

Work to be done:

Code Pack 01	1
A. Hello Python	1
B. Travelling to Jupyter	2
C. Anaconda can bite you	11
Code Pack 02	12
A. Spyder for now	12
Code Pack 03	12
A. Python fundamentals:	12

Code Pack 01

A. Hello Python

1. Run python on cmd:

```
print(Hello world)
quit()
```

2. Run

```
type helloworld.py
python helloworld.py
```

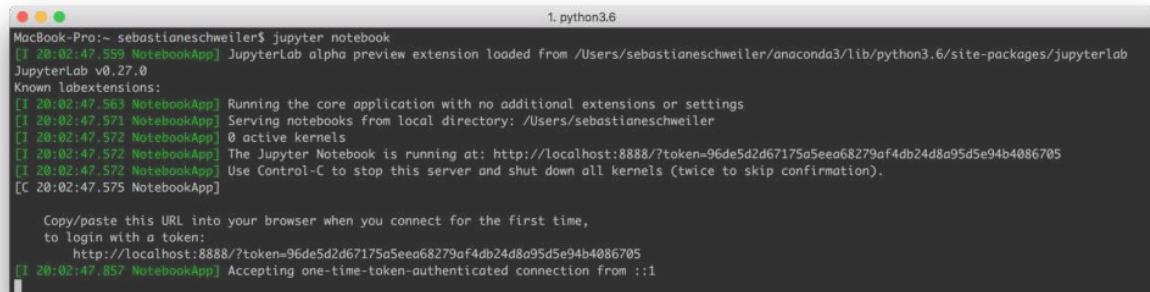
Coding Bootcamp Code in Python

B. Travelling to Jupyter

Start Jupyter Notebook by using the following command:

```
$ jupyter notebook
```

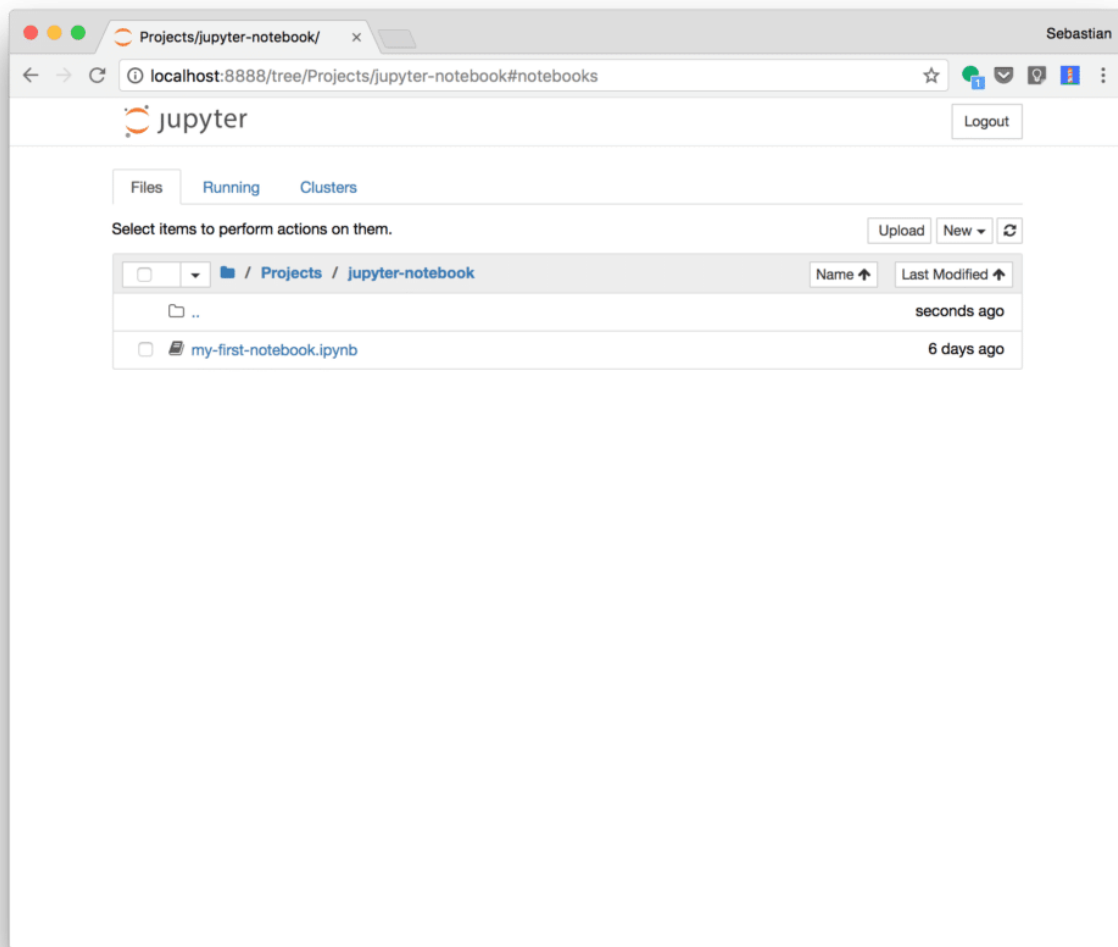
You'll see the following response on the command line:

A terminal window titled '1. python3.6' on a Mac. The prompt is 'MacBook-Pro:~ sebastianschweiler\$ jupyter notebook'. The output shows JupyterLab alpha preview extension loading, version v0.27.0, and known labextensions. It then runs the core application with no additional extensions or settings, serving notebooks from the local directory. It reports 0 active kernels and that the Jupyter Notebook is running at 'http://localhost:8888/?token=96de5d2d67175a5eea68279af4db24d8a95d5e94b4086705'. It instructs to use Control-C to stop the server. Finally, it provides a URL to login with a token and accepts a one-time-token-authenticated connection from '::1'.

```
MacBook-Pro:~ sebastianschweiler$ jupyter notebook
[I 20:02:47.559 NotebookApp] JupyterLab alpha preview extension loaded from /Users/sebastianschweiler/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/sebastianschweiler
[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).

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:



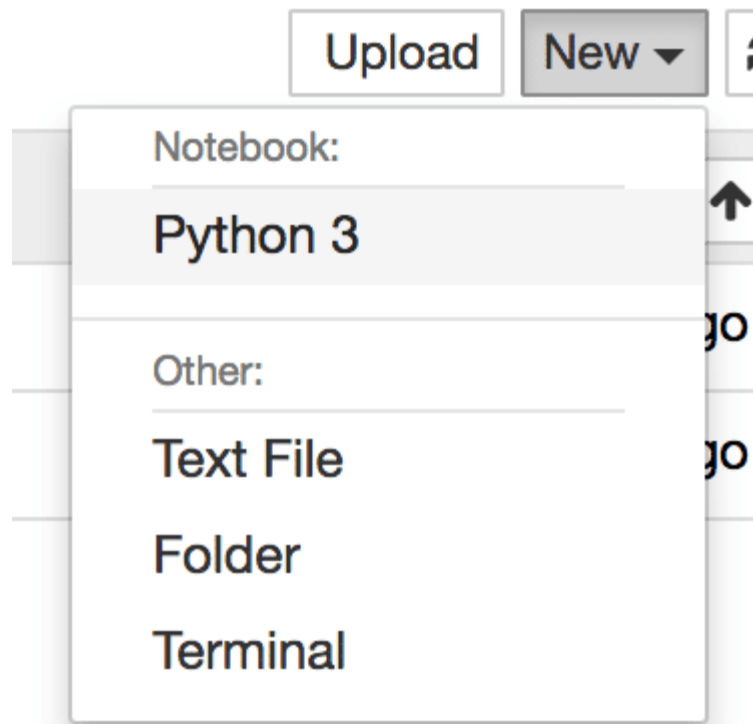
As you can see the user interface of Jupyter Notebook is split up into three sections (tabs):

- Files
- Running
- Clusters

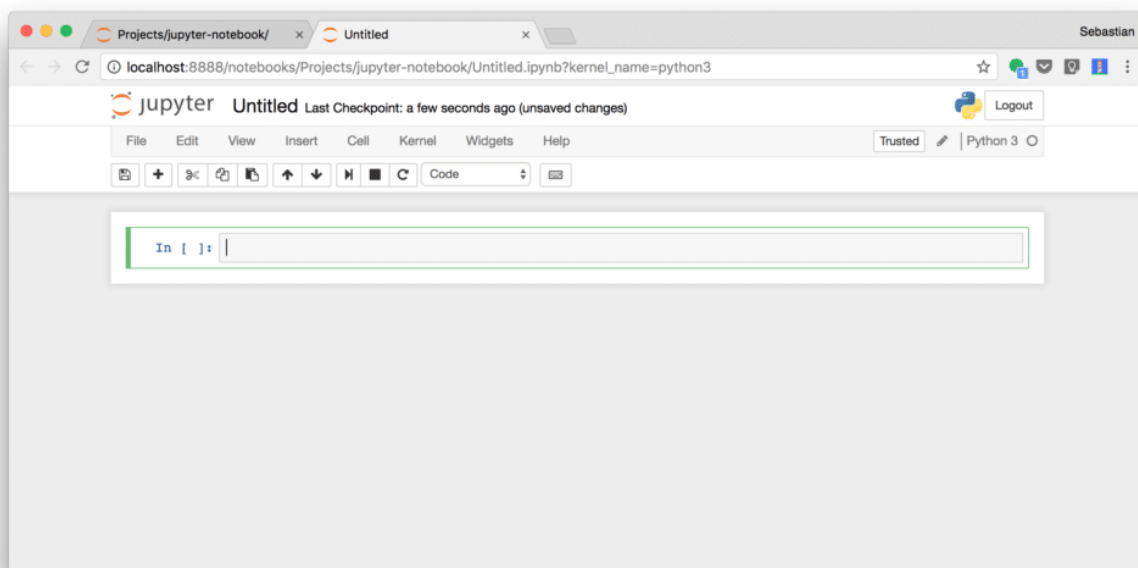
The default view is the *Files* tab from where you can open or create notebooks.

