

Lecture 1

Description, installation and using Jupyter Notebook

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. This lecture will walk you through how to use Jupyter Notebooks for data science projects and how to set it up on your local machine.

First, though: **what is a “notebook”?**

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

Using Notebooks is now a major part of the data science workflow at companies across the globe. If your goal is to work with data, using a Notebook will speed up your workflow and make it easier to communicate and share your results.

Best of all, as part of the open source Project Jupyter (<https://jupyter.org/>), Jupyter Notebooks are completely free. You can download the software on its own (<https://jupyter.org/install>), or as part of the Anaconda (<https://www.anaconda.com/products/individual>) data science toolkit.

Although it is possible to use many different programming languages in Jupyter Notebooks, the most common use case of it is Python.

Installation:

The easiest way for a beginner to get started with Jupyter Notebooks is by installing Anaconda (<https://www.anaconda.com/products/individual>).



Anaconda is the most widely used Python distribution for data science and comes pre-loaded with all the most popular libraries and tools. Some of the biggest Python libraries included in Anaconda include NumPy (<https://numpy.org/>), pandas (<https://pandas.pydata.org/>), and Matplotlib (<https://matplotlib.org/>).

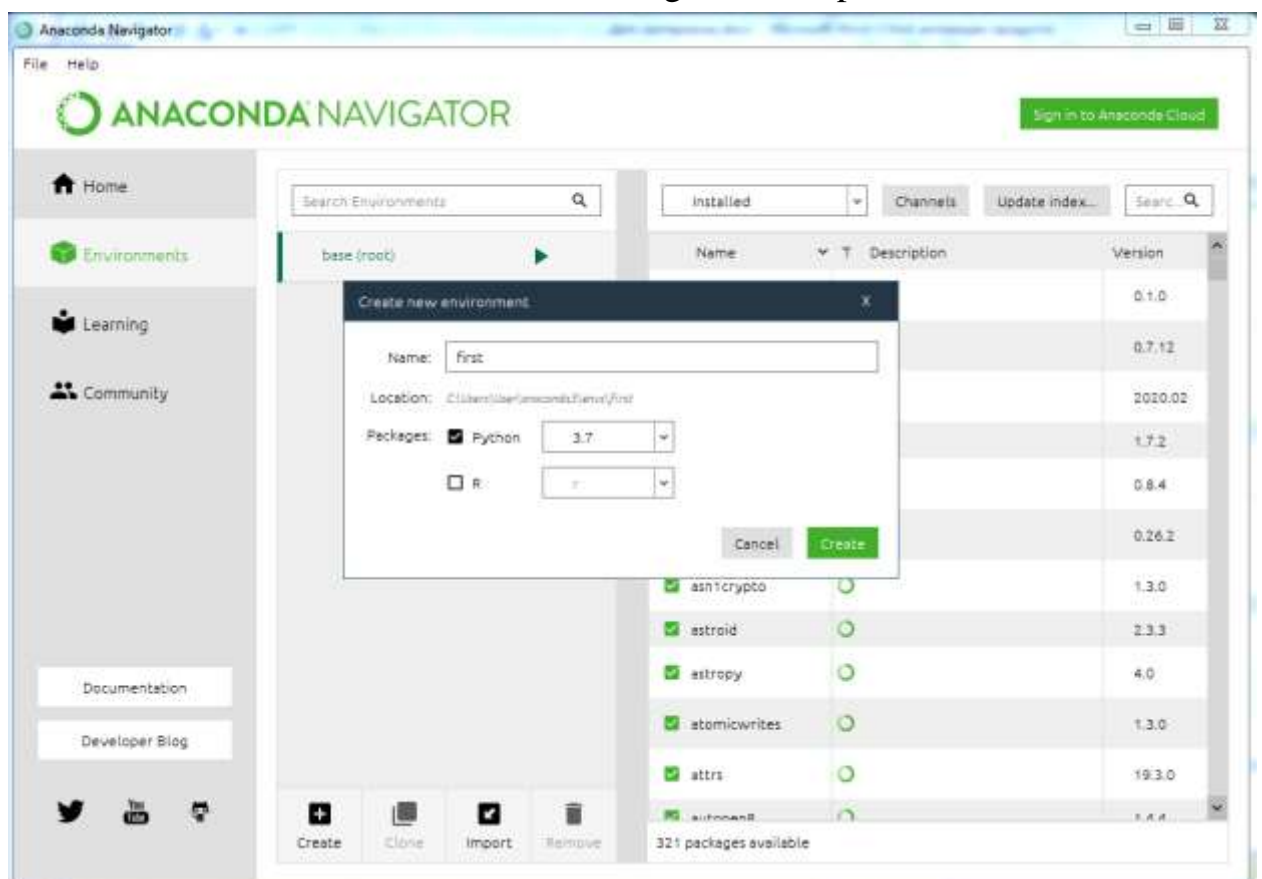
The full list is on – <https://docs.anaconda.com/anaconda/packages/pkg-docs/>.

