

# Methods of Geoinformation Science

## How to use JupyterHub

Prof. Dr. Martin Kada

---

Chair Methods of Geoinformation Science (GIS)  
Institute of Geodesy and Geoinformation Science

# Copyright Notice

---

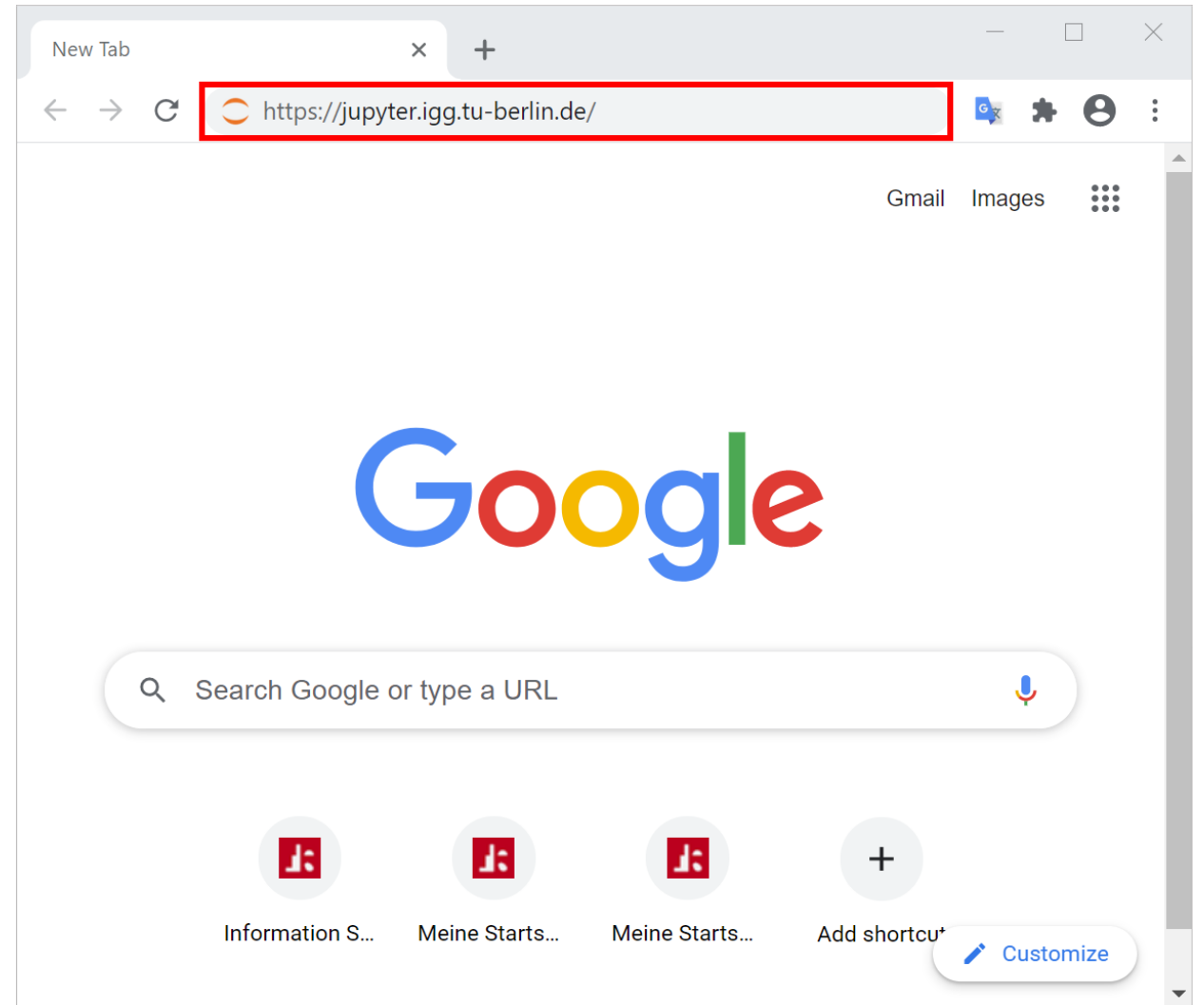
The course material provided contains material protected by international copyright laws. It may only be used for accompanying the studies for the corresponding course.

Any reproduction and redistribution of the material without written permission is prohibited other than the following: You may print or download it for your personal use only while attending the Geodesy and Geoinformation Science master's programme. You may not distribute or commercially exploit the content. You may not transmit it or store it on any other website.

# Login

To login into the technical infrastructure provided by the chair “Methods of Geoinformation Science” (GIS) for this course, you need to type the following link in your browser:

<https://jupyter.igg.tu-berlin.de/>

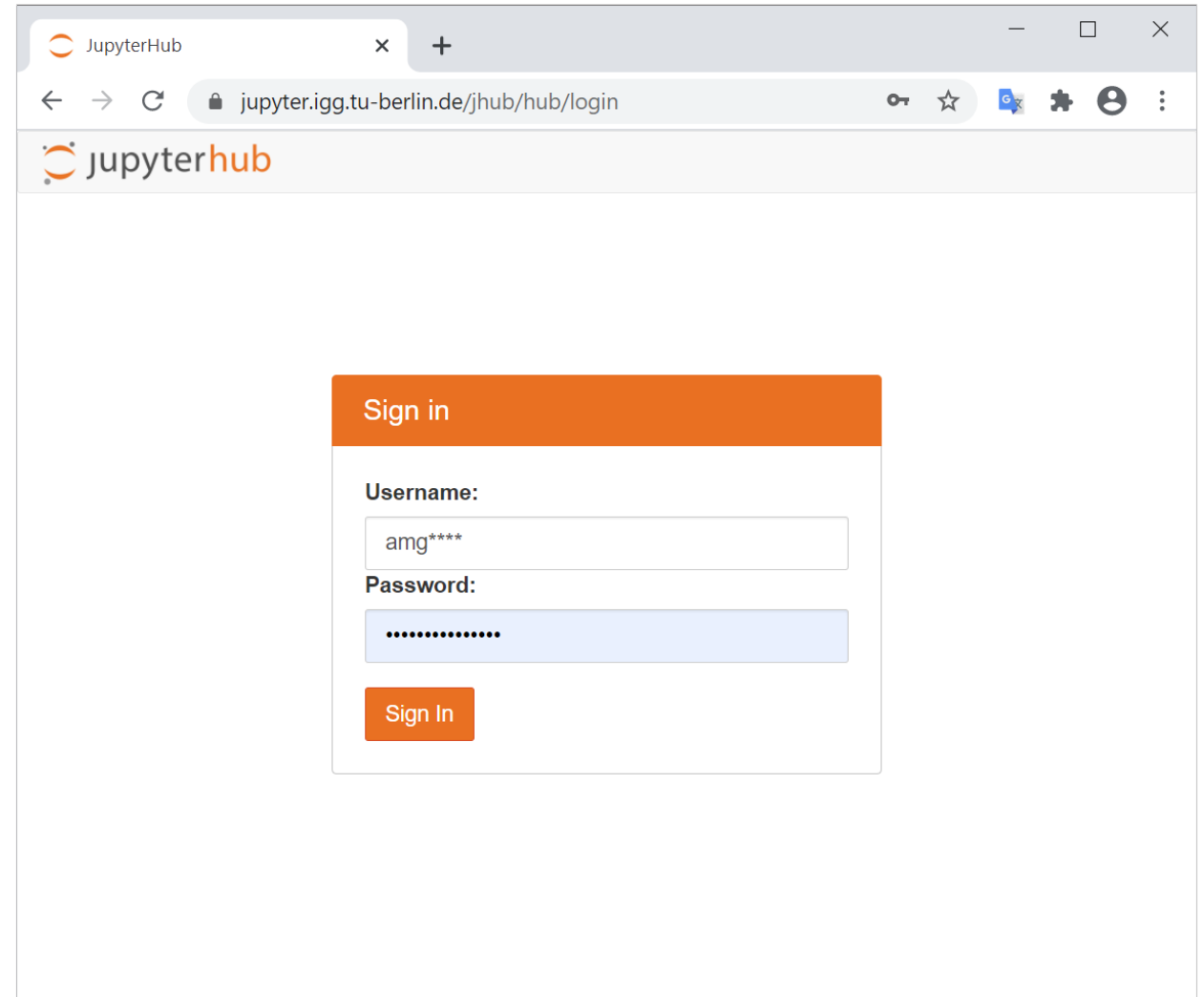


# Login

You need to sign in with your TU Berlin user account with your **Tubit username** and **password**.

We use the official LDAP authentication of Tubit and neither see nor store your password in any way.

JupyterHub is used to run our services; for more details, please refer to the [documentation](#).



The screenshot shows a web browser window with the JupyterHub login page. The browser's address bar shows the URL `jupyter.igg.tu-berlin.de/jhub/hub/login`. The page features a "Sign in" form with an orange header. The form contains two input fields: "Username:" with the text "amg\*\*\*\*" and "Password:" with masked characters ".....". Below the fields is an orange "Sign In" button.

# Login

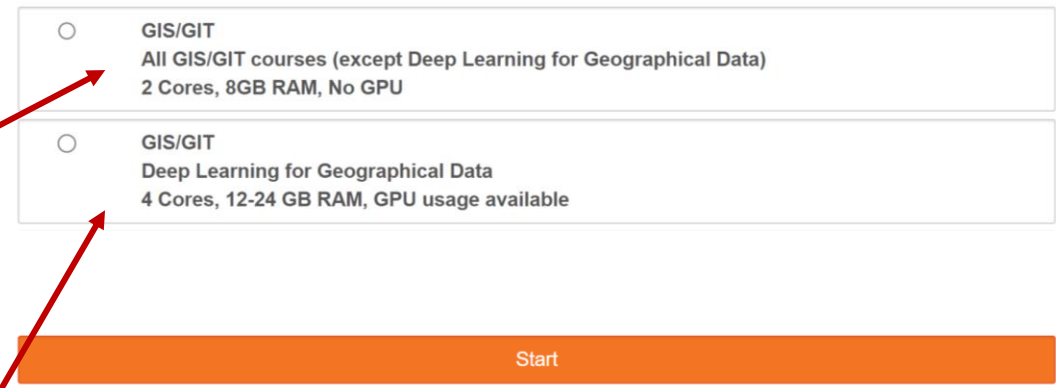
Once you logged in, you will see the server options we provide.

Please note that you will only be able to use the server option for the courses you registered in.

The general server option is "All GIS/GIT courses ...".

The other server option is only accessible to students registered in the course "Deep Learning for Geographical Data" and offers high performance GPUs.

## Server Options



☐ GIS/GIT  
All GIS/GIT courses (except Deep Learning for Geographical Data)  
2 Cores, 8GB RAM, No GPU

☐ GIS/GIT  
Deep Learning for Geographical Data  
4 Cores, 12-24 GB RAM, GPU usage available

Start

# Access to JupyterHub

---

- Send an email with the topic “exercise server” to the following email address asking for access to JupyterHub providing us your **family name, first name, matriculation number, TU Berlin email address, possibly TU Berlin user name, and the course(s) you want to use the server for**

[andreas.fuls@tu-berlin.de](mailto:andreas.fuls@tu-berlin.de)

- **Please do not send us your password!!!**
- Access will be granted as soon as possible (within office hours), but it may from time to time also take a few days

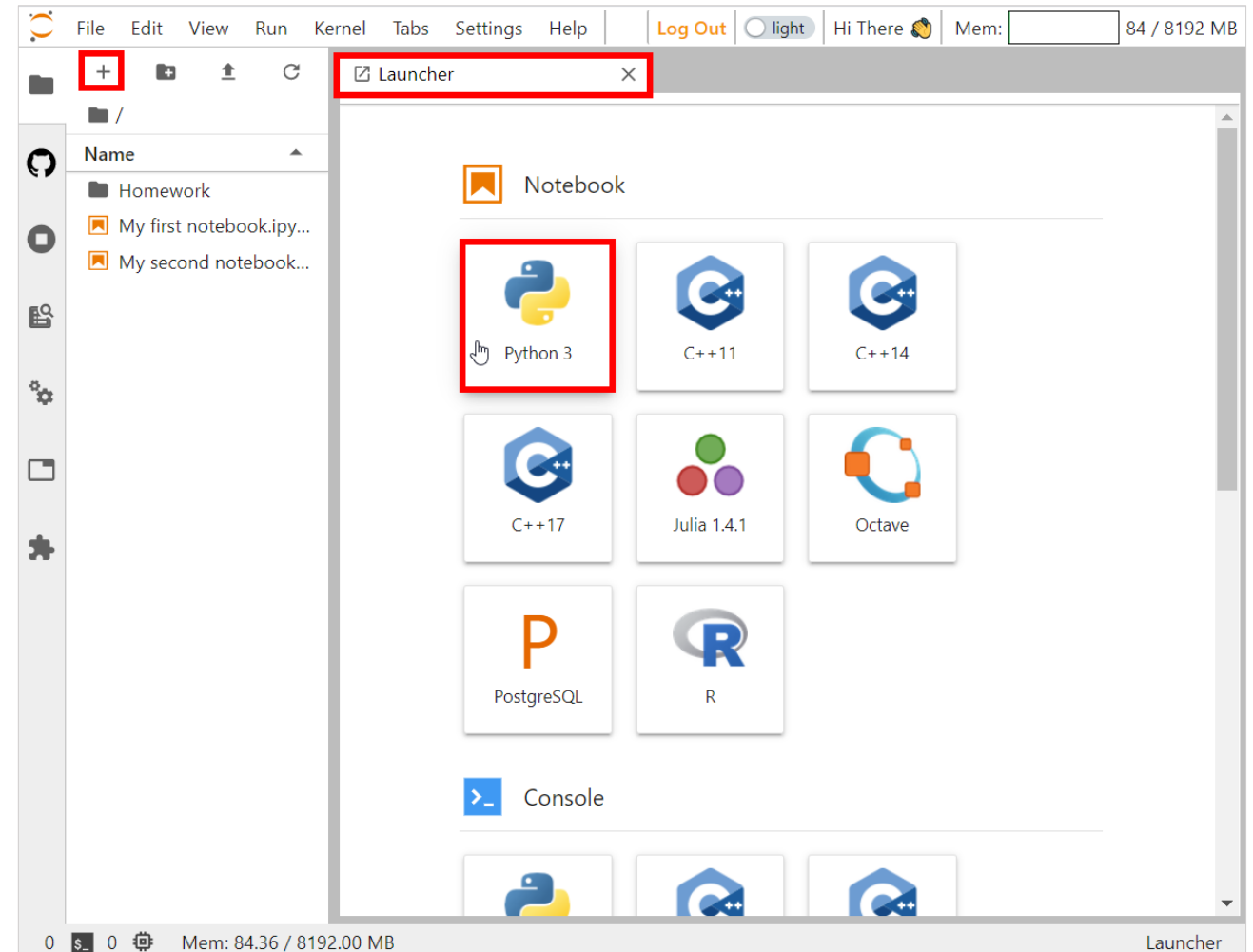
# New Launcher

You have a variety of so called kernels to choose from that provide different programming languages.

