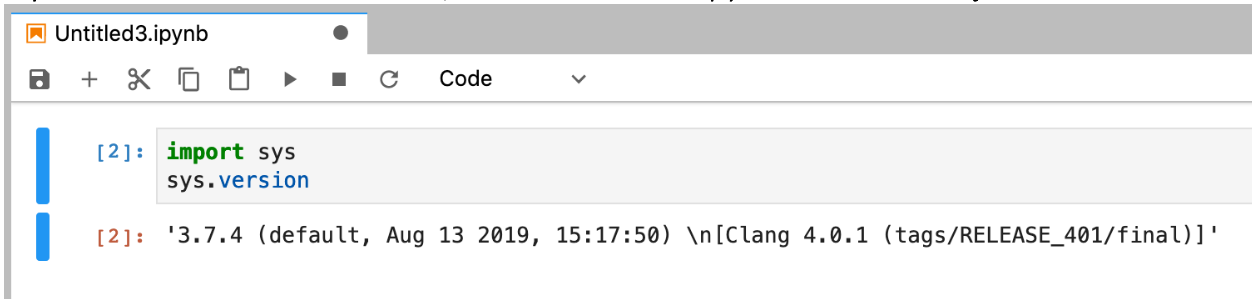
Lab: Jupyter Notebooks - Advanced Features 1

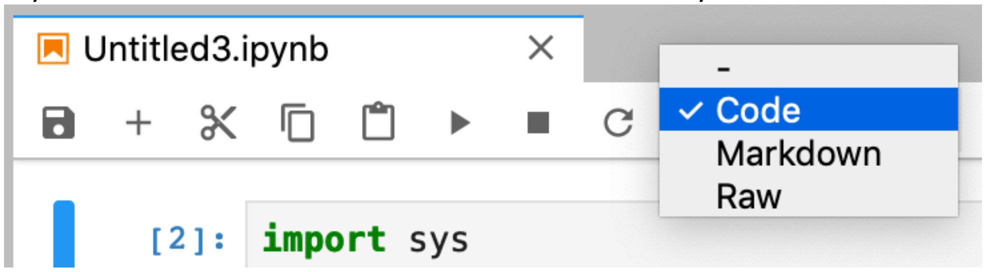
Welcome to the 2nd Jupyter Lab.
In this lab we'll explore some of the features which will help you becoming more productive using Jupyter.
If you remember from the last lab, we ended with a Jupyter notebook and just one cell:



```
[2]: import sys
     sys.version

[2]: '3.7.4 (default, Aug 13 2019, 15:17:50) \n[Clang 4.0.1 (tags/RELEASE_401/final)]'
```

If you have a look at the control bar of the Notebook you'll notice the following symbols:



```
[2]: import sys
```

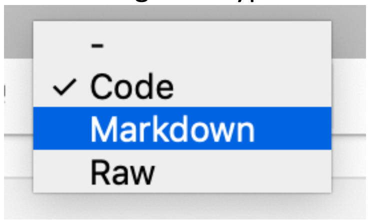
They are (in order from left to right): save, insert cell, cut cell, copy cell, paste cell, run cell, stop cell, restart kernel, change cell type between Code and Documentation.
Now since you know the icons and where to click we'll use shortcuts instead where possible to increase productivity. So it is always a good idea to save the notebook once in a while using Ctrl-S (Cmd-S on a Mac).
Now let's get started:

1. Hit the enter key to get into edit mode of the cell (alternatively just click on the cell with your mouse pointer)

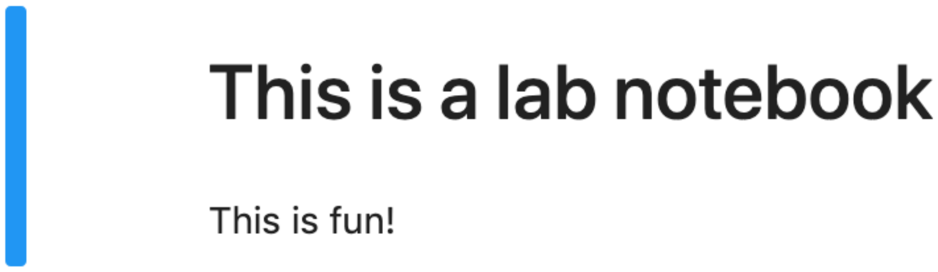
2. Hit the "+" symbol to create a new cell below

3. Enter the following two lines into the cell:
   # This is a Lab Notebook
   This is fun!

4. Now change the type of the cell to “Markdown” as indicated before



5. Execute the cell by hitting “Shift-Enter”



This is a lab notebook

This is fun!

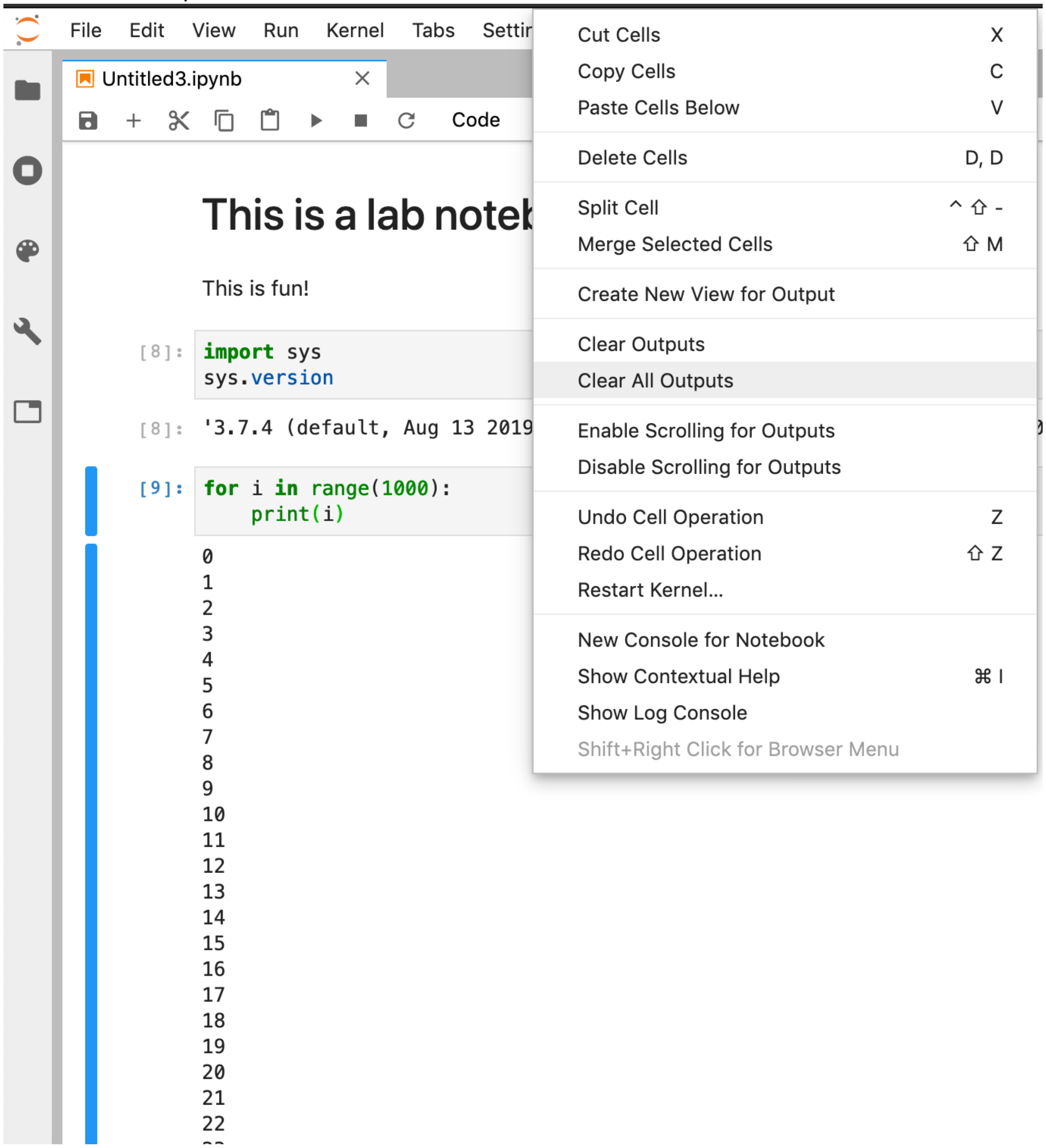
6. We’ve created a nice looking title for our Notebook, but now we want to move it to the top of our Notebook. Therefore, please hover over the area between the vertical blue bar and the title until you see a cross. Click and drag the cell to the top of the Notebook. Using the cut cell, copy cell and paste cell icons you can modify the notebook with respect to individual cells.