Creating A New Notebook

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

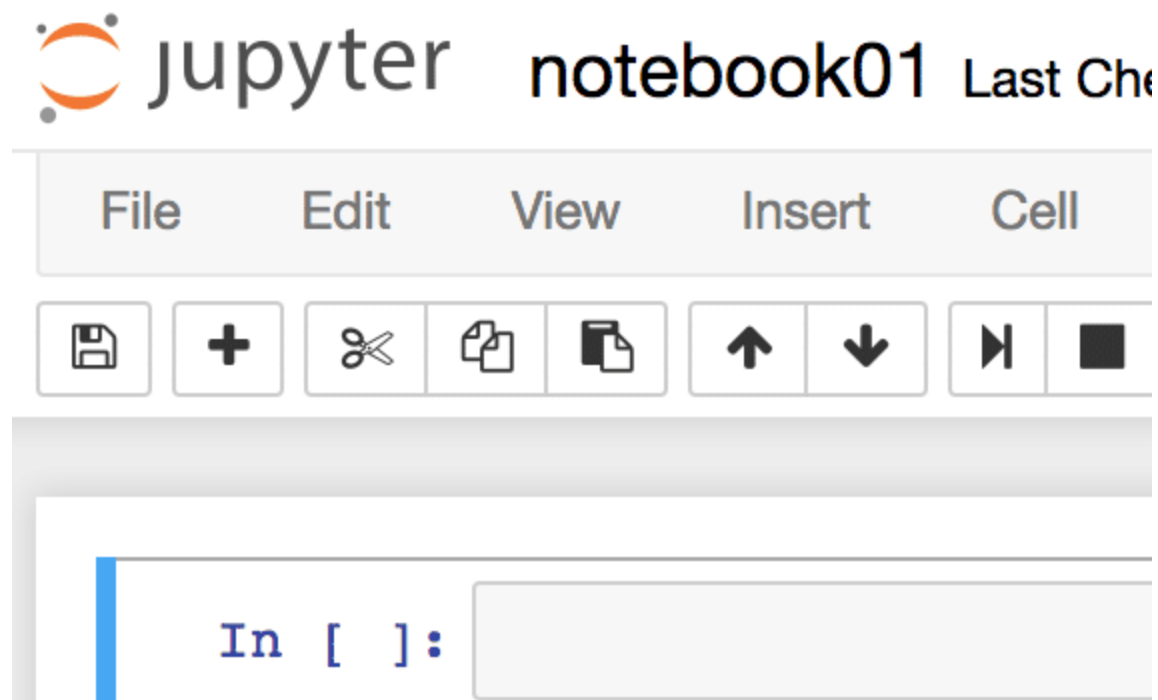


Select option *Python 3* to open a new Jupyter Notebook for Python. The notebook is created and you should be able to see something similar to:



Coding Bootcamp Code in Python

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*:



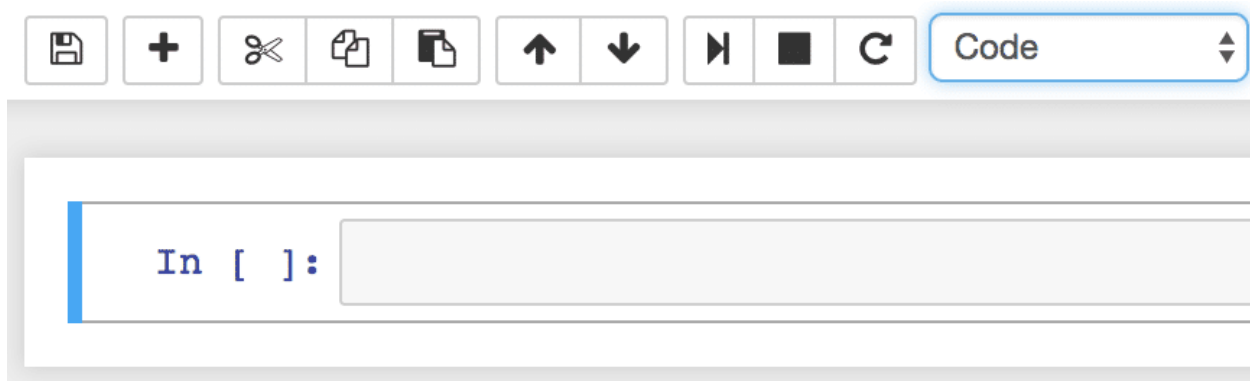
Because this notebook file is opened right now the file is marked with status *Running*. From here you can decide to shutdown this notebook by clicking on button *Shutdown*.

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.