With the Launcher window, you can specify a programming language to work with and create a new Jupyter notebook.

If you cannot see this window then press the **New Launcher** button (the one with the + icon).

**And check the next slide!!!**

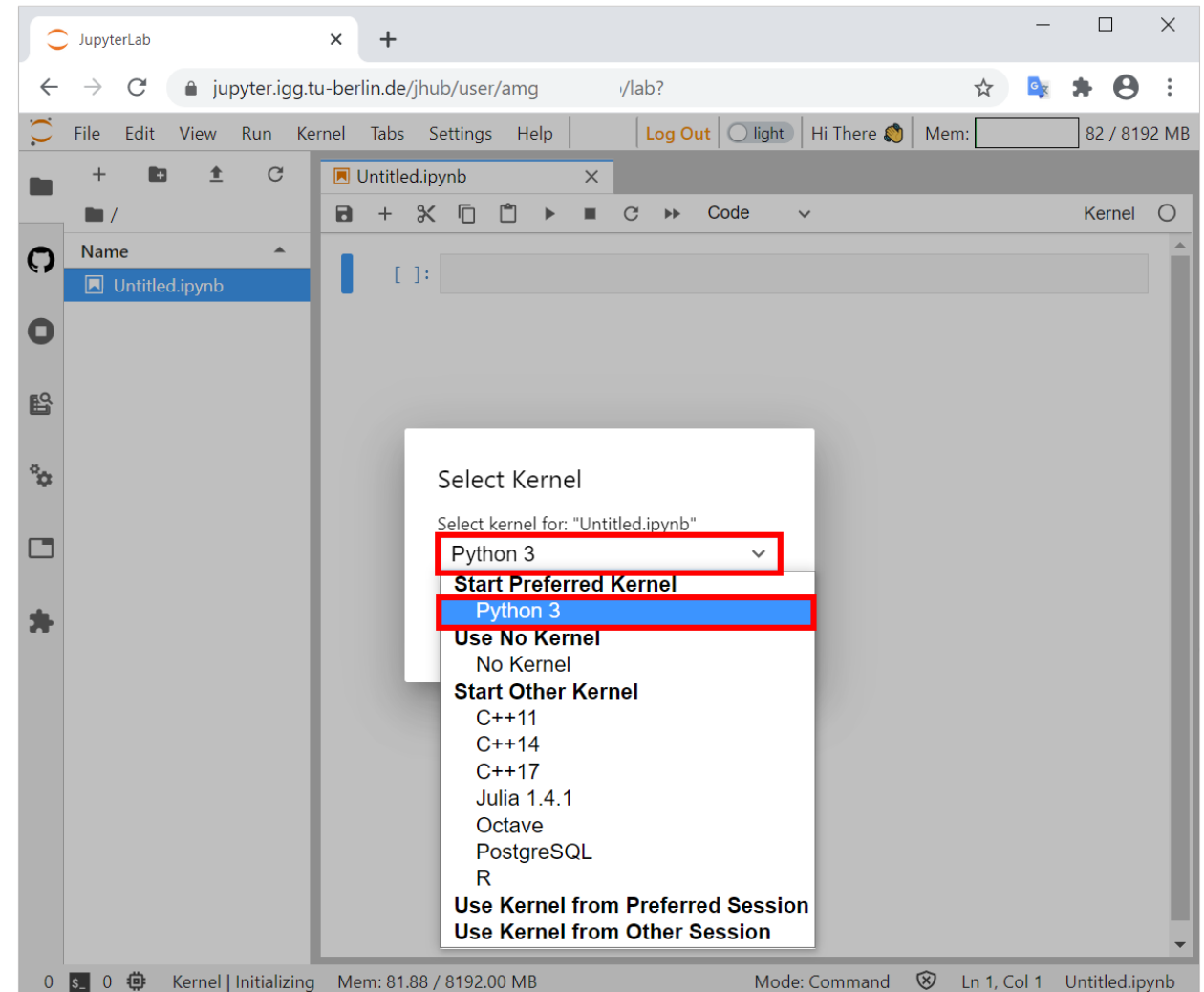


# Kernel

When you log in for the first time (after selecting the desired server option) you will be greeted with a new notebook and asked to select a kernel.

Select the default kernel, which is the Python 3 kernel.

**Almost all courses given by the chair GIS will use Python.**



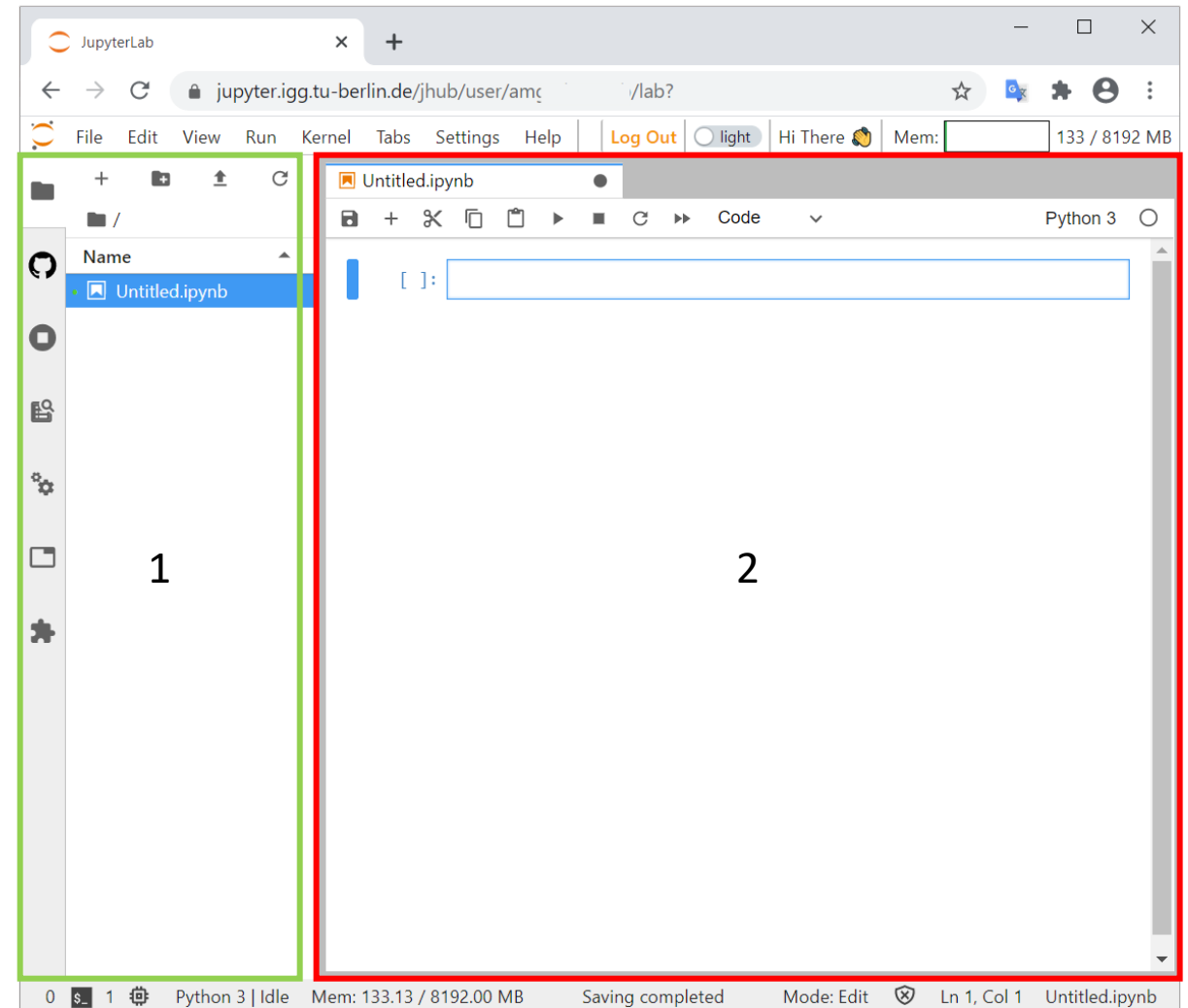


# JupyterLab

You should now see the JupyterLab working environment.

Take your time to explore all the available options for you.

The left sidebar (area 1) contains a number of commonly-used tabs (the icons on the left side), such as a file browser, a list of running kernels, the command palette, and a list of open tabs in the main working area.

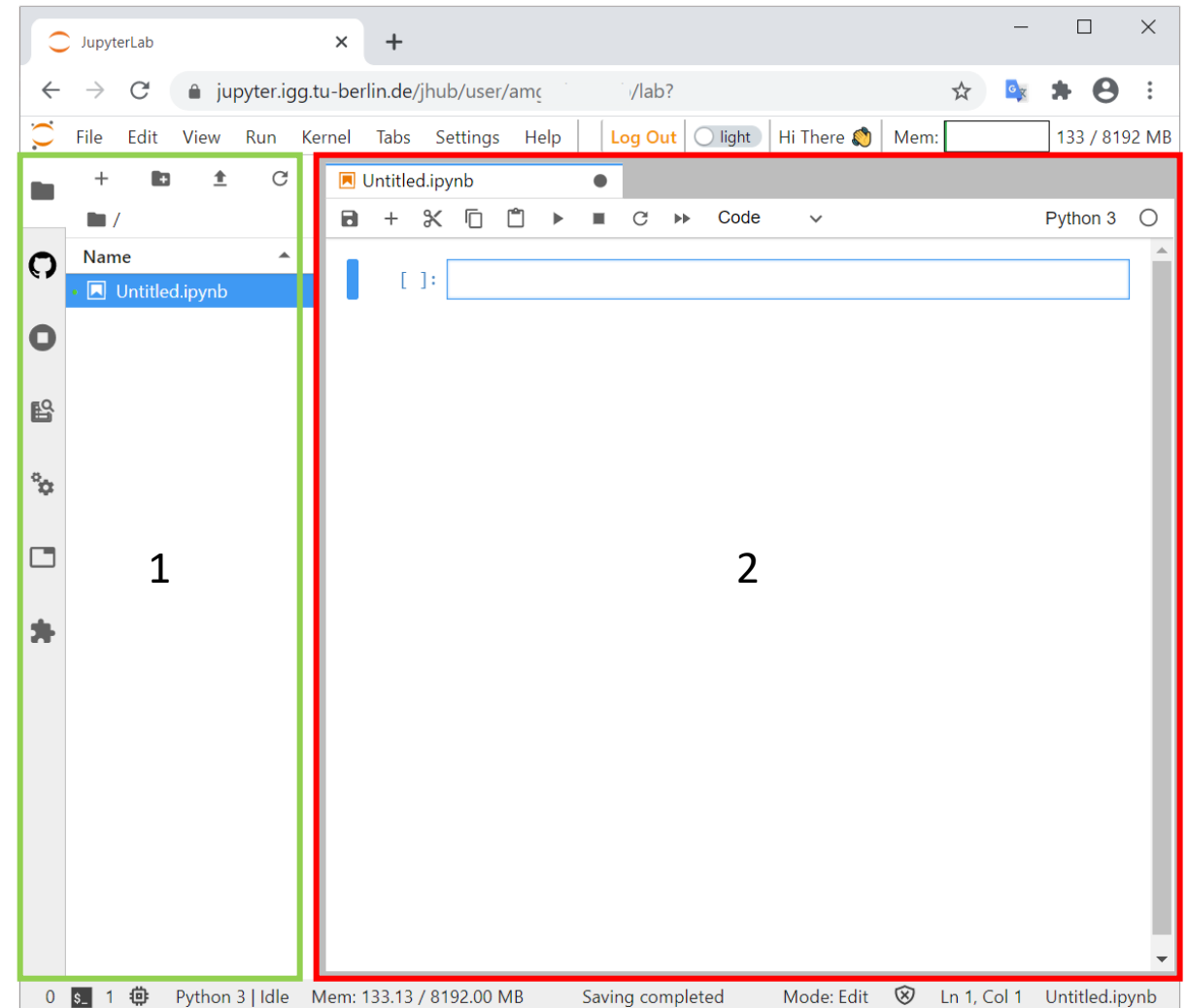


# JupyterLab

The main working area (area 2) in JupyterLab allows you to arrange notebooks, text files, and others (terminals, code consoles, etc.).

Currently, you should see one open Jupyter notebook in this area.

