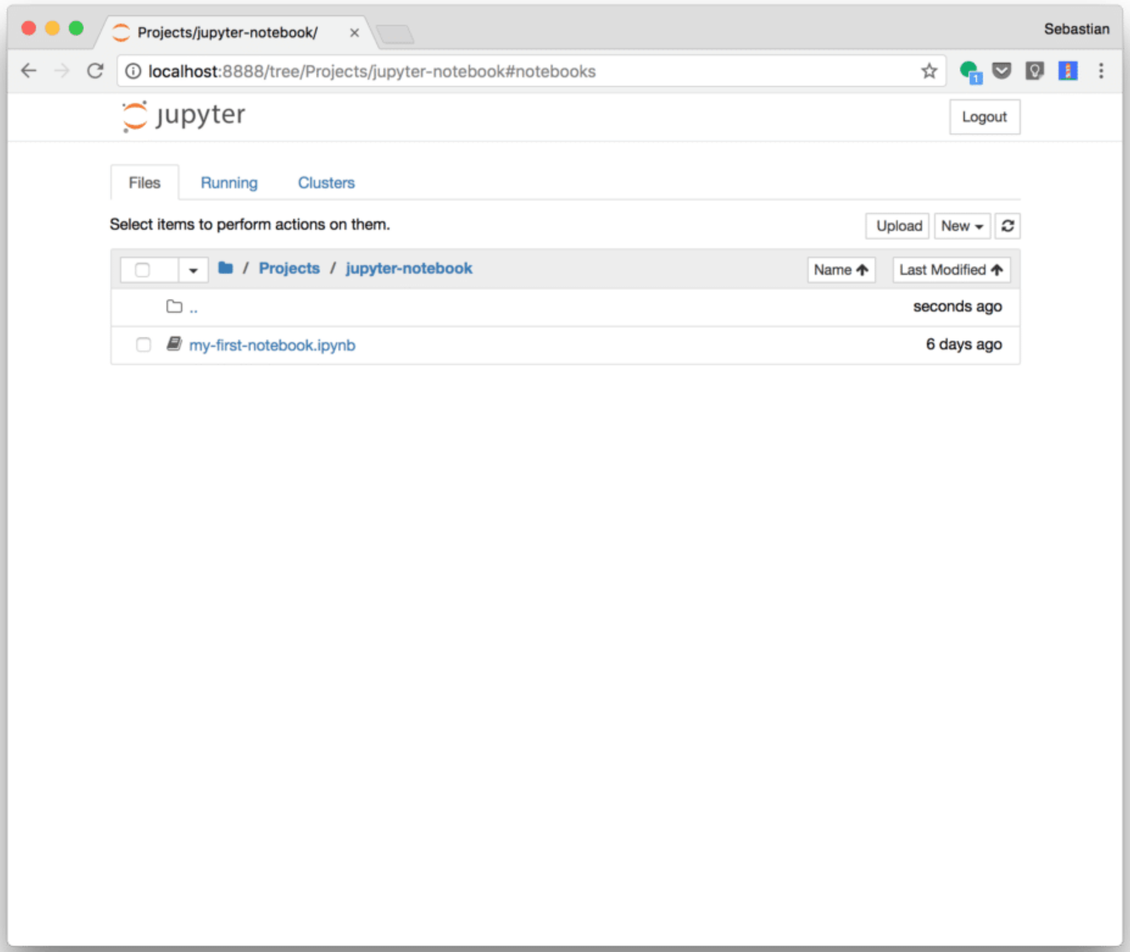
```
$ jupyter notebook
```

You’ll see the following response on the command line:



```
MacBook-Pro:~ sebastianeschweiler$ jupyter notebook
[I 20:02:47.559 NotebookApp] JupyterLab alpha preview extension loaded from /Users/sebastianeschweiler/anaconda3/lib/python3.6/site-packages/jupyterlab
JupyterLab v0.27.0
Known labextensions:
[I 20:02:47.563 NotebookApp] Running the core application with no additional extensions or settings
[I 20:02:47.571 NotebookApp] Serving notebooks from local directory: /Users/sebastianeschweiler
[I 20:02:47.572 NotebookApp] 0 active kernels
[I 20:02:47.572 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=96de5d2d67175a5eea68279af4db24d8a95d5e94b4086705
[I 20:02:47.572 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:02:47.575 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=96de5d2d67175a5eea68279af4db24d8a95d5e94b4086705
[I 20:02:47.857 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

The web server is started and the Jupyter Notebook application is opened in your default browser automatically. You should be able to see a browser output which is similar to the following screenshot:


Projects/jupyter-notebook/
Sebastian
localhost:8888/tree/Projects/jupyter-notebook#notebooks
jupyter
Logout
Files
Running
Clusters
Upload
New
Name
Last Modified

Select items to perform actions on them.

/ Projects / jupyter-notebook

.. seconds ago

my-first-notebook.ipynb 6 days ago

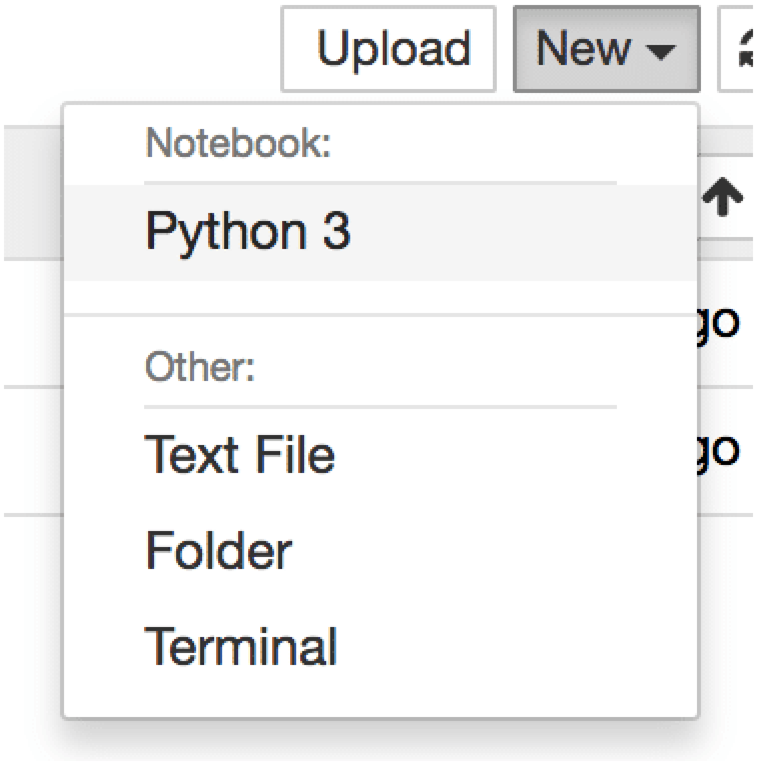
As you can see the user interface of Jupyter Notebook is split up into three sections (tabs):

- Files
- Running
- Clusters

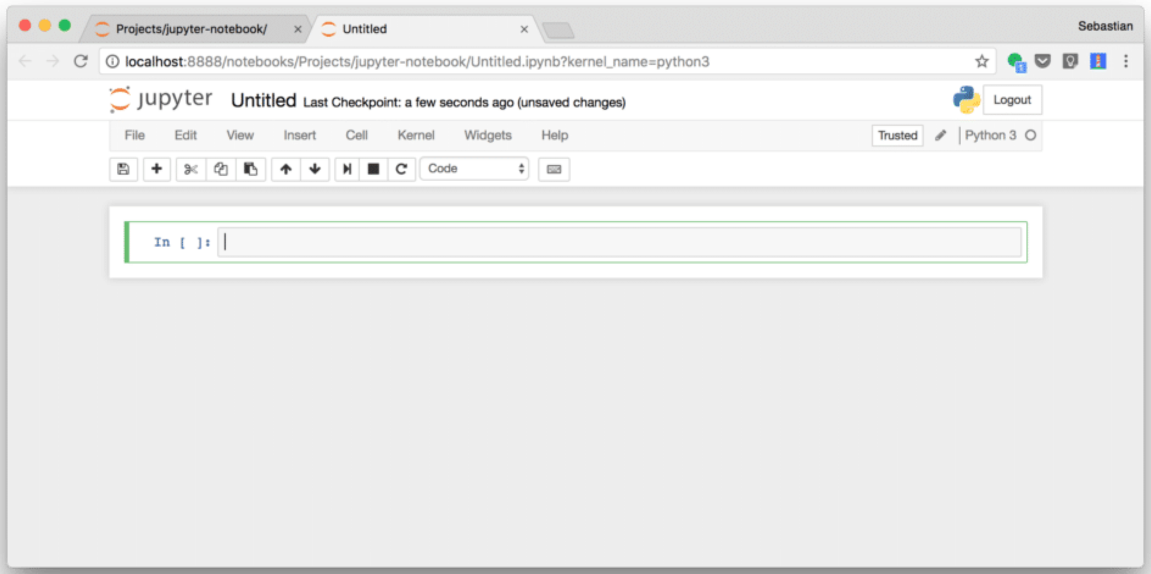
The default view is the *Files* tab from where you can open or create notebooks.

Creating A New Notebook

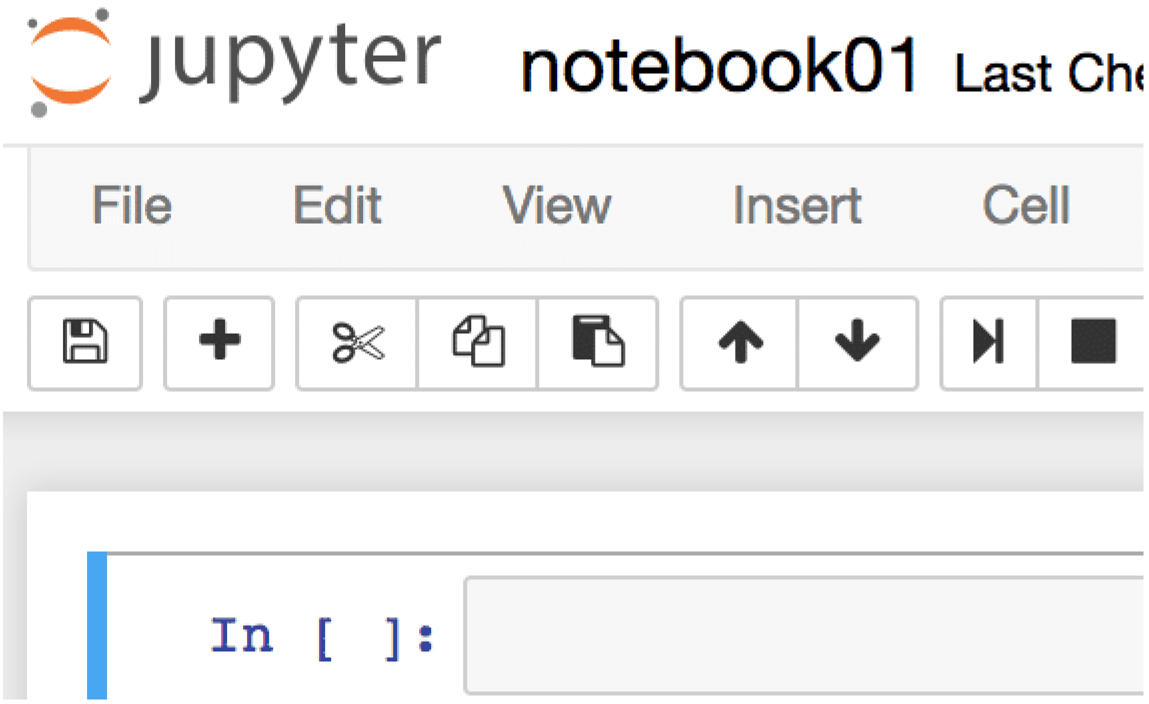
Creating a new Jupyter Notebook is easy. Just use the *New* dropdown menu and you'll see the following options:



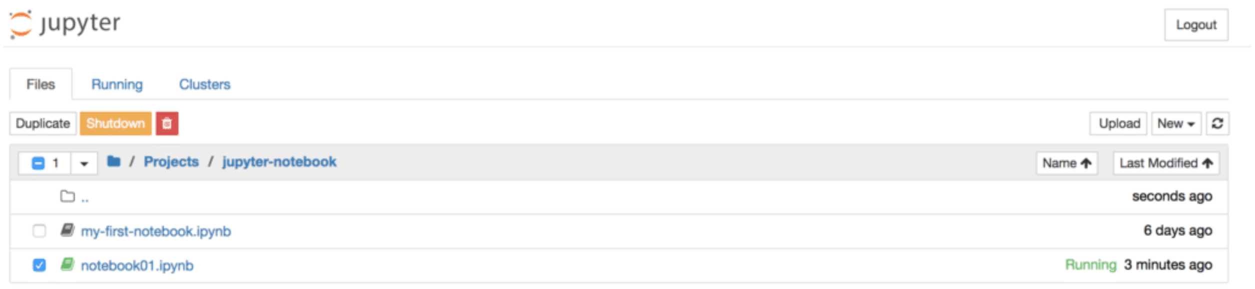
Select option *Python 3* to open a new Jupyter Notebook for Python. The notebook is created and you should be able to see something similar to:


Projects/jupyter-notebook/
Untitled
Sebastian
localhost:8888/notebooks/Projects/jupyter-notebook/Untitled.ipynb?kernel_name=python3
jupyter
Untitled
Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Code
In [ ]:

The notebook is created but still untitled. By clicking into the text “Untitled” on the top you can give it a name. By giving it a name the notebook will also be saved as a file of the same name with extension *.ipynb*. E.g. name the notebook *notebook01*:



Switching back to the Files tab you’ll be able to see a new file *notebook01.ipynb*:



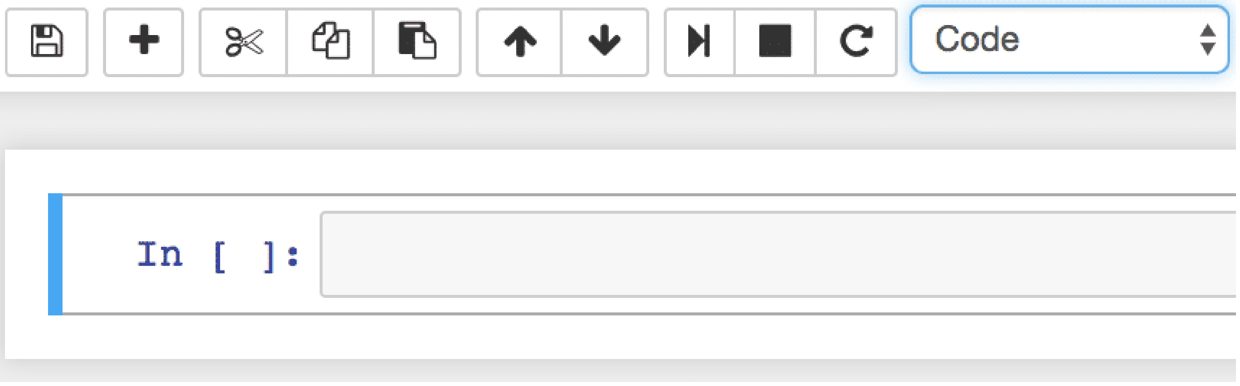
..	seconds ago
my-first-notebook.ipynb	6 days ago
notebook01.ipynb	Running 3 minutes ago

Because this notebook file is opened right now the file is marked with status *Running*. From here you can decided to shutdown this notebook by clicking on button *Shutdown*.

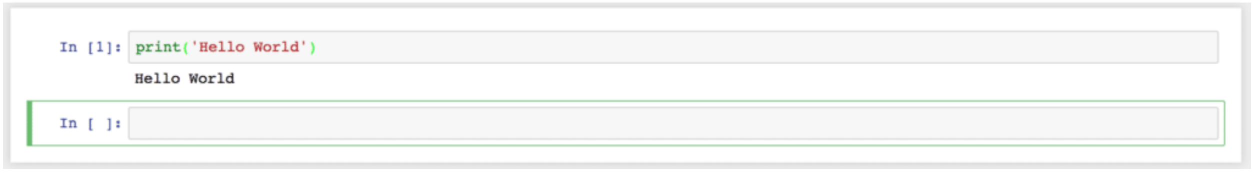
However before shutting down the notebook let’s switch back to the notebook view and try out a few things to get familiar with the notebook concept.

Working With The Notebook

The notebook itself consists of cells. A first empty cell is already available after having created the new notebook:



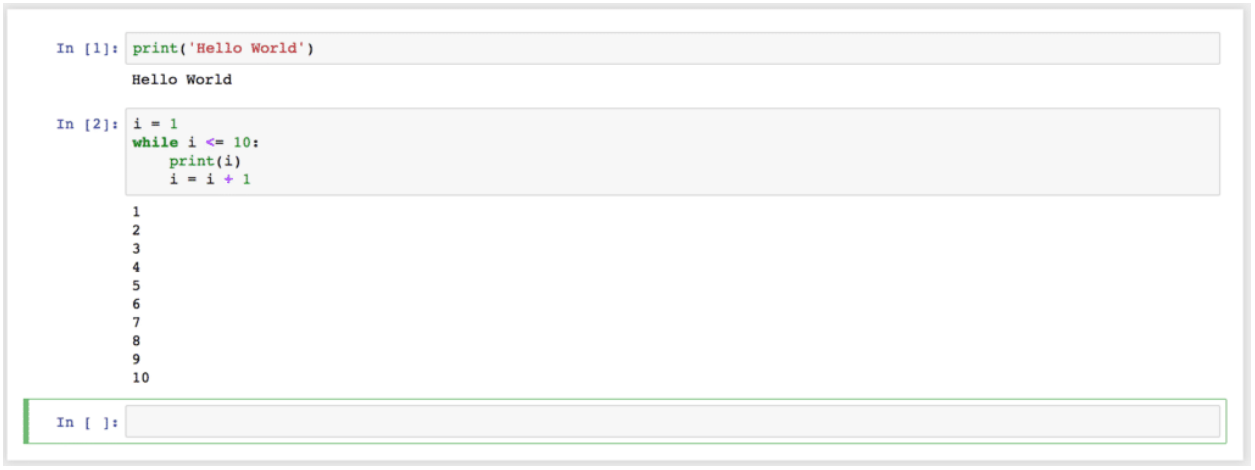
This cell is of type “Code” and you can start typing in Python code directly. Executing code in this cell can be done by either clicking on the *run cell* button or hitting Shift + Return keys:



```
In [1]: print('Hello World')

        Hello World

In [ ]:
```

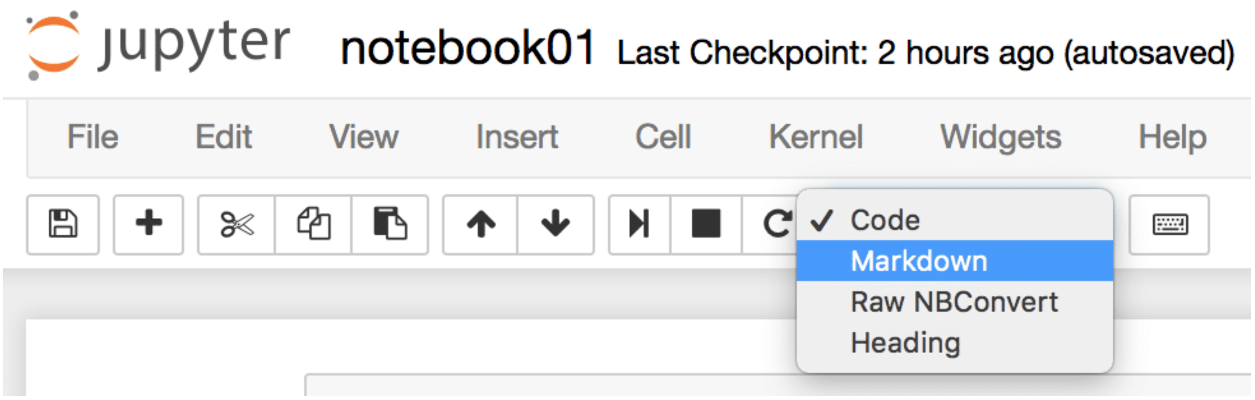
The resulting output becomes visible right underneath the cell.

The next empty code cell is created automatically and you can continue to add further code to that cell. Just another example:

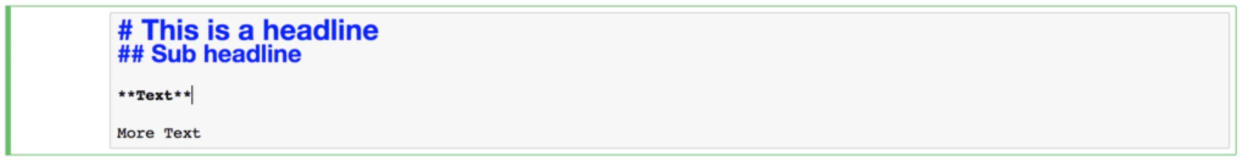


```
In [1]: print('Hello World')

        Hello World

In [2]: i = 1
        while i <= 10:
            print(i)
            i = i + 1

        1
        2
        3
        4
        5
        6
        7
        8
        9
        10

In [ ]:
```

You can change the cell type from *Code* to *Markdown* to include explanatory text in your notebook. To change the type you can use the dropdown input control:

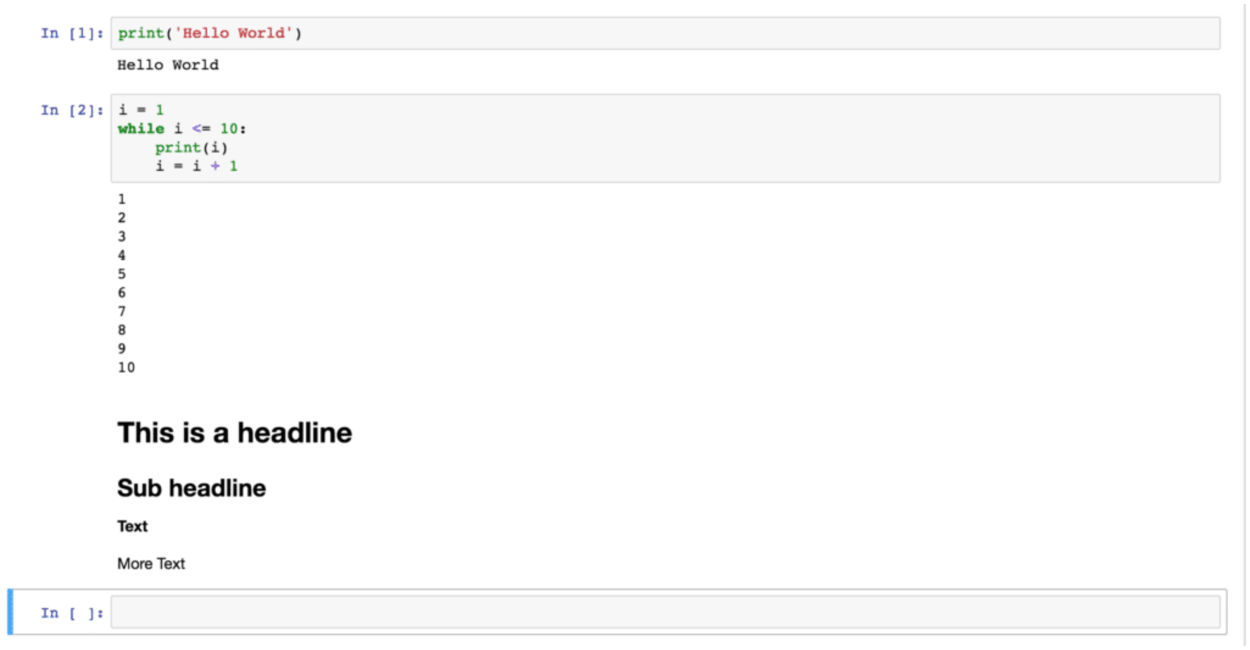


Once switched the type to *Markdown* you can start typing in markdown code:



```
# This is a headline
## Sub headline

**Text**

More Text
```

After having entered the markdown code you can compile the cell by hitting Shift + Return once again. The markdown editor cell is then replaced with the output:



```
In [1]: print('Hello World')
Hello World

In [2]: i = 1
        while i <= 10:
            print(i)
            i = i + 1
1
2
3
4
5
6
7
8
9
10
```

This is a headline

Sub headline

Text

More Text

```
In [ ]:
```

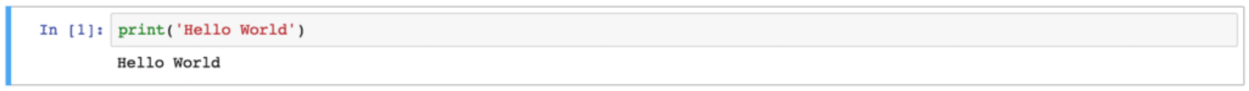
If you want to change the markdown code again you can simply click into the compiled result and the editor mode opens again.

Edit And Command Mode

If a cell is active two modes distinguished:

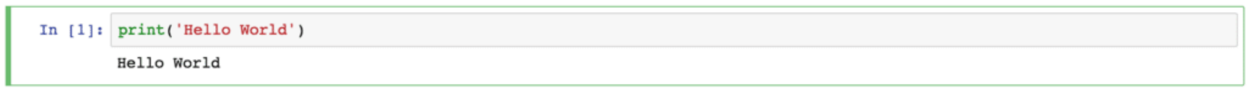
- edit mode
- command mode

If you just click in one cell the cell is opened in command mode which is indicated by a blue border on the left:



```
In [1]: print('Hello World')
        Hello World
```

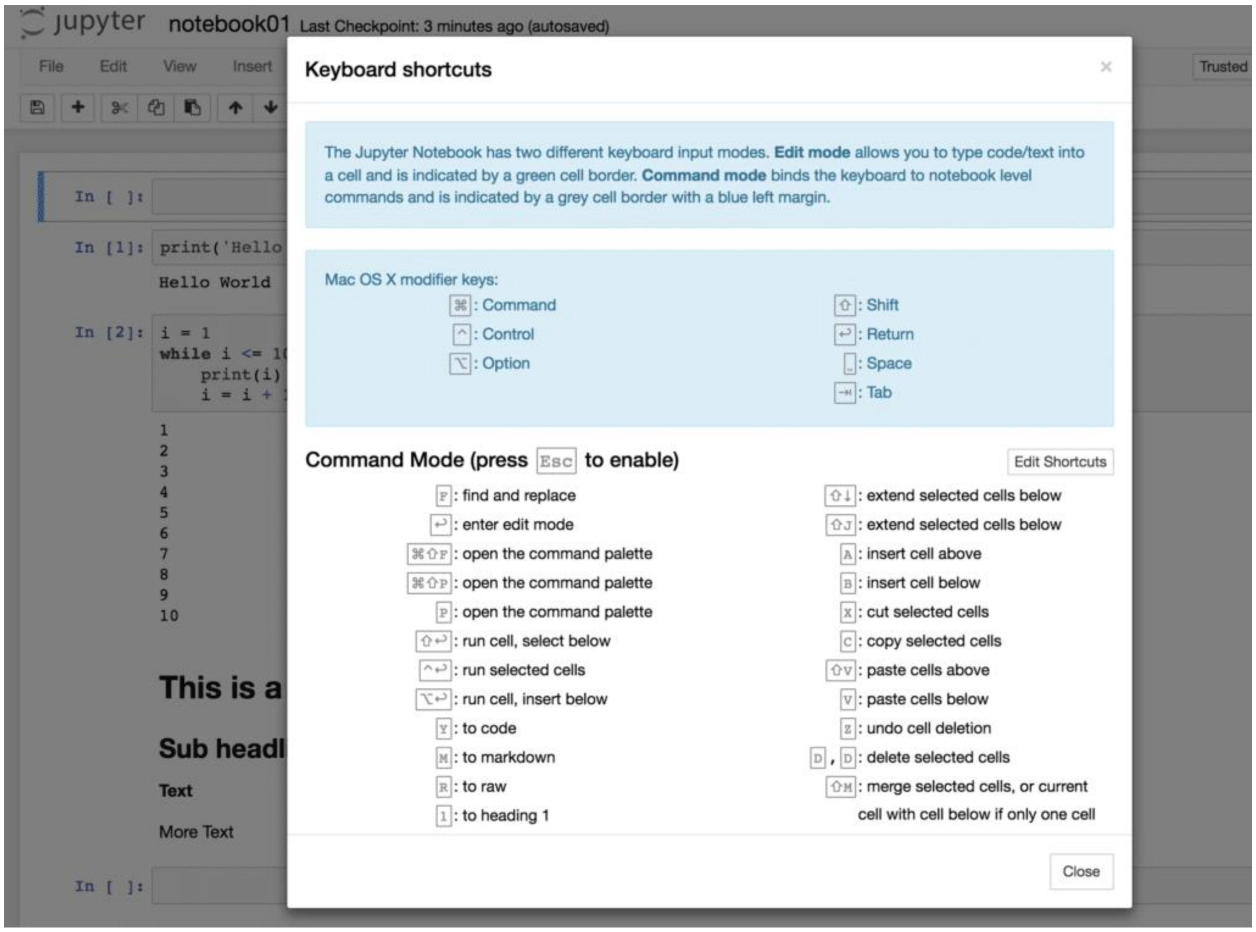
The edit mode is entered if you click into the code area of that cell. This mode is indicated by a green border on the left side of the cell:



```
In [1]: print('Hello World')
        Hello World
```

If you'd like to leave edit mode and return to command mode again you just need to hit ESC.

To get an overview of functions which are available in command and in edit mode you can open up the overview of key shortcuts by using menu entry *Help → Keyboard Shortcuts*:


jupyter notebook01 Last Checkpoint: 3 minutes ago (autosaved)
File Edit View Insert
Trusted

```
In [ ]:

In [1]: print('Hello
        Hello World

In [2]: i = 1
        while i <= 1
            print(i)
            i = i + 
        1
        2
        3
        4
        5
        6
        7
        8
        9
        10
```

This is a

Sub headl

Text

More Text