Anaconda thus lets us running with a fully stocked data science workshop without the hassle of managing countless installations or worrying about dependencies and OS-specific installation issues.

To get Anaconda, simply:

1. Download the latest version of Anaconda for Python 3.8 from <https://www.anaconda.com/products/individual>.
2. Install Anaconda by following the instructions on the download page and/or in the executable.

After installation, the Anaconda Navigator will open:



For convenience, it is recommended to create a separate environment - this will avoid version conflicts (for example, when you need another version of the Python or library). To do this, run the following in the Anaconda Prompt:

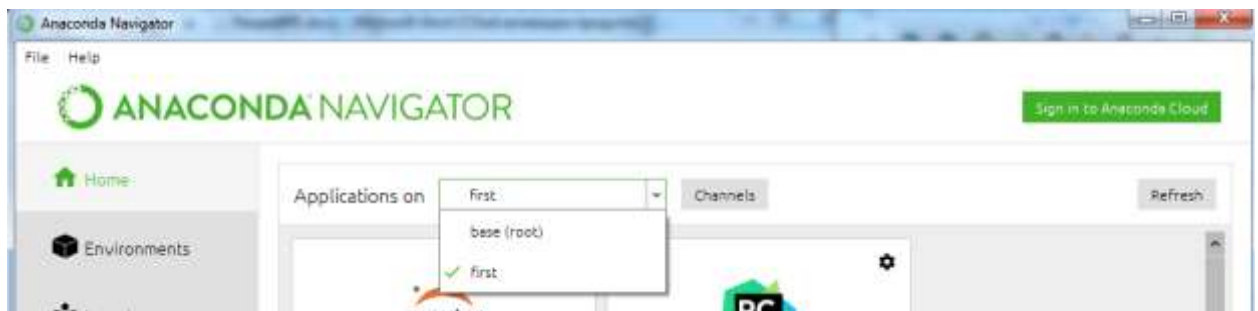
```
conda create --name first python=3.7
```

where **first** is the name of environment. You can also specify the desired version of Python. After creation the environment you should activate it by following:

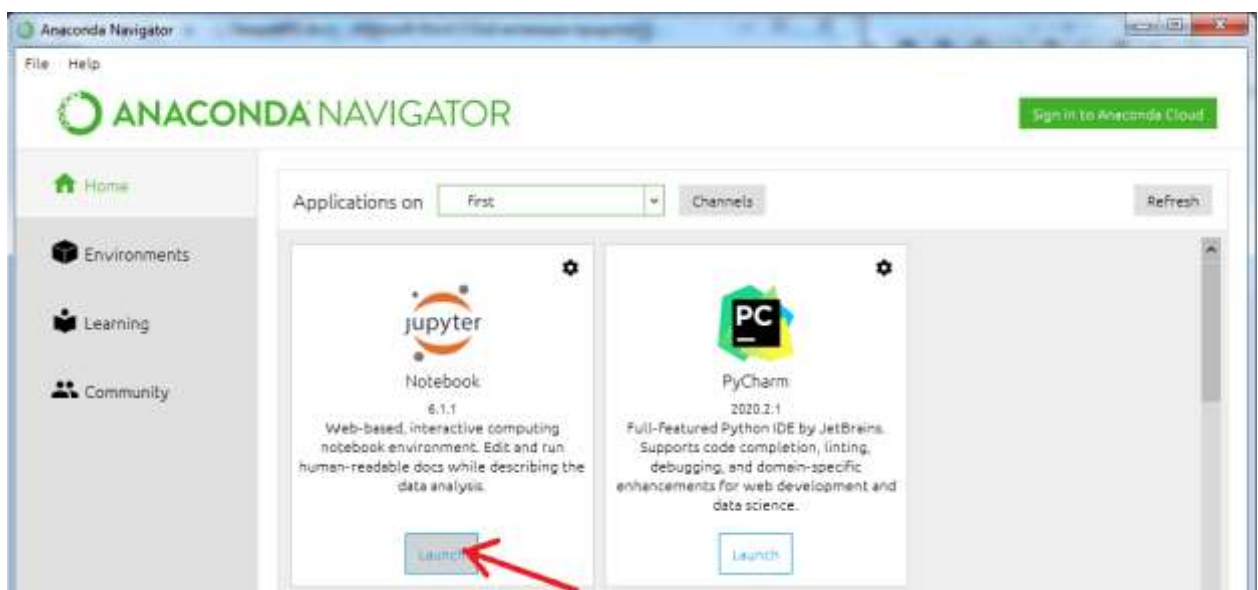
```
conda activate first
```

where **first** is the name of environment. You can also create a new environment in the environments tab of the Anaconda Navigator by clicking on Create

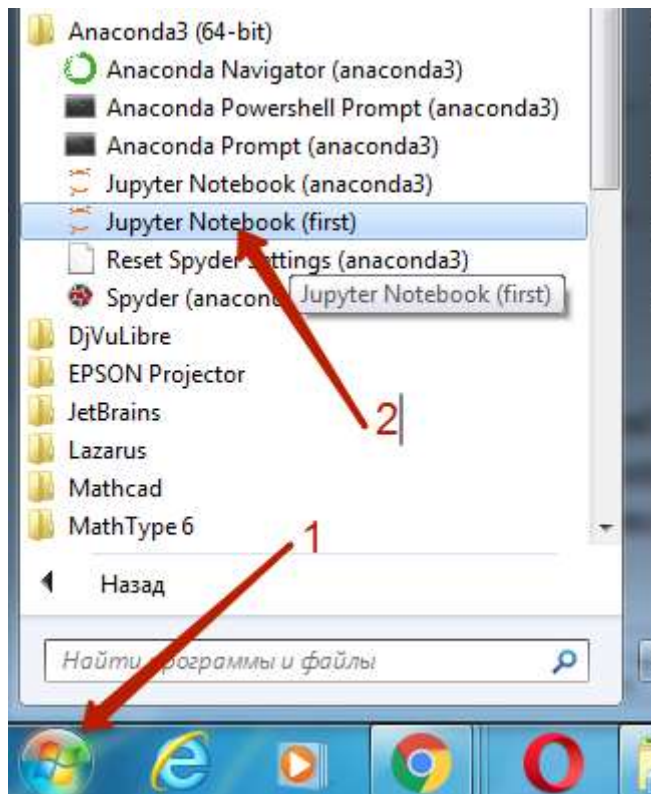
Creating an environment can take several minutes. After installation in the Anaconda Navigator you can select the environment:



Next, you need to install and run the Jupyter Notebook in your chosen environment.



You can also launch Notepad from the Start menu in the Anaconda application group