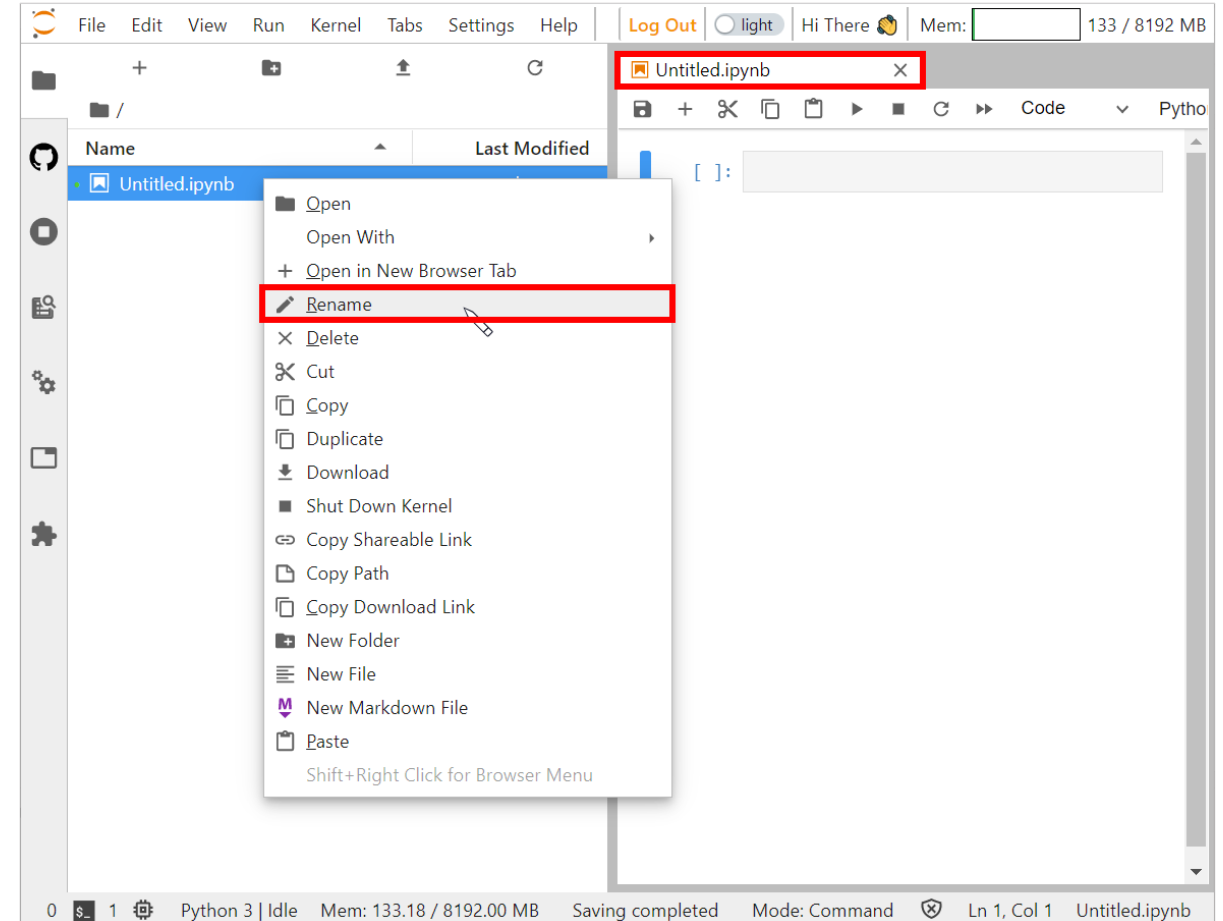
We cannot cover all the details here, so please have a look into the JupyterLab [documentations](#).



# Jupyter Notebook

Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.

Let's first rename our first Jupyter notebook to a more meaningful name, e.g. "My first notebook".

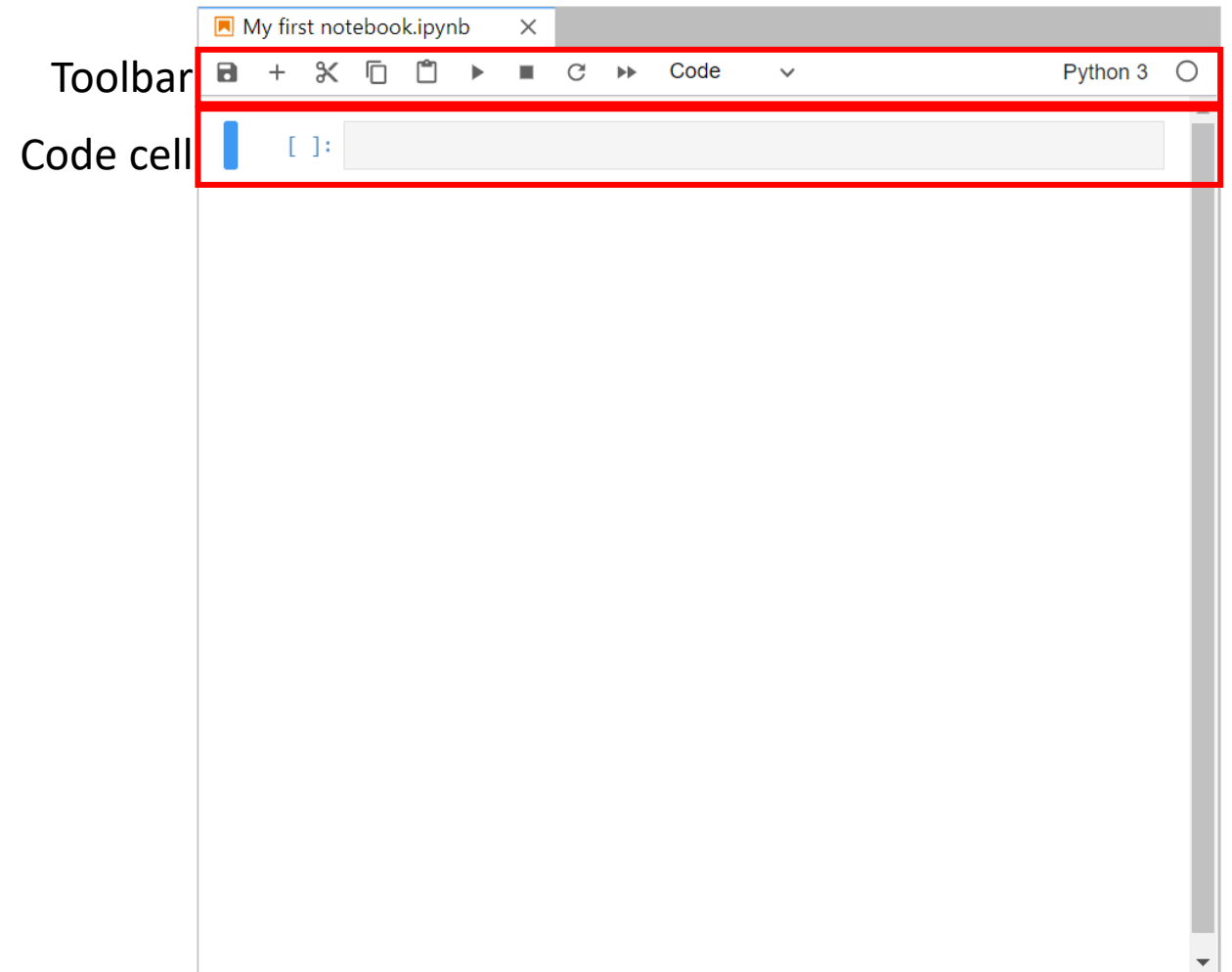


# Jupyter Notebook

Notice that you are provided with multiple tools to work with by default.

The toolbar gives a quick way of performing the most-used operations within the notebook.

When a notebook is created, there is one empty code cell, which we will use to write our first piece of Python code.

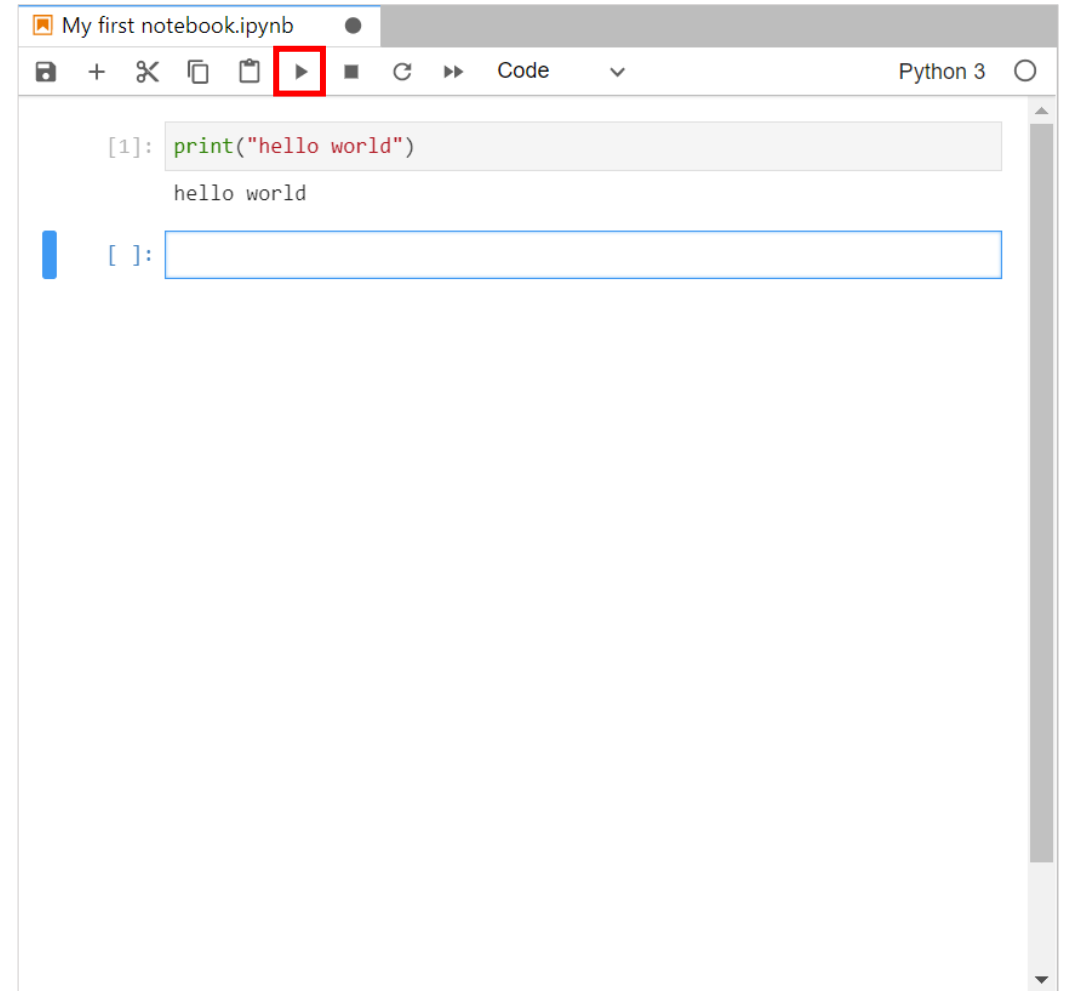


# Code Cells

A code cell allows you to write and edit code with full syntax highlighting and tab completion.

You can type the code in the image, which will print the words “hello world” in the notebook.

After typing the code press the **Run** icon which will execute the active cell and insert a new empty cell in the notebook. Notice that the empty cell is now active.

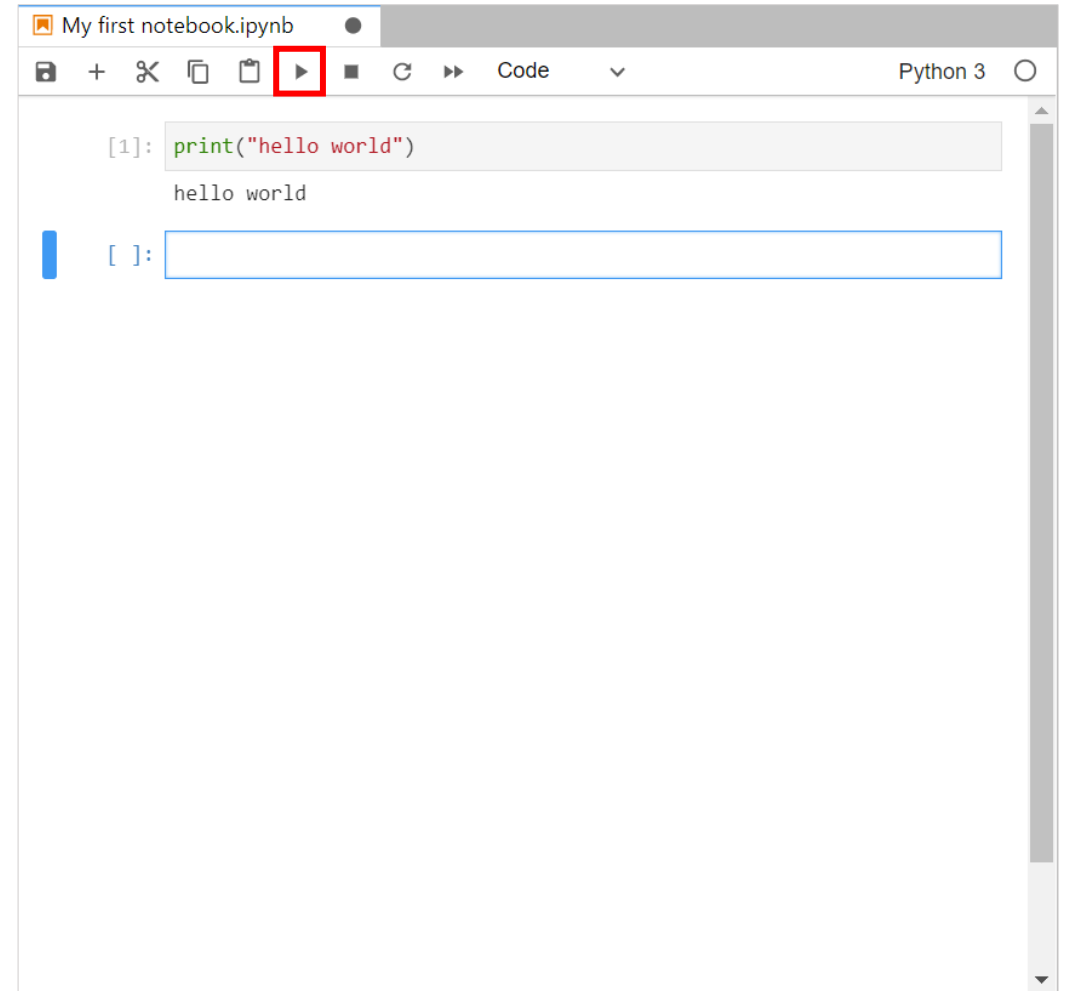


# Code Cells

You can also use the keyboard shortcut **Ctrl+Enter (Return)** to execute the cell.