```
In [ ]:
```

Keyboard shortcuts

The Jupyter Notebook has two different keyboard input modes. **Edit mode** allows you to type code/text into a cell and is indicated by a green cell border. **Command mode** binds the keyboard to notebook level commands and is indicated by a grey cell border with a blue left margin.

Mac OS X modifier keys:

⌘ : Command	⇧ : Shift
^ : Control	↩ : Return
⌥ : Option	␣ : Space
	⇥ : Tab

Command Mode (press Esc to enable)

Edit Shortcuts

F : find and replace	⇧↓ : extend selected cells below
↩ : enter edit mode	⇧J : extend selected cells below
⌘⇧F : open the command palette	A : insert cell above
⌘⇧P : open the command palette	B : insert cell below
P : open the command palette	X : cut selected cells
⇧↩ : run cell, select below	C : copy selected cells
^↩ : run selected cells	⇧V : paste cells above
⌥↩ : run cell, insert below	V : paste cells below
Y : to code	Z : undo cell deletion
M : to markdown	D , D : delete selected cells
R : to raw	⇧M : merge selected cells, or current cell with cell below if only one cell
1 : to heading 1	

Close

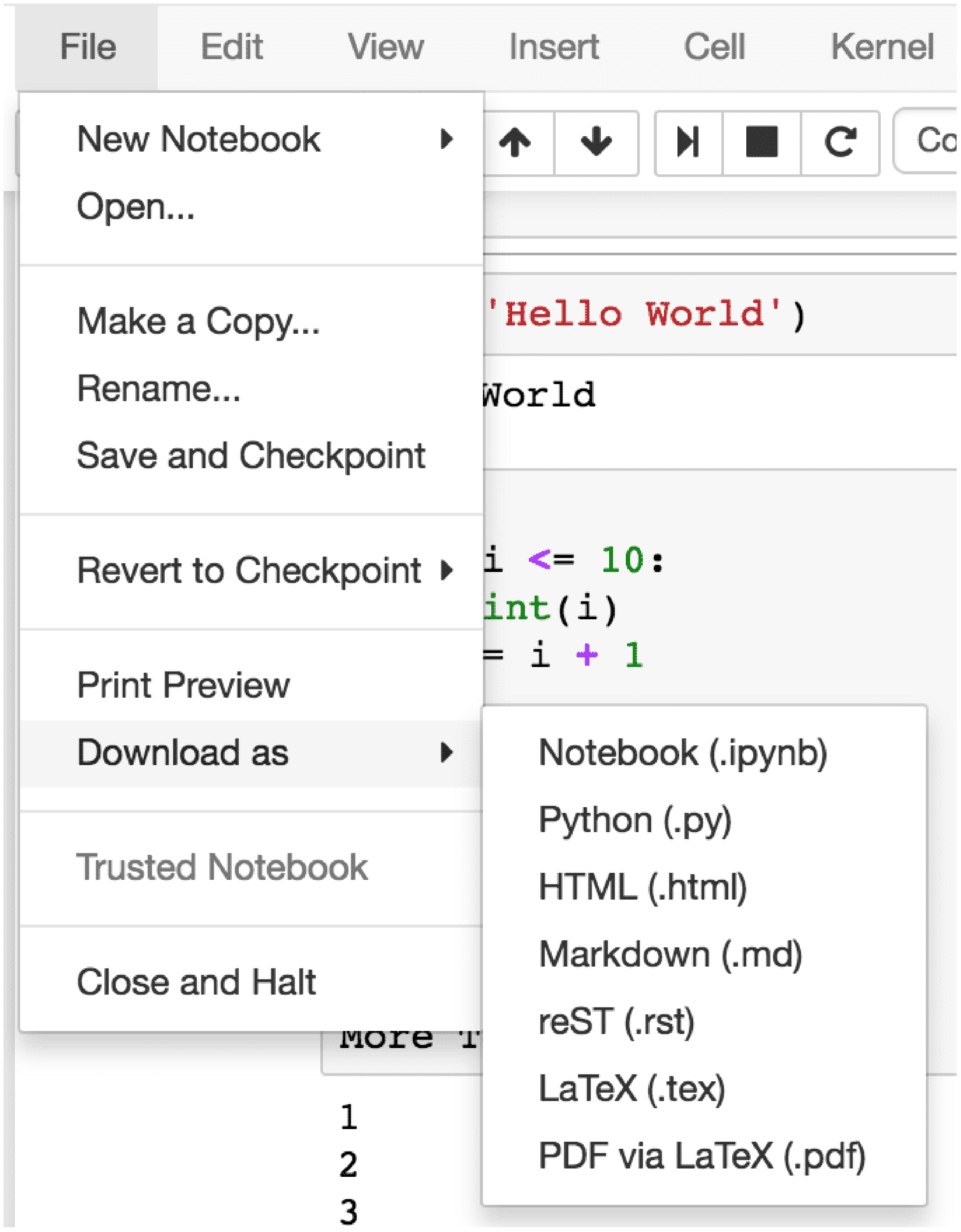
Checkpoints

Another cool function of Jupyter Notebook is the ability to create checkpoint. By creating a checkpoint you're storing the current state of the notebook so that you can later on go back to this checkpoint and revert changes which have been made to the notebook in the meantime.

To create a new checkpoint for your notebook select menu item *Save and Checkpoint* from the *File* menu. The checkpoint is created and the notebook file is saved. If you want to go back to that checkpoint at a later point in time you need to select the corresponding checkpoint entry from menu *File → Revert to Checkpoint*.

Exporting The Notebook

Jupyter Notebook gives you several options to export your notebook. Those options can be found in menu *File → Download as*:



```
'Hello World')
World
i <= 10:
int(i)
= i + 1
More T
1
2
3
```

C. Anaconda can bite you

Try:

```
conda

conda env list

conda activate

conda deactivate

conda activate base
```

Conda: Multiple Environment

```
conda  create  -n science  python=3  \
      numpy scipy matplotlib


conda env list

conda activate science


conda  create  -n data_science  --clone science  \
      pandas seaborn

conda  env  list

conda  remove  --name data_science  --all

conda deactivate
```

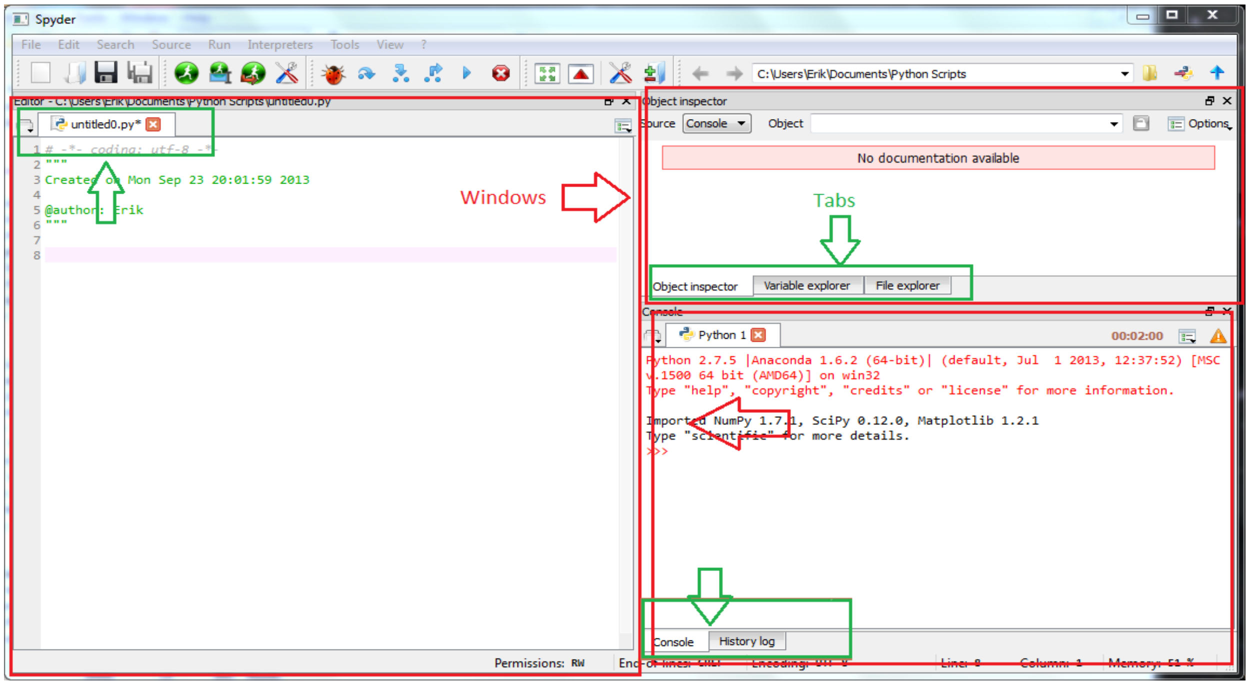
Conda: installing & updating

```
conda  install  holoviews

conda  update  holoviews

conda  update  --all

conda  remove  holoviews

conda  list
```

Code Pack 02

A. Spyder for now

After installing Anaconda, find the "Spyder" app in the installation folder or your programs menu, which will provide a nice user interface. Spyder may take a while to start up, because of all the python libraries it is loading.



```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 23 20:01:59 2013

@author: Erik
"""
```

```
Python 2.7.5 |Anaconda 1.6.2 (64-bit)| (default, Jul  1 2013, 12:37:52) [MSC
v.1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.

Imported NumPy 1.7.1, SciPy 0.12.0, Matplotlib 1.2.1
Type "scientific" for more details.
>>>
```

See F5 vs F9

Code Pack 03

A. Python fundamentals:

Follow the trainer:

1. Primitive Datatypes and Operators

2. Variables and Collections

Code Pack 04

A. Python fundamentals:

Follow the trainer:

3. Control Flow and Iterables

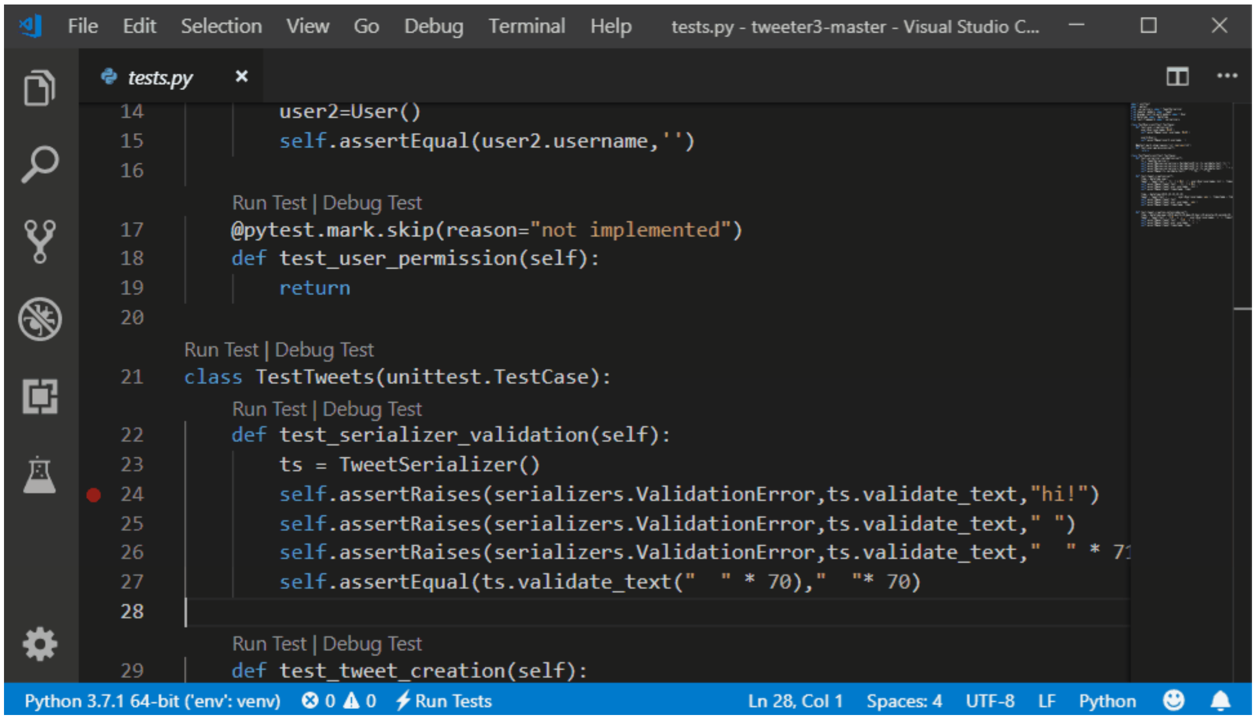
4. Functions

B. Port Scanning:

Open the notebook file and complete the function

Code Pack 05

A. Enter VS Code:


File Edit Selection View Go Debug Terminal Help
tests.py - tweeter3-master - Visual Studio C...
tests.py
Run Test | Debug Test
Python 3.7.1 64-bit ('env': venv)
0 0
Run Tests
Ln 28, Col 1
Spaces: 4
UTF-8
LF
Python