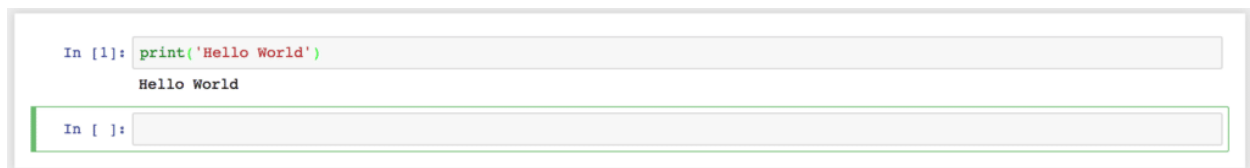
Working With The Notebook

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

Coding Bootcamp Code in Python

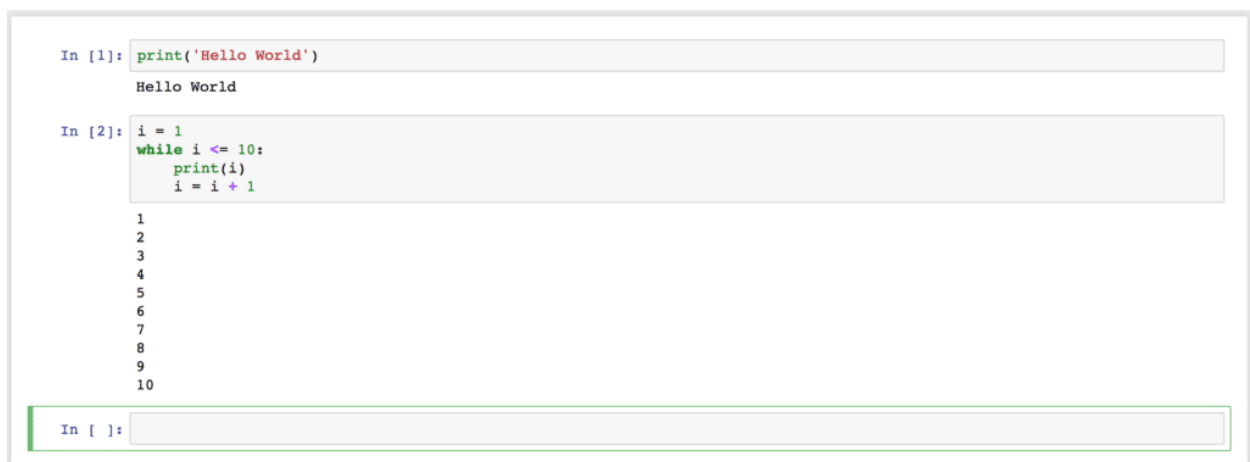


This cell is of type “Code” and you can start typing in Python code directly. Executing code in this cell can be done by either clicking on the *run cell* button or hitting Shift + Return keys:

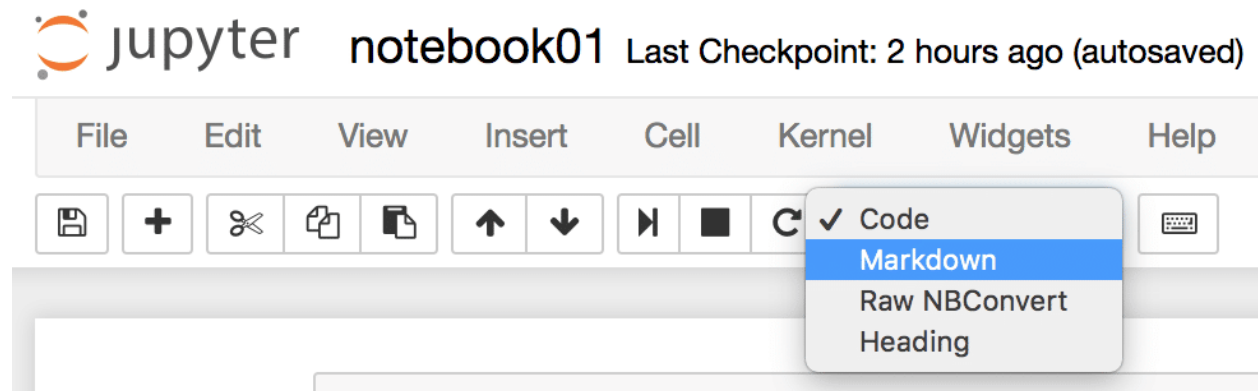


The resulting output becomes visible right underneath the cell.

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



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.