7. So let’s consider a cell which produces a very large quantity of output. For simulating this, please enter the following code into a new cell and run it (since this is Python, please make sure that there is a “tab” in front of the print statement):

```
for i in range(1000):


    print(i)
```

8. Since this might screw up your notebook you can easily right-click into the cell and select "Clear outputs"



This is a lab notek

This is fun!

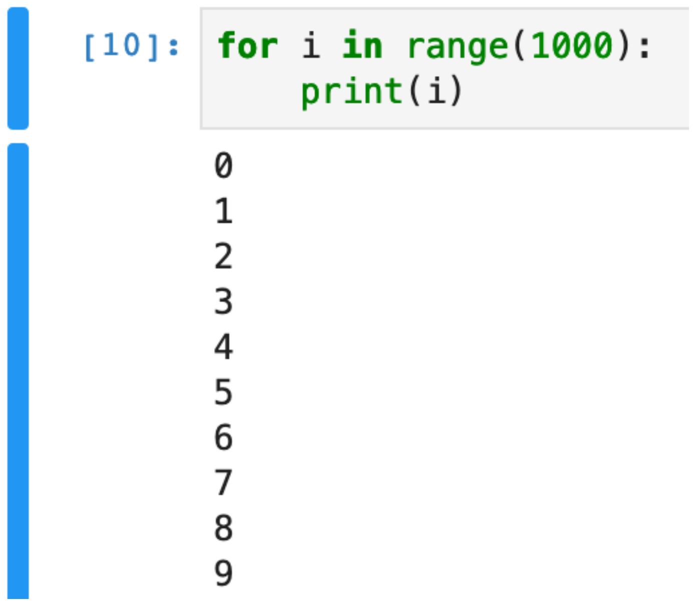
```
[8]: import sys
     sys.version

[8]: '3.7.4 (default, Aug 13 2019

[9]: for i in range(1000):
         print(i)

     0
     1
     2
     3
     4
     5
     6
     7
     8
     9
     10
     11
     12
     13
     14
     15
     16
     17
     18
     19
     20
     21
     22
```

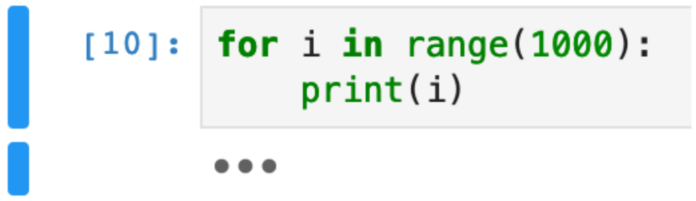
9. Alternatively you can also click on the blue bar left to the output in order to collapse it for later usage:



```
[10]: for i in range(1000):
          print(i)

      0
      1
      2
      3
      4
      5
      6
      7
      8
      9
```