```
        user2=User()
        self.assertEqual(user2.username,'')

    @pytest.mark.skip(reason="not implemented")
    def test_user_permission(self):
        return

class TestTweets(unittest.TestCase):
    def test_serializer_validation(self):
        ts = TweetSerializer()
        self.assertRaises(serializers.ValidationError,ts.validate_text,"hi!")
        self.assertRaises(serializers.ValidationError,ts.validate_text,"  ")
        self.assertRaises(serializers.ValidationError,ts.validate_text,"  " * 71
        self.assertEqual(ts.validate_text("  " * 70),"  "* 70)

    def test_tweet_creation(self):
```

Add the Python extension

B. Python fundamentals:

Follow the trainer:

5. Modules

Code Pack 06

A. Python fundamentals:

Follow the trainer:

6. Classes

Code Pack 07

A. my_repl.py

Create a program that does the following behavior:

```
PROGRAM: Type something, then hit Enter:

Hello

PROGRAM: You typed: "Hello"

PROGRAM: Type something, then hit Enter:

What are you doing?

PROGRAM: You typed: "What are you doing?"

PROGRAM: Type something, then hit Enter:
```

B. bot_create_a_story.py

Open the file bot_create_a_story.py

Complete the bot by completing the code to fulfill the work part:

```
print('\n--- The Story of ' + name_a + ' and ' + name_b + '---\n')
print(name_a + ' and ' + name_b + ' were best friends who both lived in')
print('the peaceful land of ' + location + '. One day, they saw a ' + adjective)
print('grizzly bear wreaking havok in the streets. They ' + adverb + ' got their')
print('swords out and slew the beast.')
print('... The End.\n')
```

C. distance.py

Open the gif file **interactive-python-script.gif**

Make sure this is possible.

Code Pack 08

A. Create and document mymath.py

Create a module called **mymath.py** and create a basic **add** function.

1. Document it

2. Test it

Use `doctest.testmod()`

and `doctest.testfile('add.txt')`

you have the **add.txt** on your folder.

Code Pack 09

A. Create a database with Text Files:

Your goal is to write a module with 4 functions:

add_student - accepts a parameter of **first_name** and writes to a file called **students.txt.**

find_student - accepts a parameter of **first_name** and returns the first student found

update_student - accepts a parameter of **first_name** and **new_name** and updates first student found

remove_student - accepts a parameter of **first_name** and removes the student from the text file

try them with the file use_db.py

B. Working with CSV:

See these :

read_csv_1.py

read_csv_2_DictReader.py

write_csv_1.py

write_csv_2_DictWriter.py

Code Pack 10

A. Try the Web Scraping Code

Well, this is not my code.

But Is very well done… and it is in Portuguese!

Code Pack 11

A. Python fundamentals:

Follow the trainer:

6. Advanced

Code Pack 12

A. See the files:

1.ChainMap

2.Counter

3.defaultdict

4.deque

5.namedtuple

Code Pack 13

A. See the files:

1.Iterators

2.Generators

Code Pack 14

A. See the files:

1.The_Infinite_Iterators

2.Iterators_That_Terminate

3.The_Combinatoric_Generators

2.Generators

Code Pack 15

A. See the files:

1.Creating_a_Context_Manager_class

2.Creating_a_Context_Manager_using_contextlib

3.contextlib.closing(thing)

4.contextlib.suppress(exceptions)

5.contextlib.redirect_stdout_redirect_stderr

6.ExitStack

7.Reentrant_Context_Managers

Code Pack 16

A. See the files:

1.unittest

2.coverage.py

Code Pack 17

A. See the files:

1.argparse

2.configparser

Code Pack 18

A. See the files:

1.Benchmarking

2.Profiling_Your_Code_with_cProfile

Code Pack 19

A. See the files:

Logging

Code Pack 20

A. Debug your code:

python -m pdb buggy.py 5

VS Code debug : F9, F5

Code Pack 21

A. See the files:

os_module.py

Code Pack 22

A. See the files:

1.threading

2.multiprocessing

3.The_cryptography_Package

Code Pack 23

A. See the files:

The use of Basic SQL Syntax

Object Relational Mappers

Code Pack 24

A. See the files:

1.any

2.enumerate

3.eval

4.filter

5.map

6.zip

Code Pack 25

A. Numpy:

Make the code

Code Pack 26

A. Scipy:

See the code

Code Pack 27

A. Matplotlib:

1-Figures_Subplots_and_layouts

2-Plotting_Methods_Overview

3-HowToSpeakMPL

4-Limits_Legends_and_Layouts

5-Artists

6-mpl_toolkits

Code Pack 28

A. Introduction to OpenCV:

1. Introduction to OpenCV with Python
2. Core Operations
3. Image Processing in OpenCV
4. Feature Detection and Description
5. Video Analysis
6. Camera Calibration and 3D Reconstruction
7. Computational Photography
8. Object Detection

Code Pack 29

A. Pandas:

Not the bear

Code Pack 30

A. Machine Learning:

1.regression_world_happiness

2.classification_finding_regions

3.cluster_countries

Code Pack 31

A. Full App:

Flask + Machine Learning + Pickle