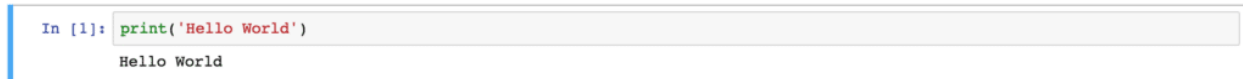
Edit And Command Mode

Coding Bootcamp Code in Python

If a cell is active two modes distinguished:

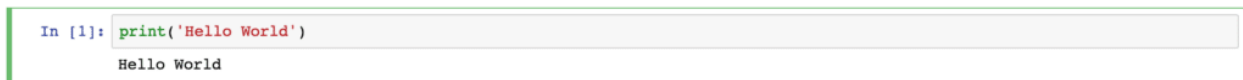
- edit mode
- command mode

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

A screenshot of a Jupyter Notebook cell. The cell has a light gray background. On the left side, there is a vertical blue bar, indicating the cell is in command mode. The text inside the cell is "In [1]: print('Hello World')" followed by "Hello World" on a new line. The code is color-coded: "In" is blue, "[1]" is blue, "print" is green, and the string is in red quotes.

```
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:

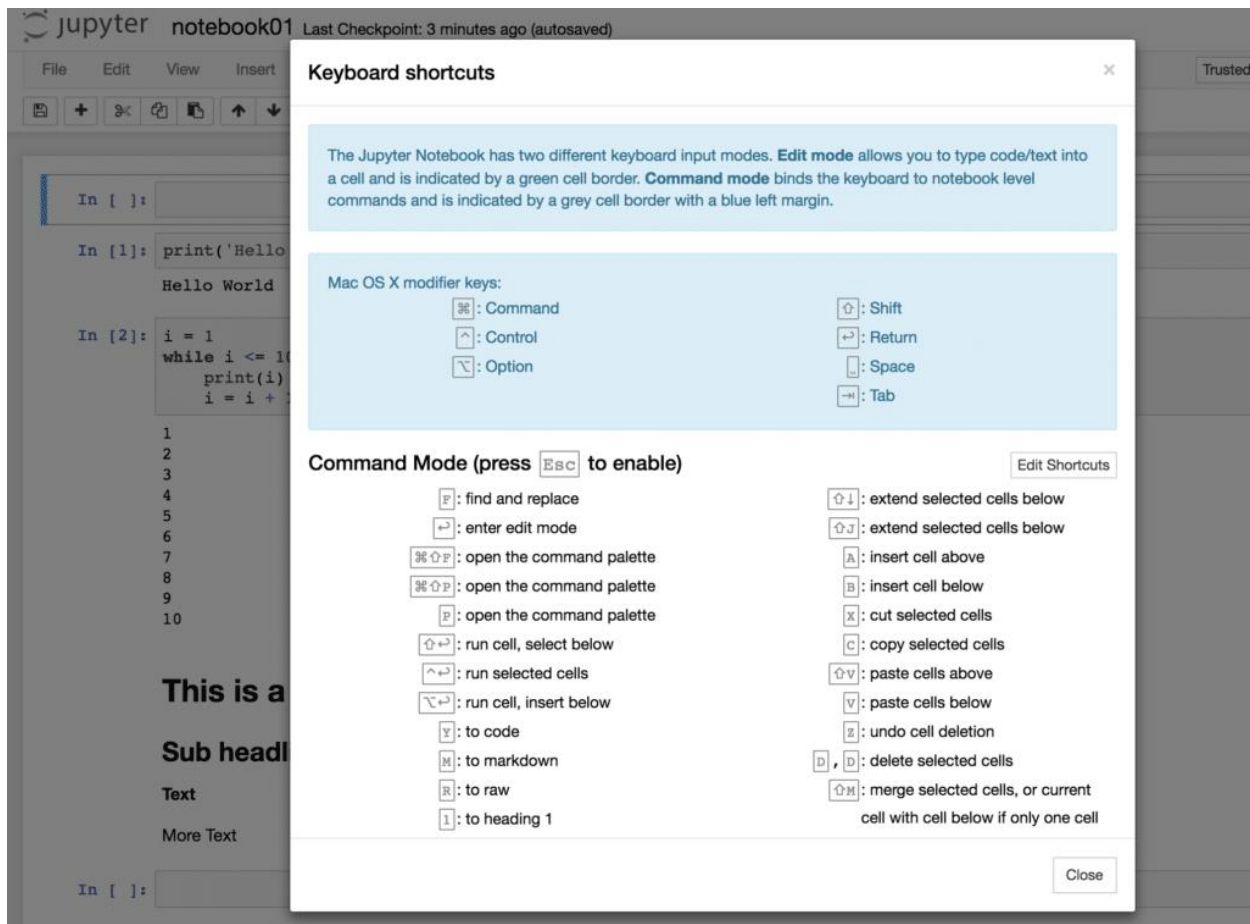
A screenshot of a Jupyter Notebook cell. The cell has a light gray background. On the left side, there is a vertical green bar, indicating the cell is in edit mode. The text inside the cell is "In [1]: print('Hello World')" followed by "Hello World" on a new line. The code is color-coded: "In" is blue, "[1]" is blue, "print" is green, and the string is in red quotes.