While **Alt+Enter (Return)** executes the active cell and inserts a new cell below. (Please see the keyboard slide for more information).

Note that there is a change in style between Python code and plain text (the cell result).



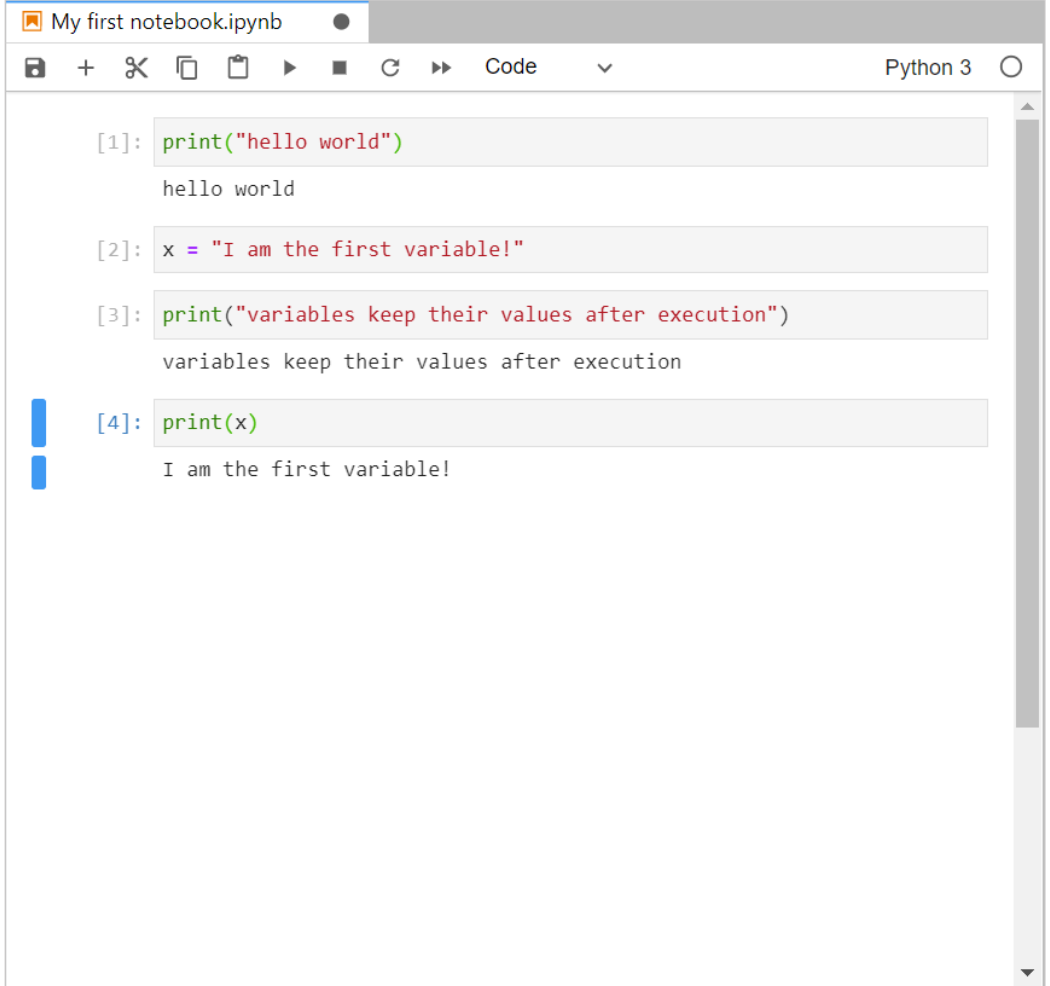
# My First Variable

Let us declare our first variable **x**.

The value of **x** is assigned the string „I am the first variable“.

Note that the print function is called in a separate cell.

The code in the cells act like a computer program and you need to execute them in an order that makes sense and follows the logic of the programming language.



```
[1]: print("hello world")
hello world

[2]: x = "I am the first variable!"

[3]: print("variables keep their values after execution")
variables keep their values after execution

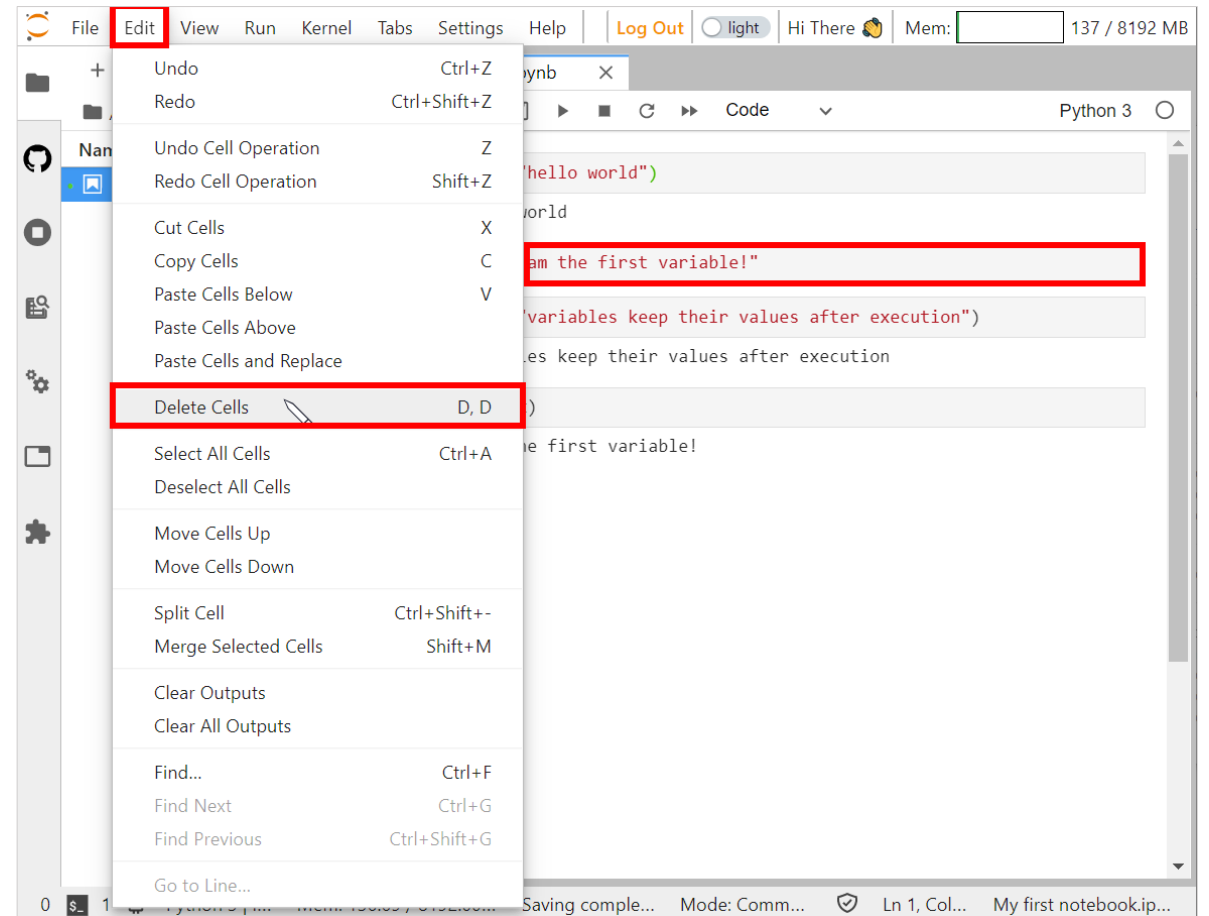
[4]: print(x)
I am the first variable!
```

# Deleting a Cell

Let us delete the second cell where the variable **x** is declared.

You can delete a cell as shown in the screenshot.

An alternative is to activate the cell (click into it), then press the **Esc** button and hit the button **D** twice on the keyboard (see the keyboard shortcuts slide for more details).





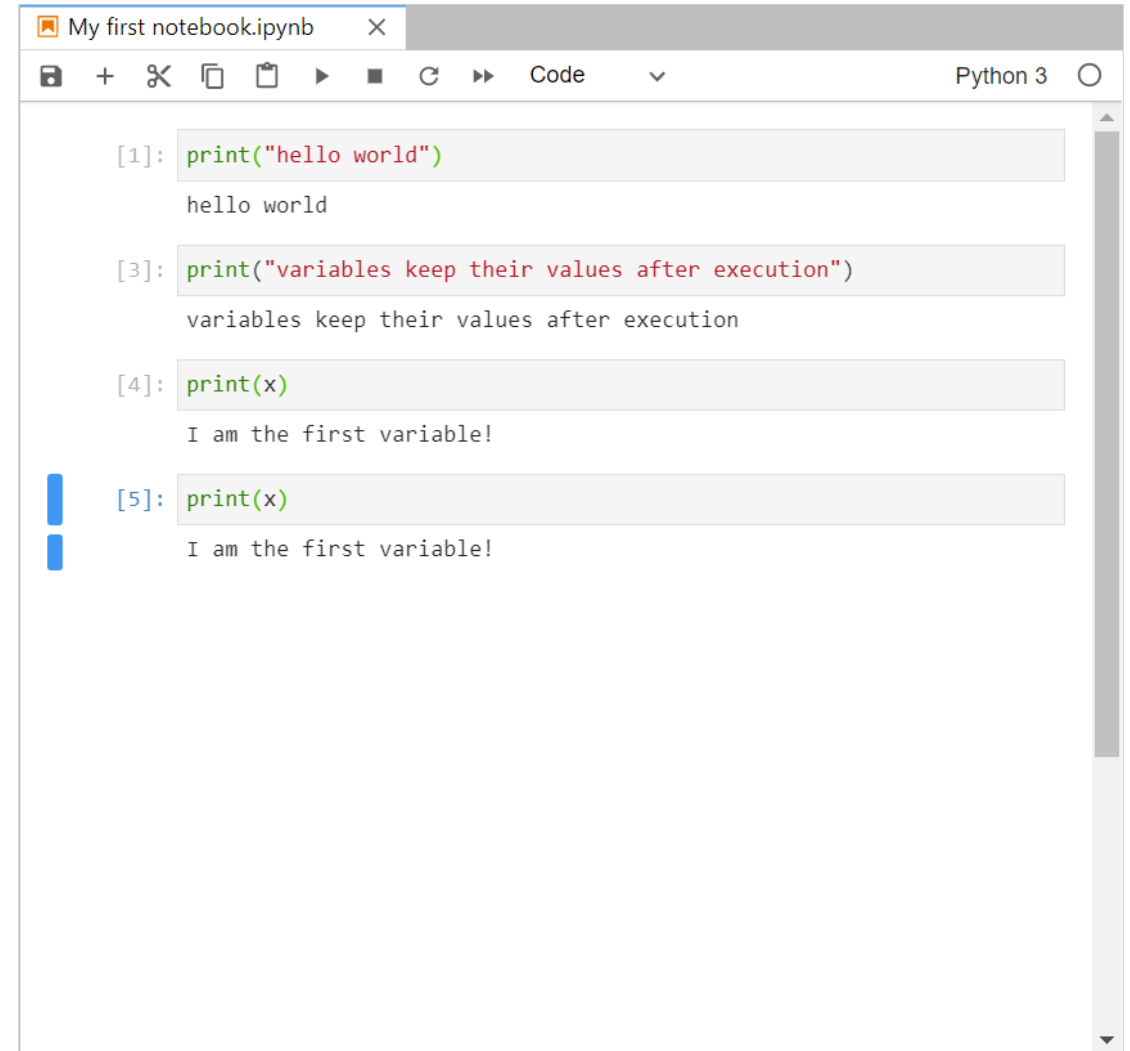
# Deleting a Cell

Add a new cell and try to print the variable `x`. We notice that even though we deleted the cell, the variable value is still stored.

Python keeps a state about the code you already executed.

Deleting a cell does not undo the execution of this cell.