10. Once collapsed, you can re-open the output by clicking on the three dots:



```
[10]: for i in range(1000):
          print(i)

      •••
```

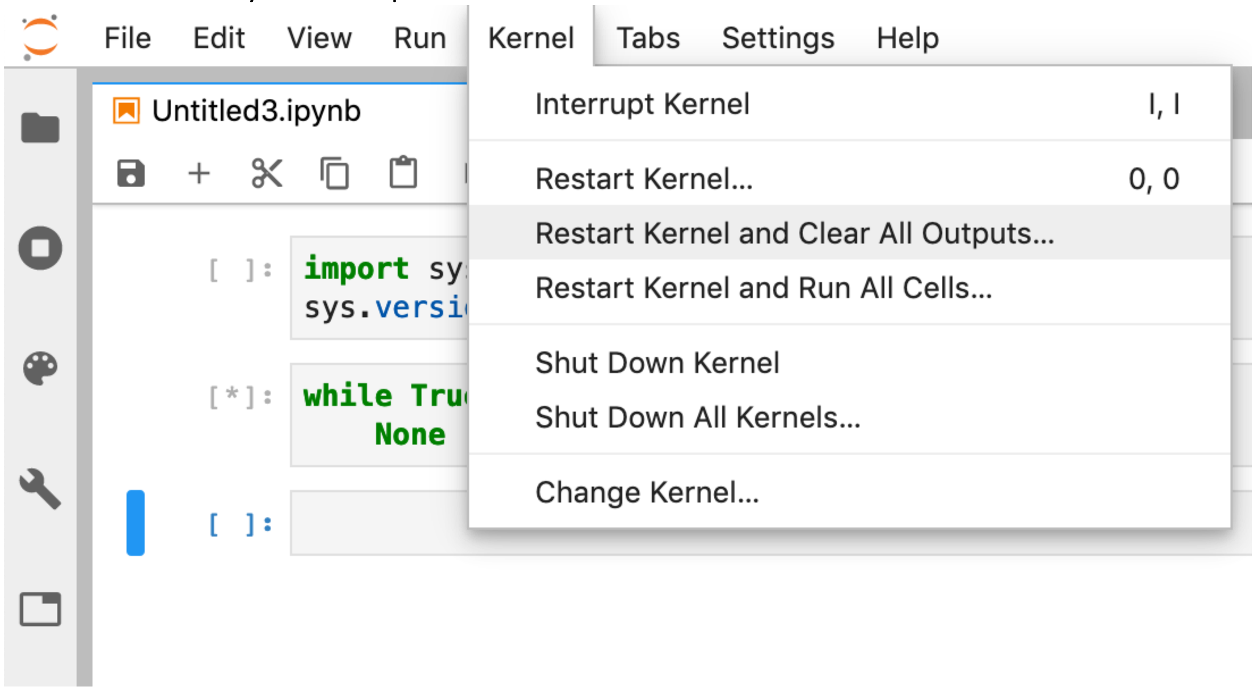
11. Sometimes it happens, that the code in you cell executes so long that you want to stop it. A task running in a cell can be identified by the "*" asterisk symbol in the brackets. To simulate this behavior, please paste the following code into a new cell and execute it (again, please don't forget the Tabulator in front of "None"):

```
while True:


              None
```

12. There are many ways of getting rid of a never ending execution, but I'm fan of the following. Click on "Kernel" and then on "Restart Kernel and Clear All Outputs". This way I'm sure that the Kernel is restarted once all output is gone. As you remember from the lecture, a Kernel is wrapping your interactive runtime environment to make it accessible to the Notebook,

in this case it is a Python interpreter.



```
[ ]: import sy
     sys.versi
[*]: while Tru
         None
[ ]:
```

13.

This concludes lab two. In the next lab we'll give you some expert tips to make you even more productive.