```
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*:

Coding Bootcamp Code in Python



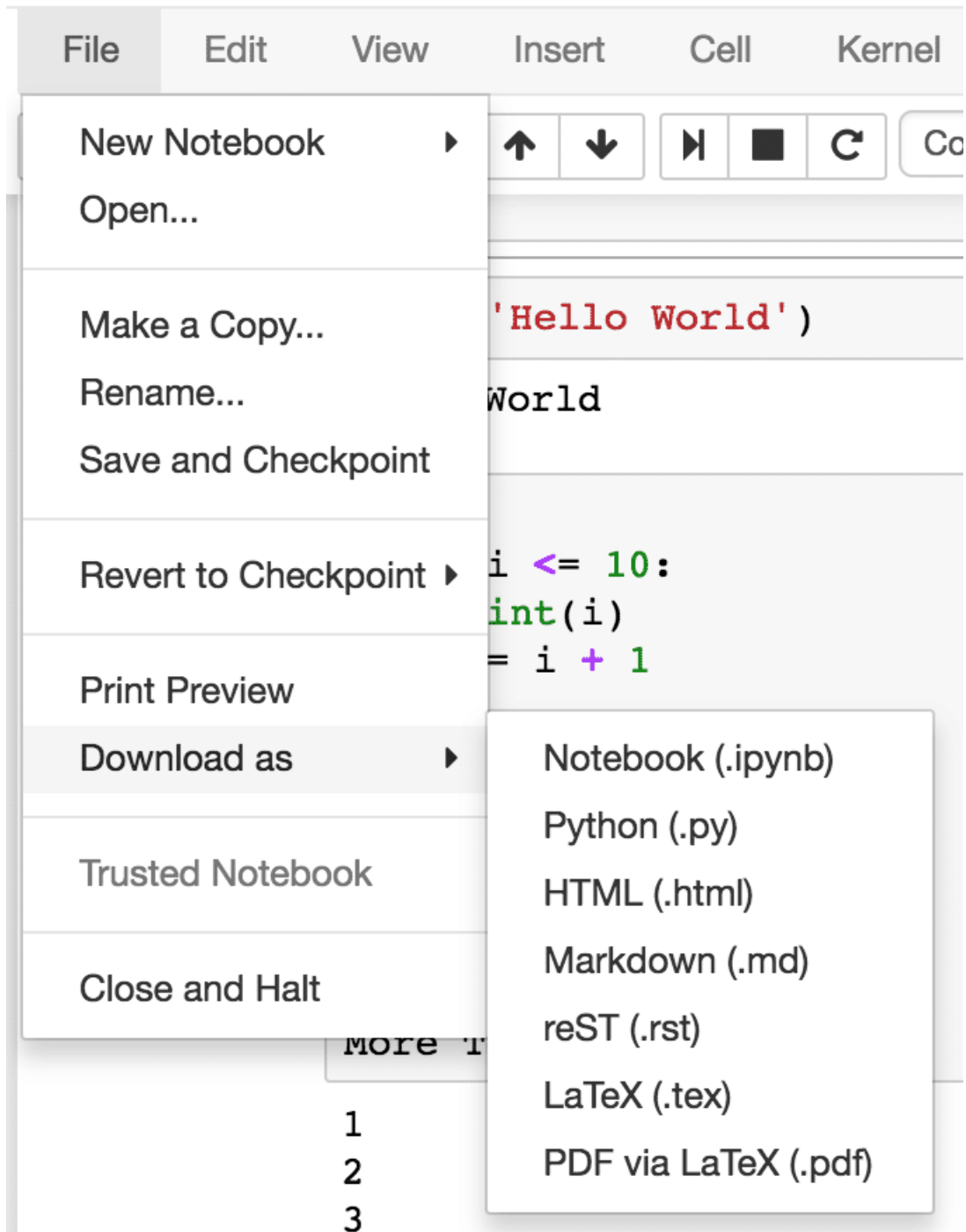
Checkpoints

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*:



Coding Bootcamp Code in Python

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