Keep this in mind as this can quickly become a source of errors if forgotten.



```
[1]: print("hello world")
hello world

[3]: print("variables keep their values after execution")
variables keep their values after execution

[4]: print(x)
I am the first variable!

[5]: print(x)
I am the first variable!
```

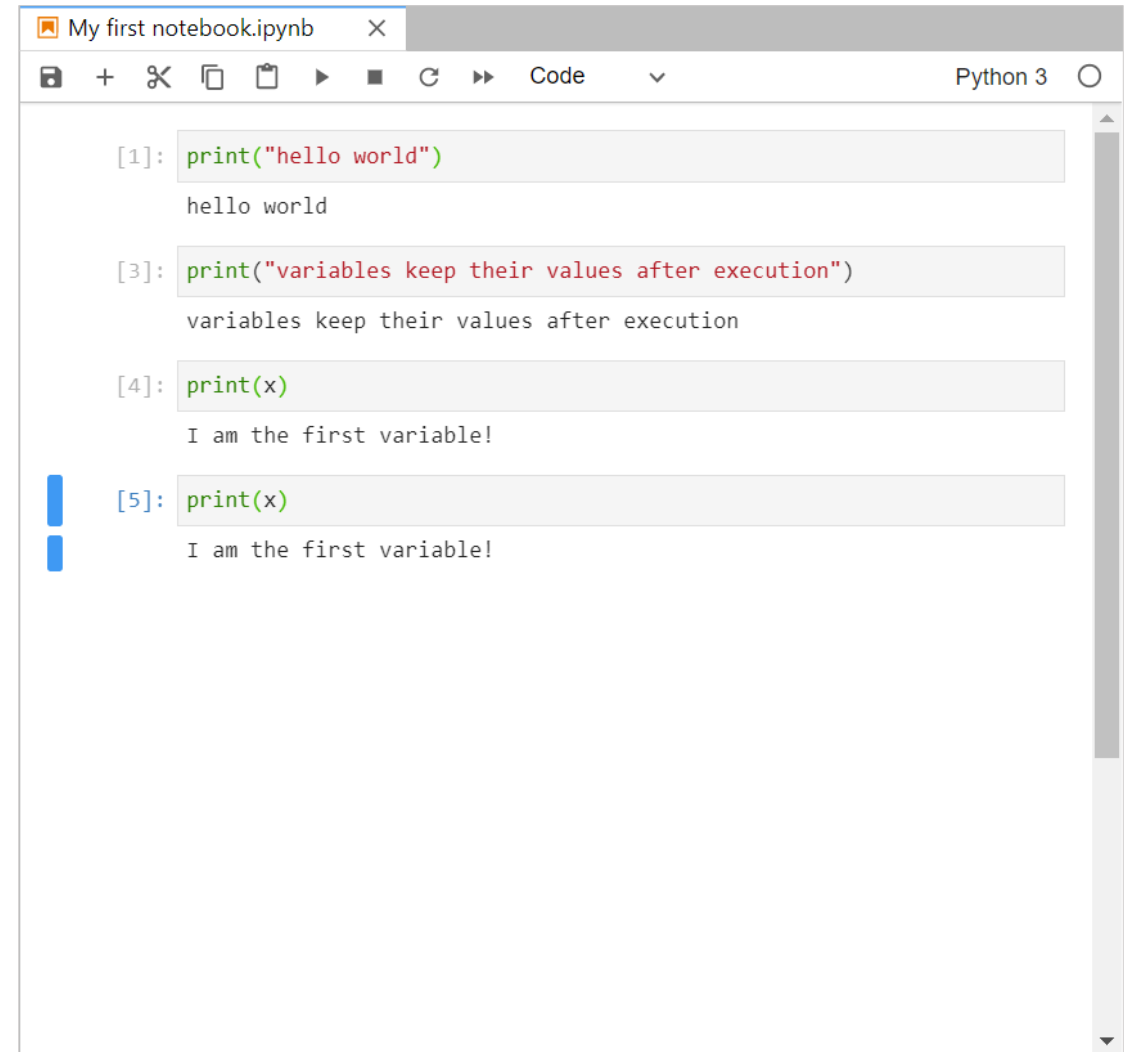
# Deleting a Cell

The numbering of the cells gives the order in which you executed cells.

And you can also execute cells many times.

But keep in mind that the order matters in which you execute them.

For example, if you execute cell 1 in the right image once more, it will be as if it was executed after cell 5 (and it will also be given the number 6). Try it out!



```
[1]: print("hello world")
hello world

[3]: print("variables keep their values after execution")
variables keep their values after execution

[4]: print(x)
I am the first variable!

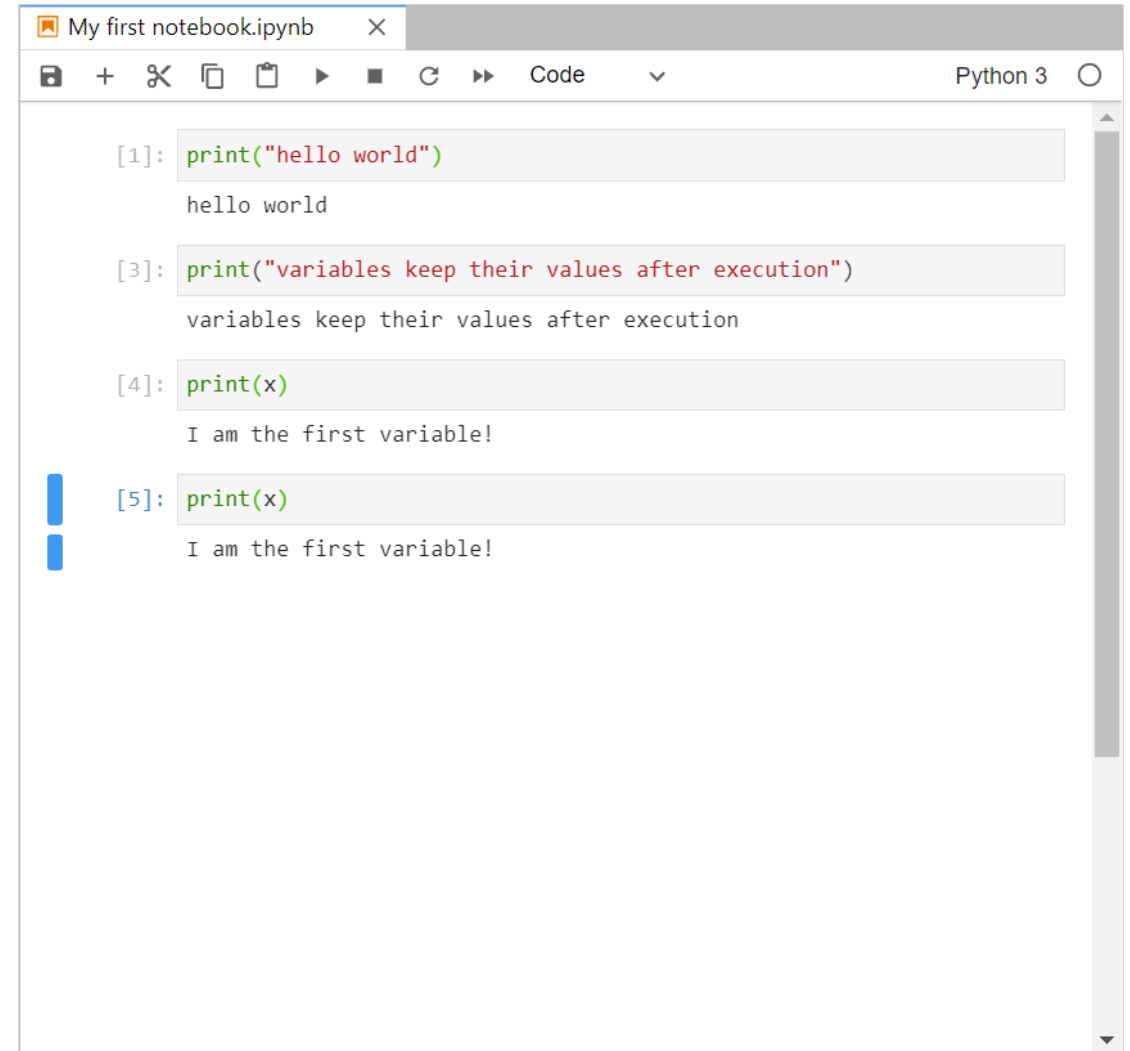
[5]: print(x)
I am the first variable!
```

# Deleting a Cell

Because of the deletion of the cell, the number 2 is now missing.

This is not a problem, but keep in mind that even though you deleted that cell, it was still executed.

If your cell order gets too messy, it might be useful to restart the kernel and clear the output of the cells (see next slide for details).



```
[1]: print("hello world")
hello world

[3]: print("variables keep their values after execution")
variables keep their values after execution

[4]: print(x)
I am the first variable!

[5]: print(x)
I am the first variable!
```

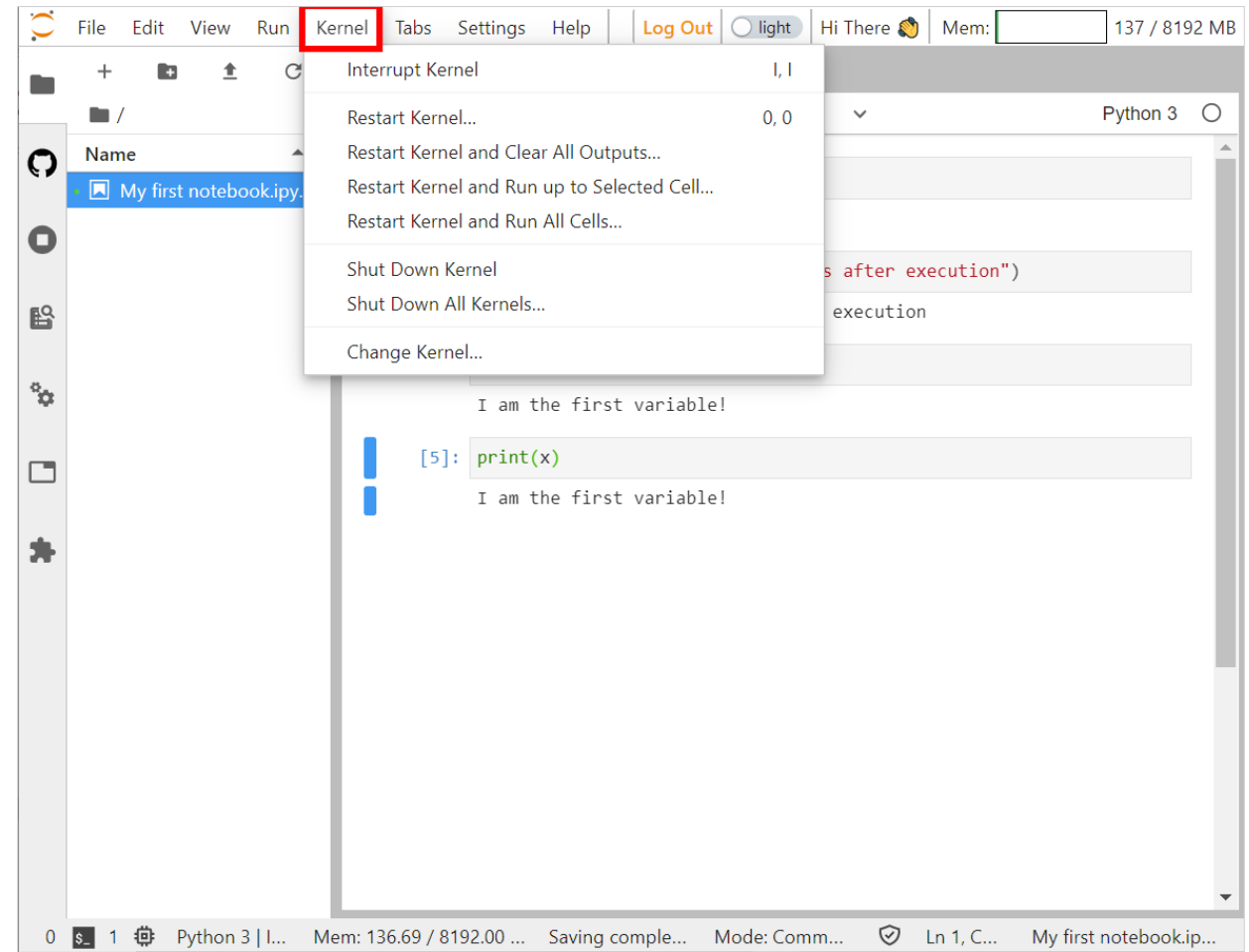
# Restarting a Kernel

You have several options when restarting the kernel as shown in the screenshot.

“Restart Kernel and Clear All Outputs...” is a good choice as it resets the kernel and deletes all output. It is like having a fresh start for a notebook.

Now you should get an error if you tried to execute the cell that contains the code `print(x)`:

```
NameError Traceback (most recent call last) <ipython-  
input-3-fc17d851ef81> in <module> ----> 1 print(x)  
NameError: name 'x' is not defined
```

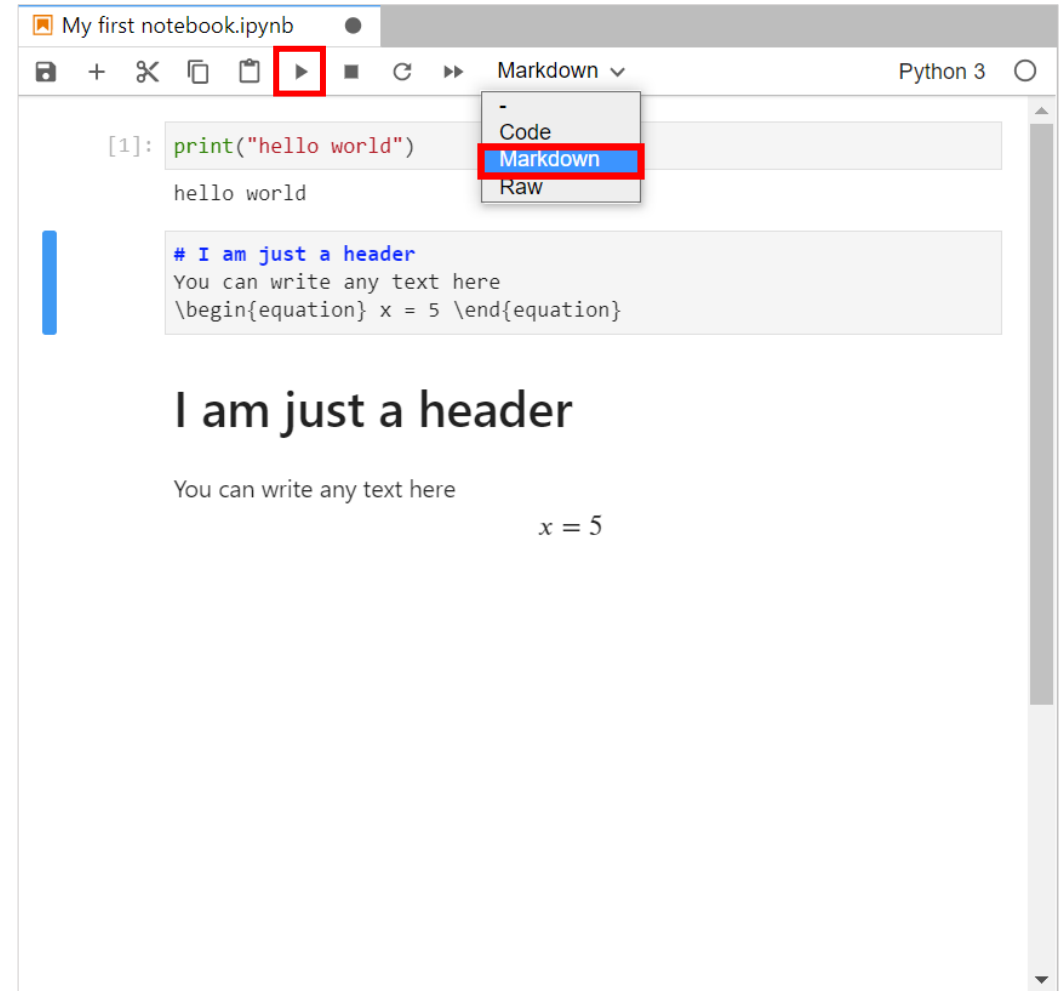


# Markdown Cells

Markdown cells display text that can be formatted using markdown language.

This is very useful to add your notes and impressions to the exercises or tasks that you are working on.

After changing the type of the cell to Markdown you will be able to use a variety of text styles (e.g. header, plain text, mathematical equation, etc.)

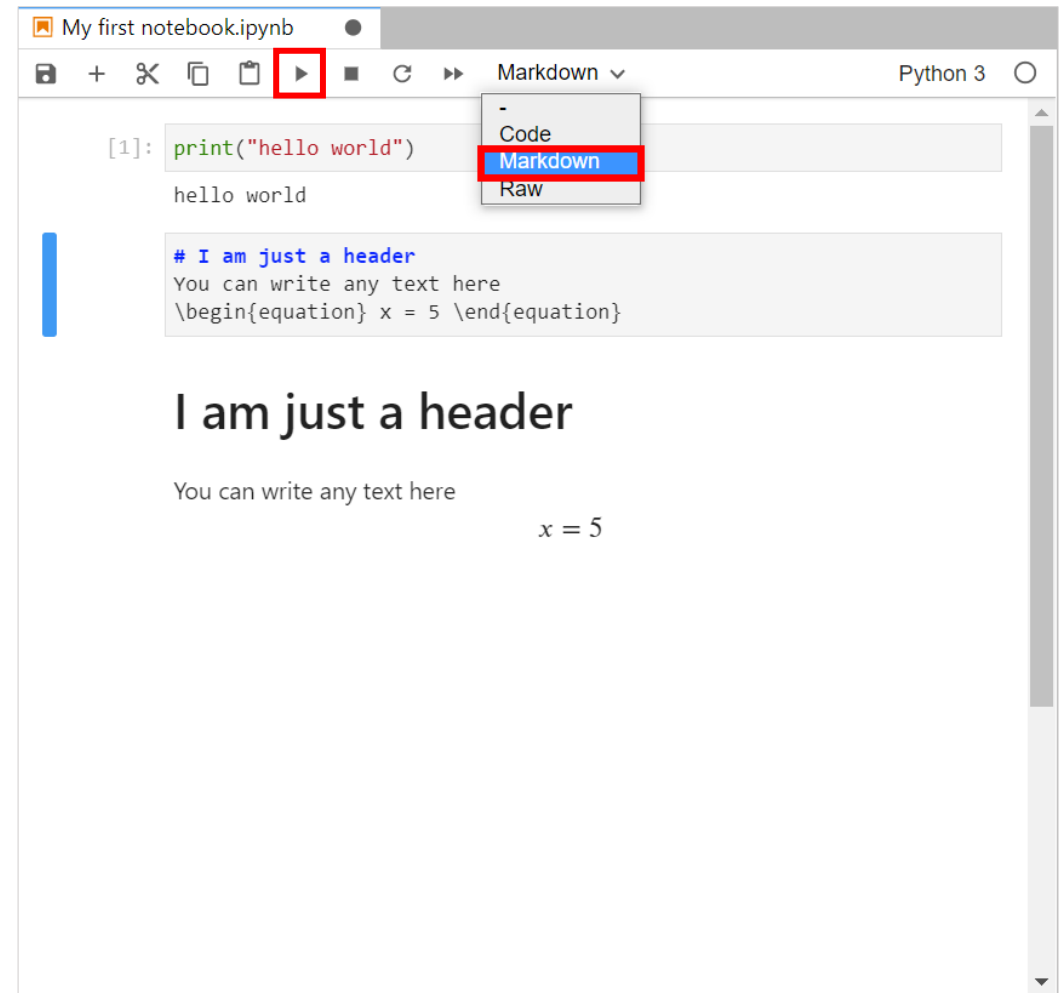


# Markdown Cells

Markdown cells must also be executed like code cells, so that they show their formatting.

But be aware that when you execute a markdown cell, the cell that is created next is also a markdown cell.

Note that a markdown cell does not have the squared brackets on the left side as a code cell.

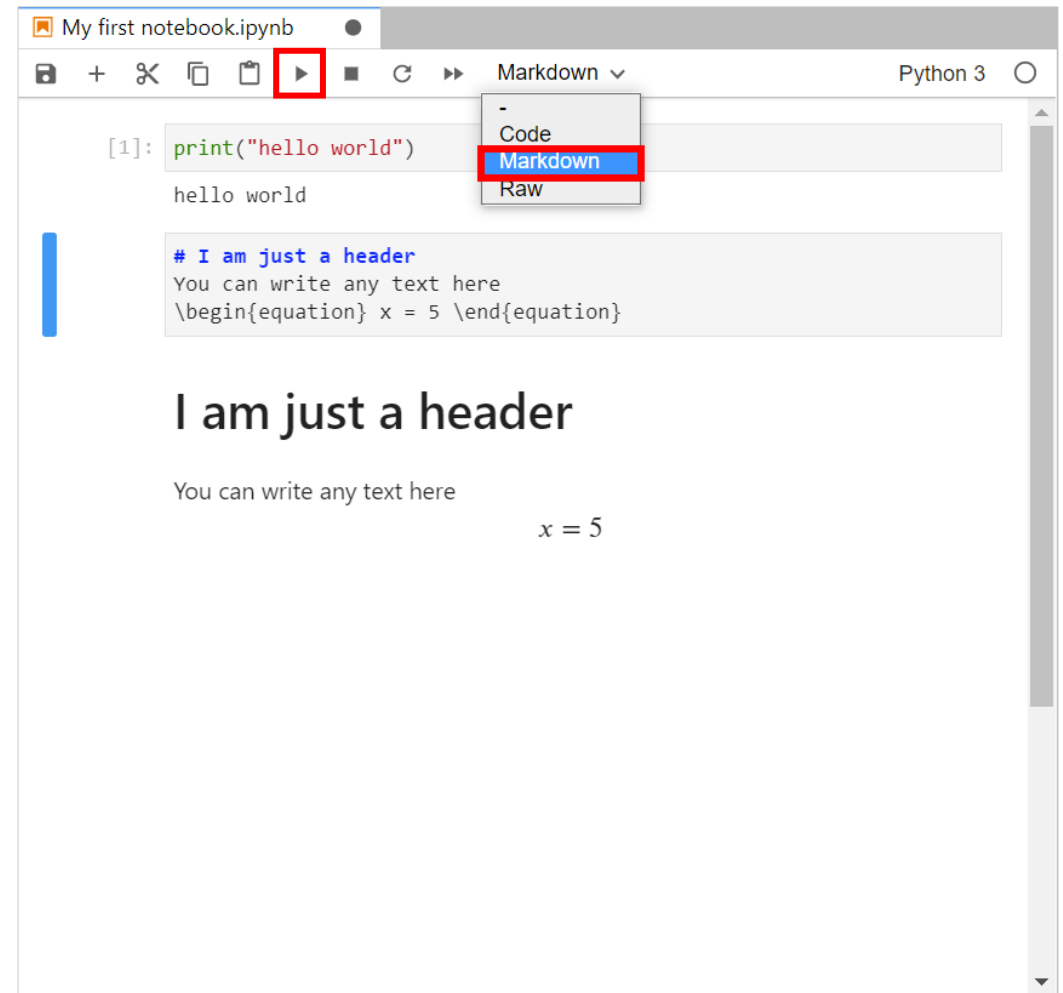


# Markdown Cells

## A common error:

If you get weird errors when executing a cell, then check that the cell is of the type you expected.

And change it to the one you wanted.  
(No need to create a new one.)



# Keyboard Shortcuts

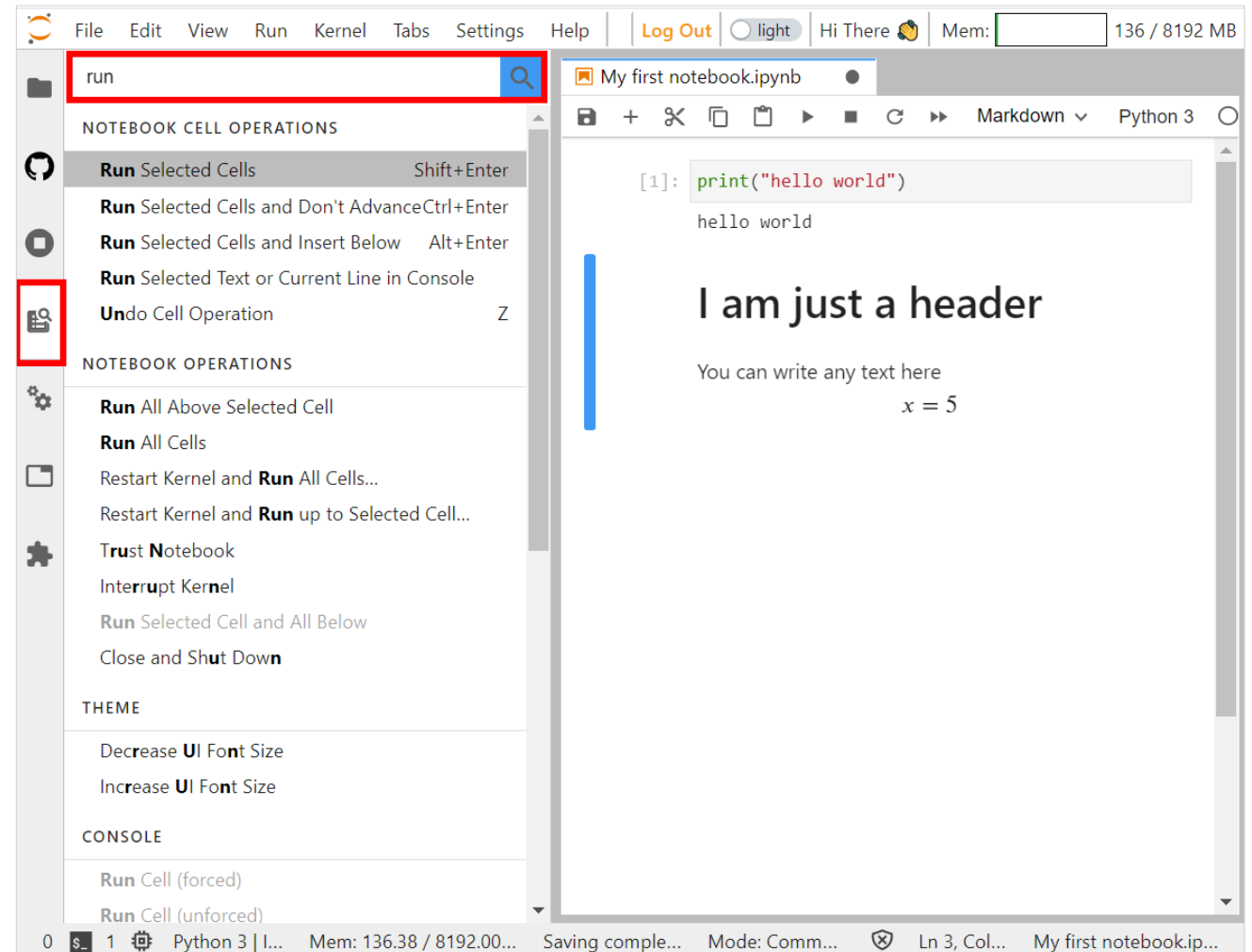
There are many keyboard shortcuts like:

**Shift-Enter**: run cell

**Esc**: Command mode

**Enter**: Edit mode

You can search for a specific command by using the search tab in JupyterLab



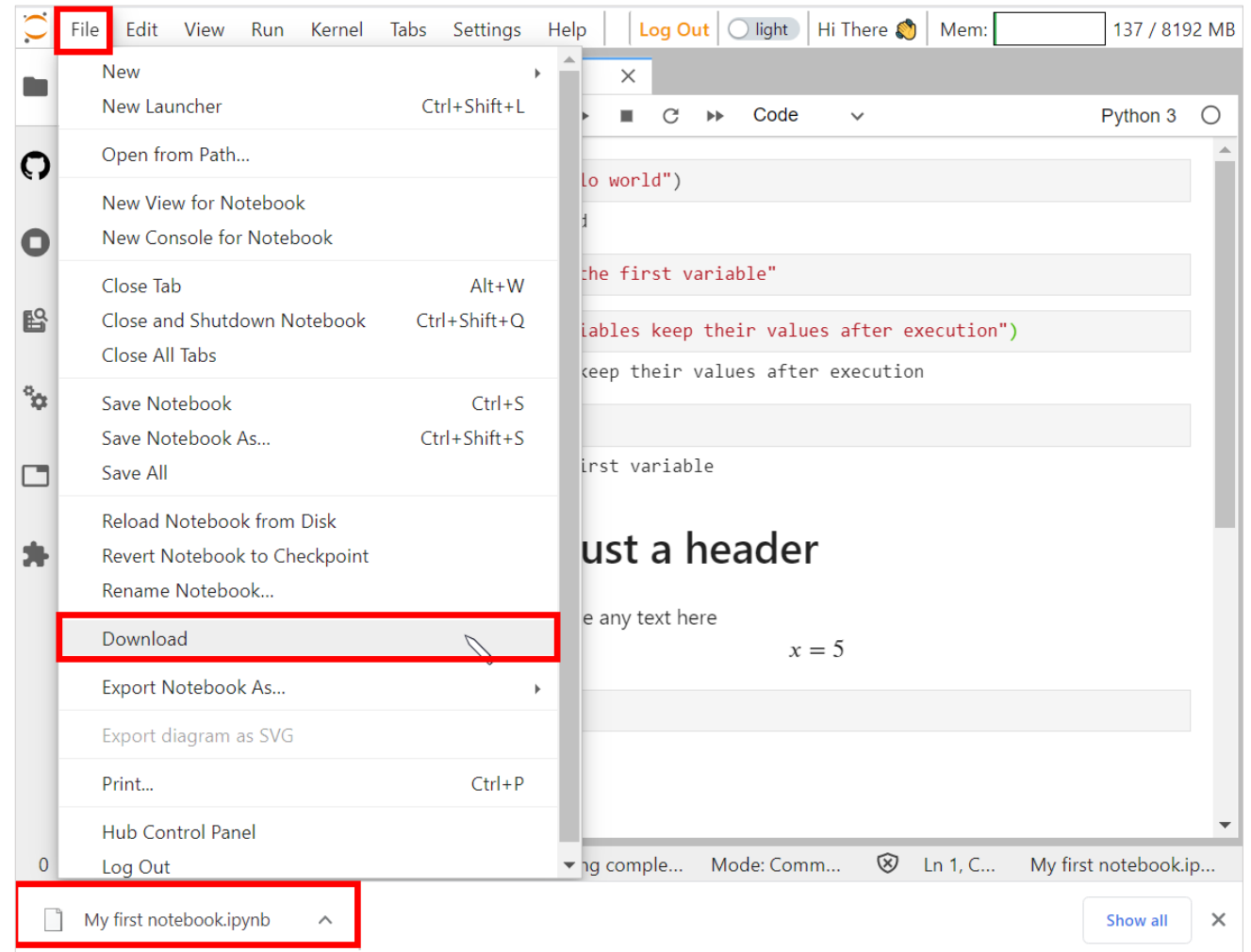


# Downloading the Notebook

You can download your active notebook from the server as seen in the image.

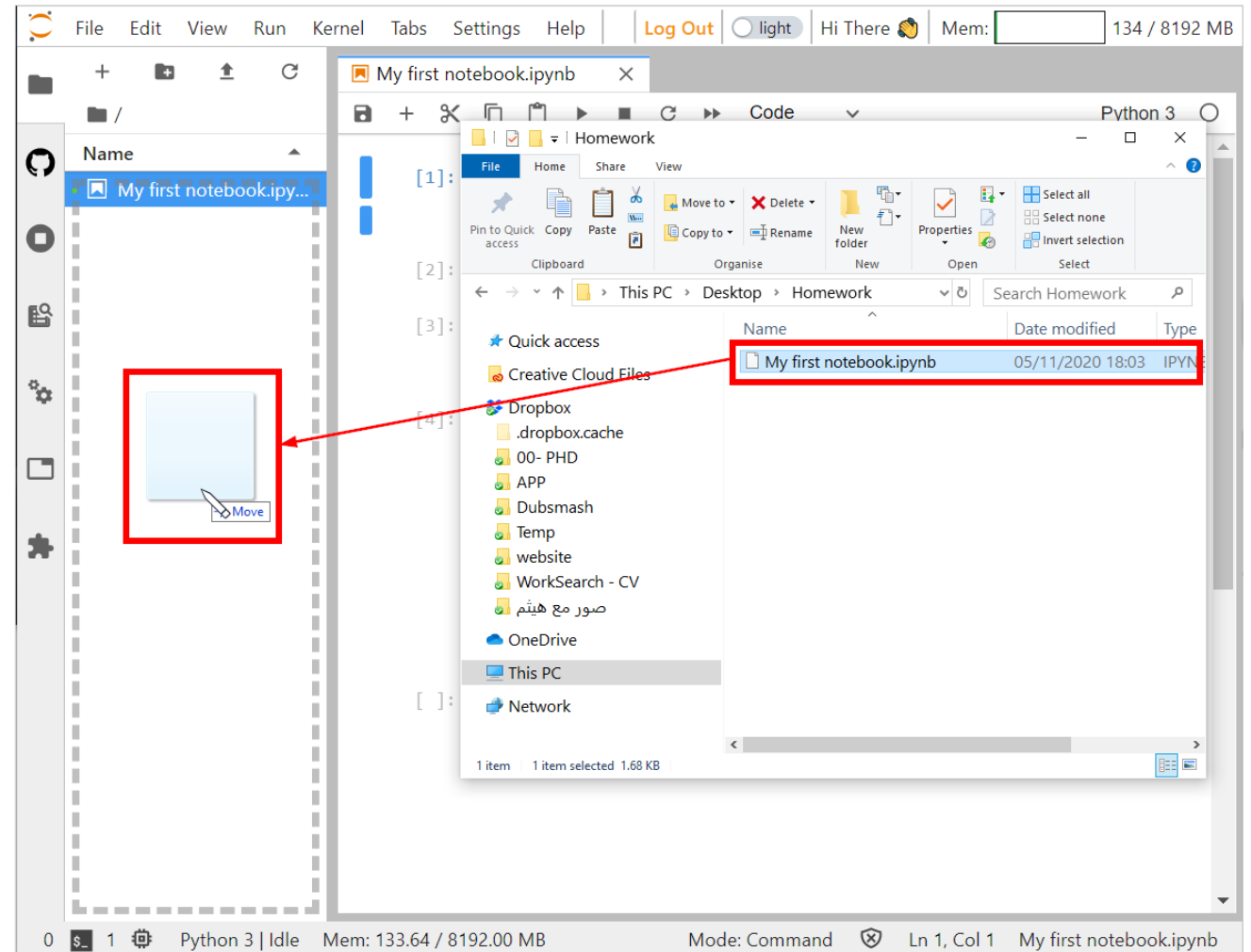
Or you can use the file browser of JupyterHub, right click on the file, and choose Download.

Jupyter notebooks have the file extension **.ipynb**, which is rather well-known nowadays.



# Uploading a Notebook

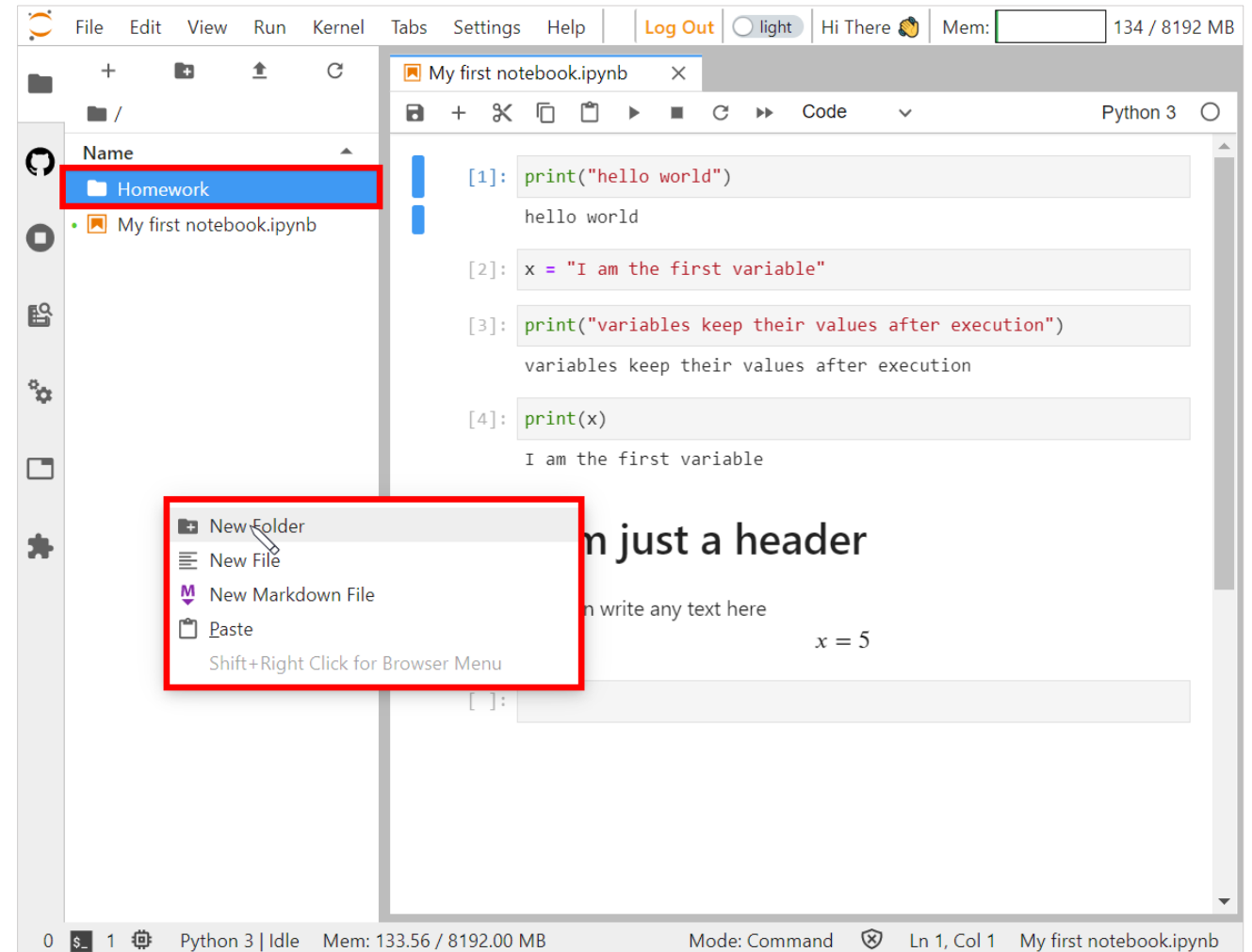
Uploading is even easier: just drag and drop a file from your local computer into the file browser area of JupyterLab



# Managing your Folder

Like with any other directory, we recommend you create a meaningful structure of files and folders to help you keep track of all your notebooks.

Notebooks are by the way shared among server options.



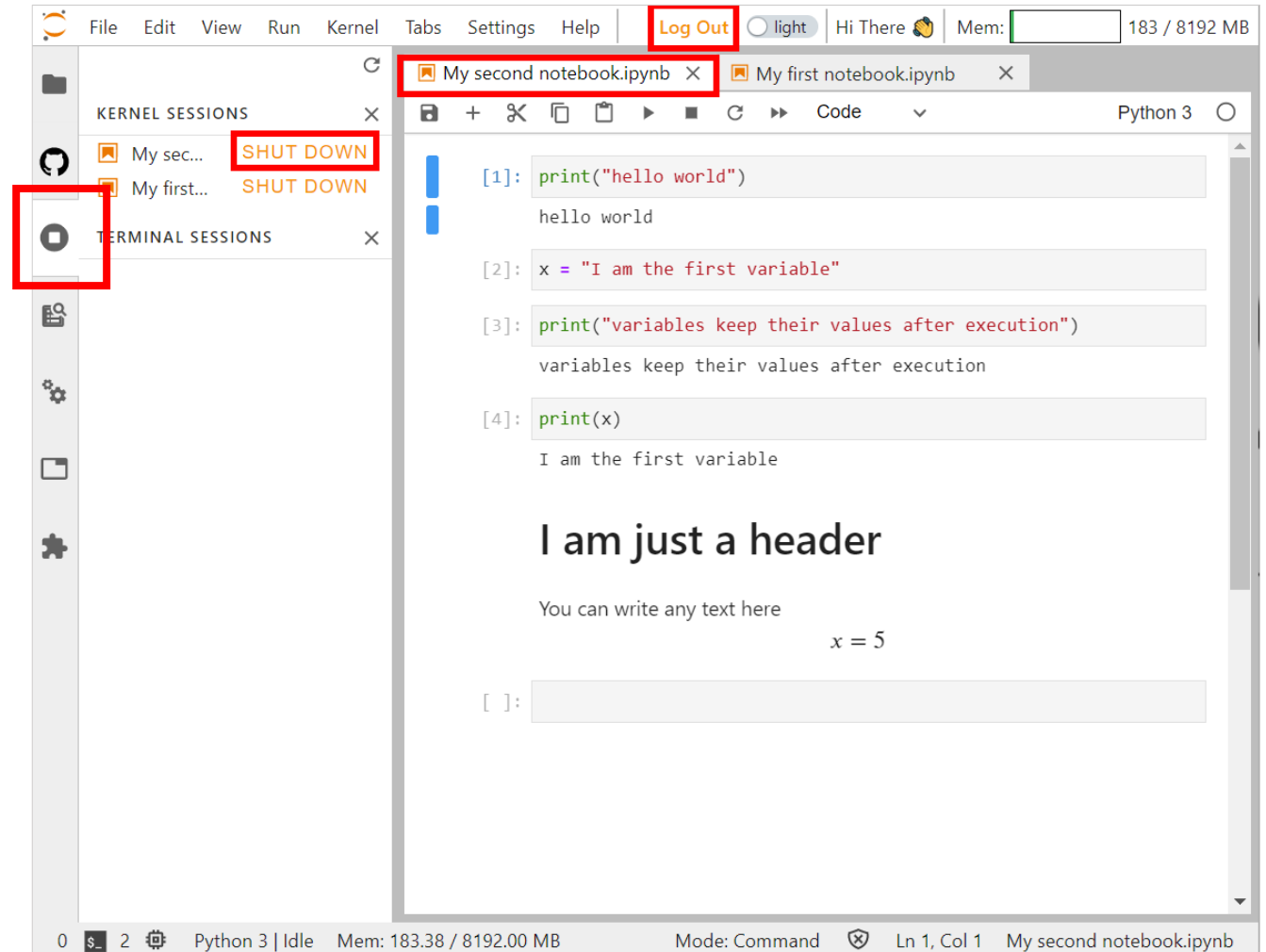
# Shutting Down Kernels and Logout

You might have many notebooks running on your server instance.