```

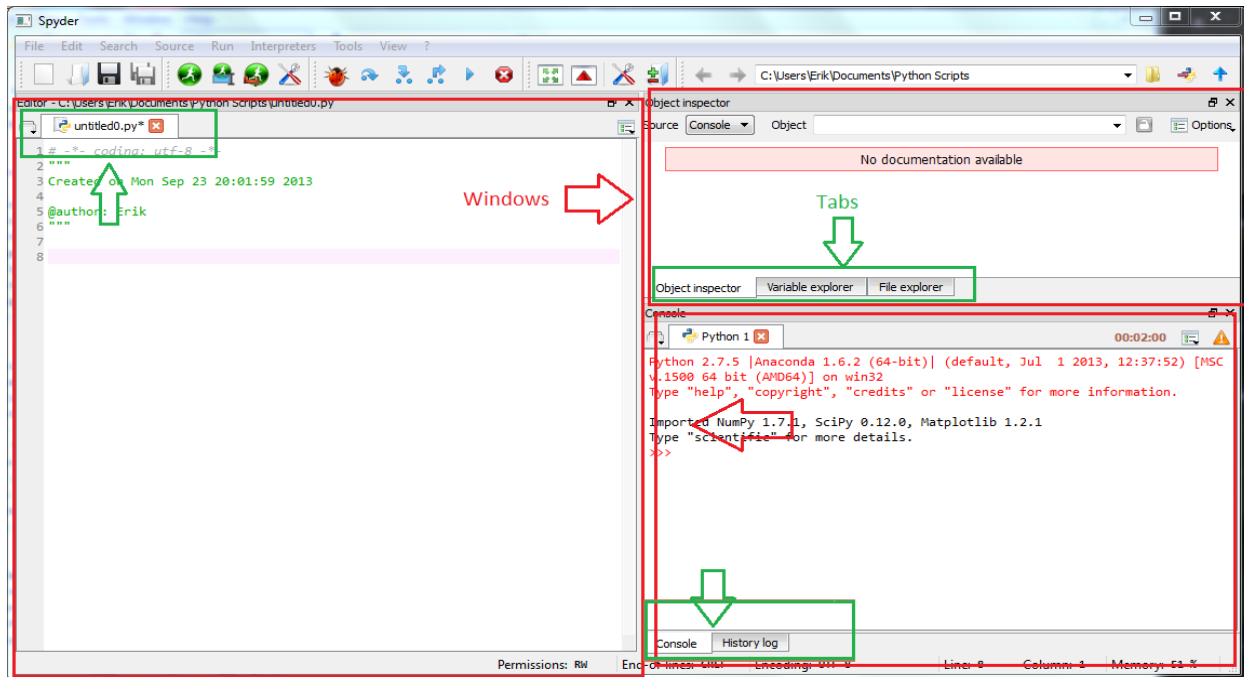
Conda: installing & updating

```
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.



See F5 vs F9

Code Pack 03

A. Python fundamentals:

Follow the trainer:

1. Primitive Datatypes and Operators
2. Variables and Collections