Therefore, it is recommended to shut down some of them if you do not need them running.

You can shut down specific kernels on the running kernels tab.

(JupyterLab will regularly save your notebooks, but click the save button before shutting a notebook down.)

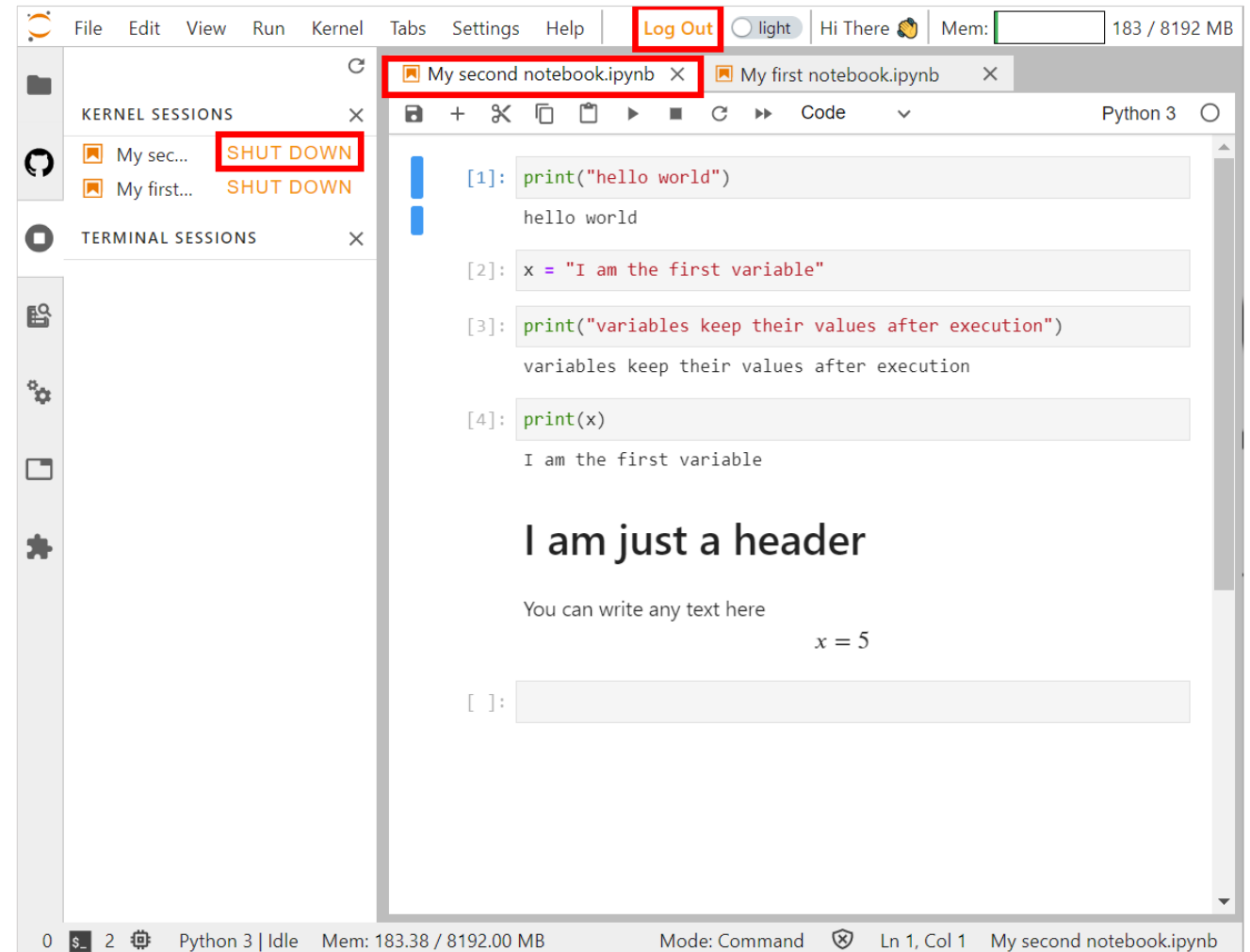


# Shutting Down Kernels and Logout

The following are the most important slides!

Once you are done working on the server, shut down all kernels, and log out by clicking the Log Out button with the yellow color.

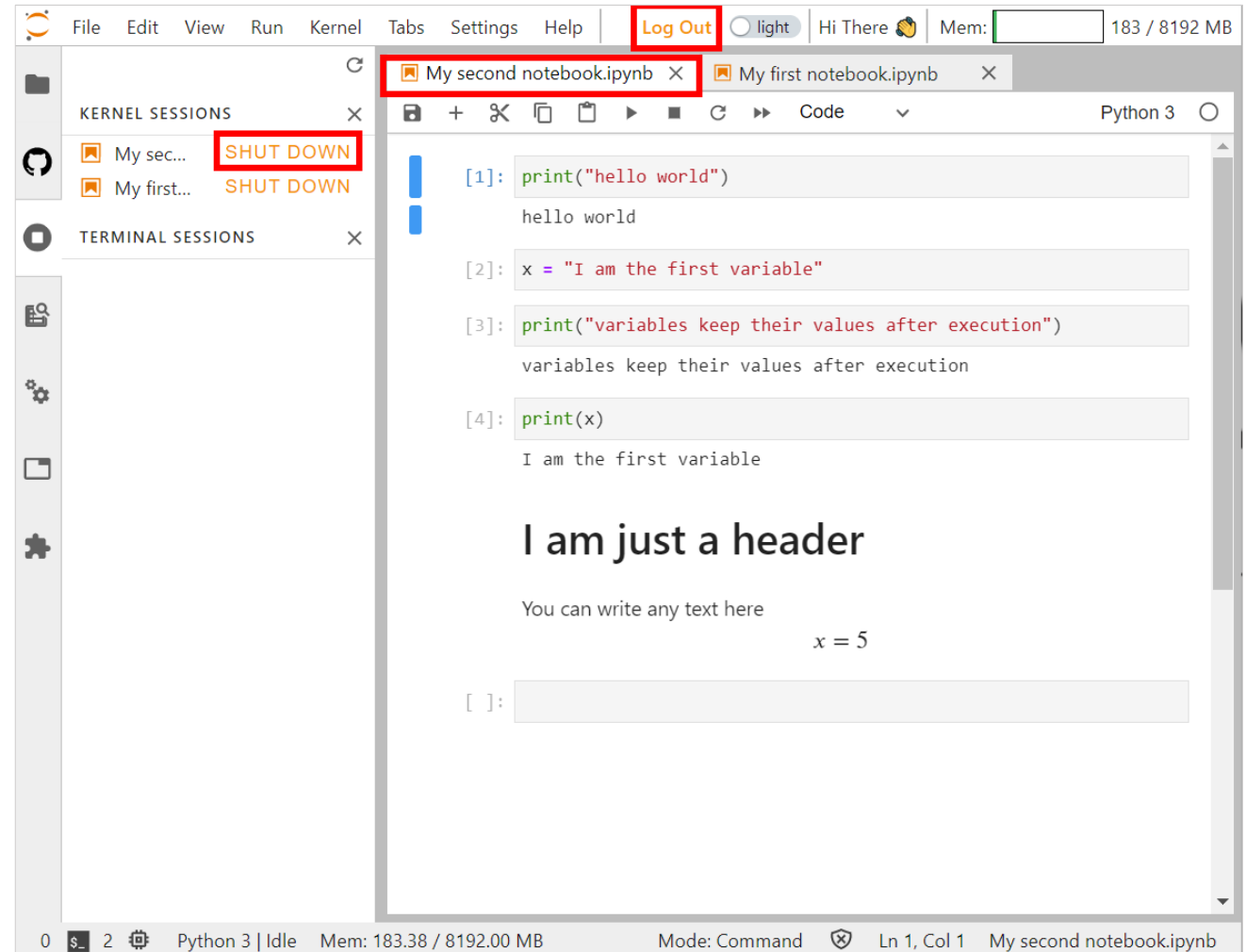
You can also log out within the File menu.



# Shutting Down Kernels and Logout

If you do not shut down the kernels and log out, then your server instance is running in the background and keeps computing, memory.

Other students will not be able to create new server instances!



**If sharing the servers does not work out, then unfortunately we are forced to take measures, which could lead up to closing them down, and then you will have to do the exercises on your private computers.**

**Please be considerate and cooperative!!!**

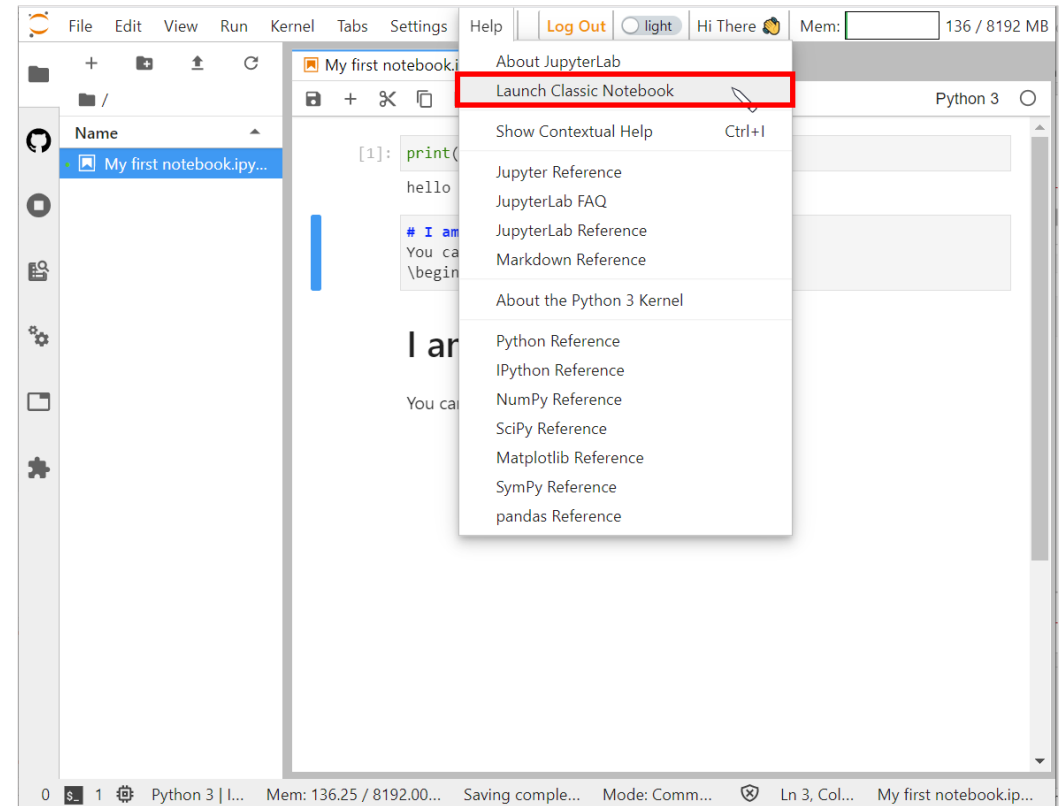
**Thank you, we know you can do it!**

# Classic View

If you are used to working with the Jupyter Notebooks Application and not with JupyterLabs, then you can change your view to a classic mode.

**If not, then no need to try it out.**

But it could prove helpful on small screens as it does not take up as much screen space.





# One Last Note

---

Do not forget to regularly make backups from your notebooks as the server makes no backup itself.

You are responsible for the safety of your files!

# More Resources

---

For further reading, check the following resources:

## **JupyterHub**

<https://jupyterhub.readthedocs.io/en/stable/>

## **JupyterLab**

[https://jupyterlab.readthedocs.io/en/stable/getting\\_started/overview.html](https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html)

## **The Jupyter Notebook**

<https://jupyter-notebook.readthedocs.io/en/stable/notebook.html>

# Quiz

---

What should you always do when you are finished working with a notebook or finished working on the server?

# Quiz

---

What should you always do when you are finished working with a notebook or finished working on the server?

Shut down the kernel(s) and log out!