For advanced users

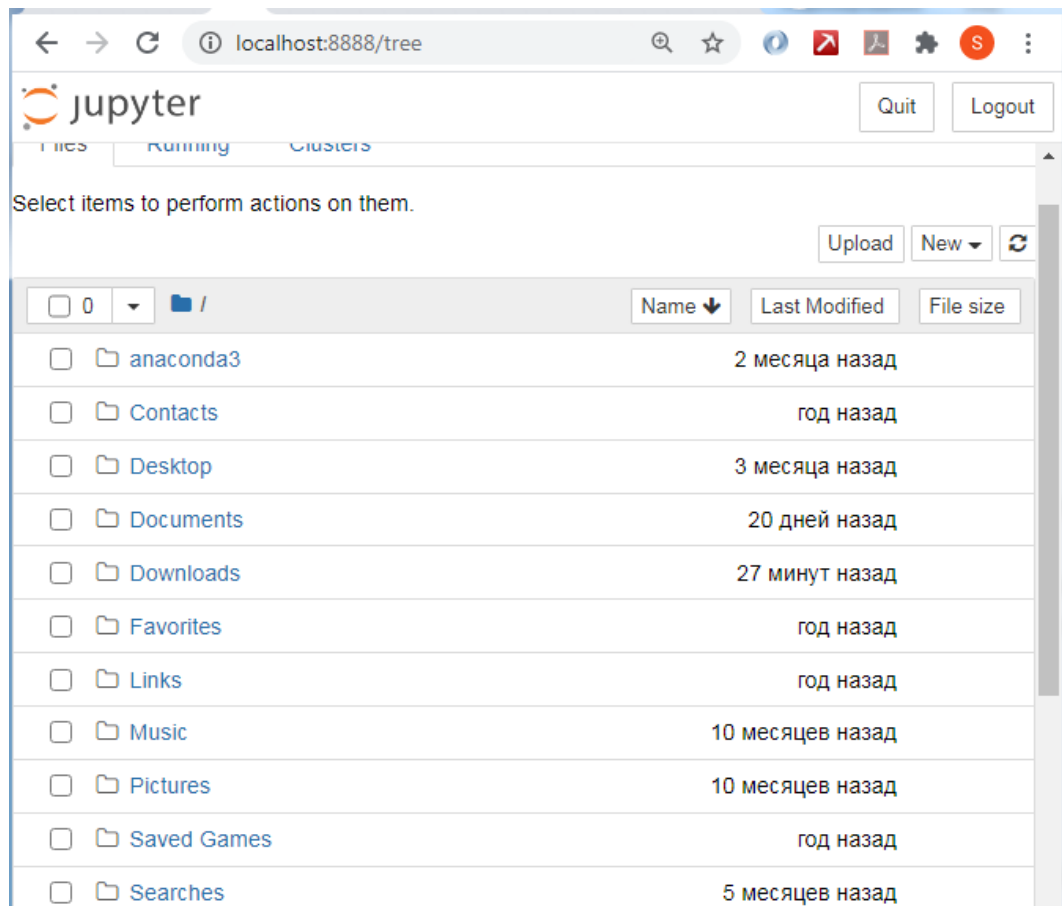
1. Python installation - download file from the site <https://www.python.org/>
2. Run the file. Set PATH - Control Panel > System and Security > System > Advanced System Settings, “Advanced” tab, “Environment Variables” button.
3. Install Jupyter Notebook

```
pip3 install jupyter
```
4. Launch notebook

```
jupyter notebook
```

Creating Your First Notebook

After launching the notebook, a new tab will open in the browser, which looks like this:



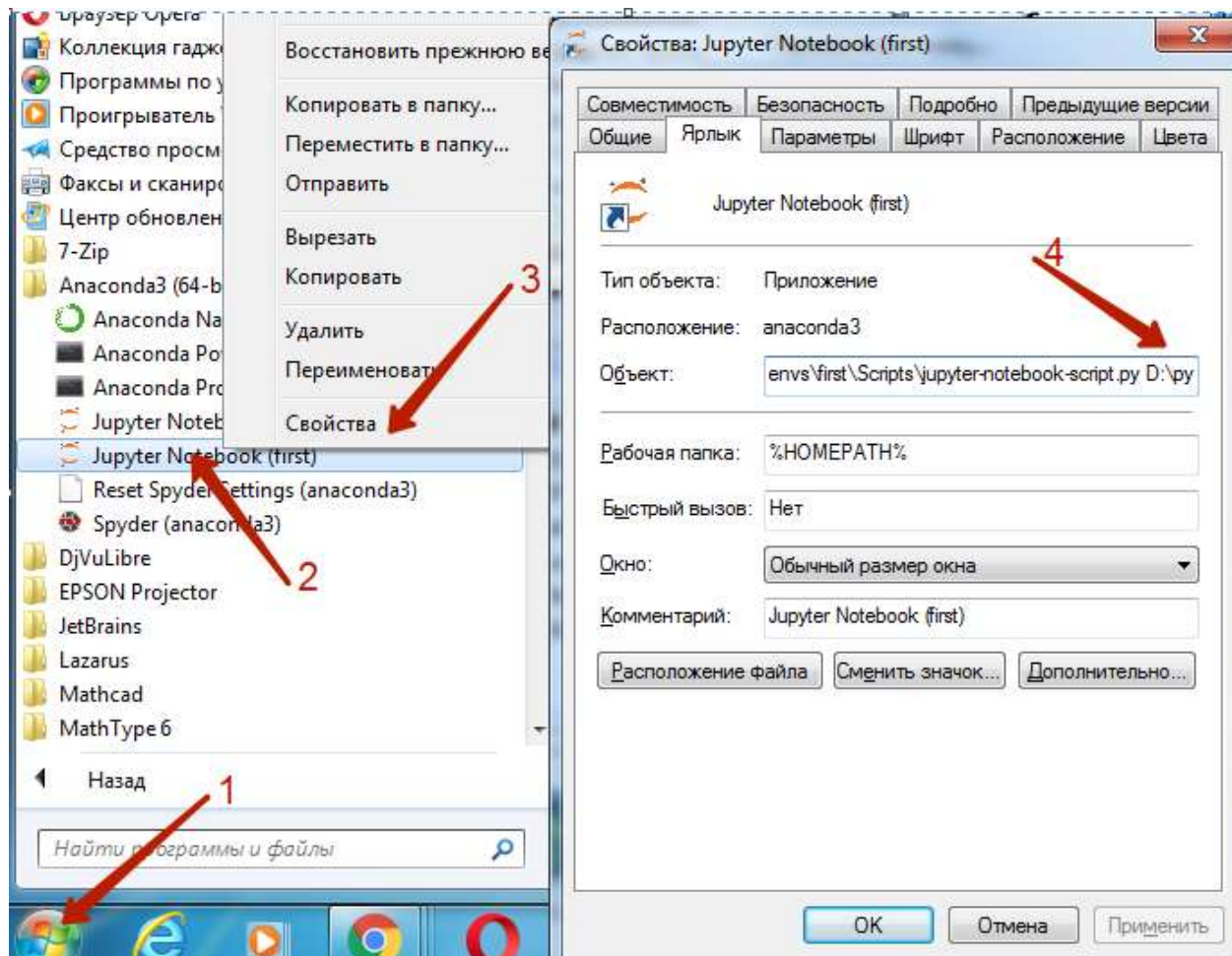
This is the Notebook Dashboard, specifically designed for managing your Jupyter Notebooks.

Be aware that the dashboard will give you access only to the files and sub-folders contained within Jupyter's start-up directory; however, the start-up directory can be changed.

Change the start-up directory of Jupyter Notebook.

When you start from the Start panel:

1. Click Start > All Programs > Anaconda
2. Right-click on the name of the desired notebook and select "Properties"
3. Replace %USERPROFILE% in the "Object" field with the path to the desired folder, for example, D: \ py



When starting from the terminal

1. Change folder using the `cd` command, for example

```
jupyter notebook -notebook-dir "full_path_to_the_folder"
```

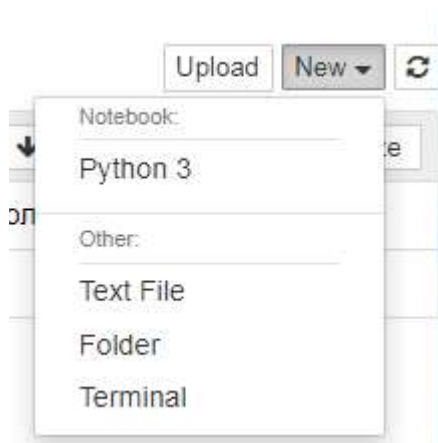
For example,

```
jupyter notebook -notebook-dir "D:\py"
```

Other ways you can view at the link <https://stackoverflow.com/questions/35254852/how-to-change-the-jupyter-start-up-folder>

The URL for the dashboard is something like `http://localhost:8888/tree`. Localhost is not a website, but indicates that the content is being served from your *local* machine: your own computer. Jupyter's Notebooks and dashboard are web apps, and Jupyter starts up a local Python server to serve these apps to your web browser, making it essentially platform independent and opening the door to easier sharing on the web.

To get started, browse to the folder in which you would like to create your first notebook, click the "New" drop-down button in the top-right and select "Python 3"

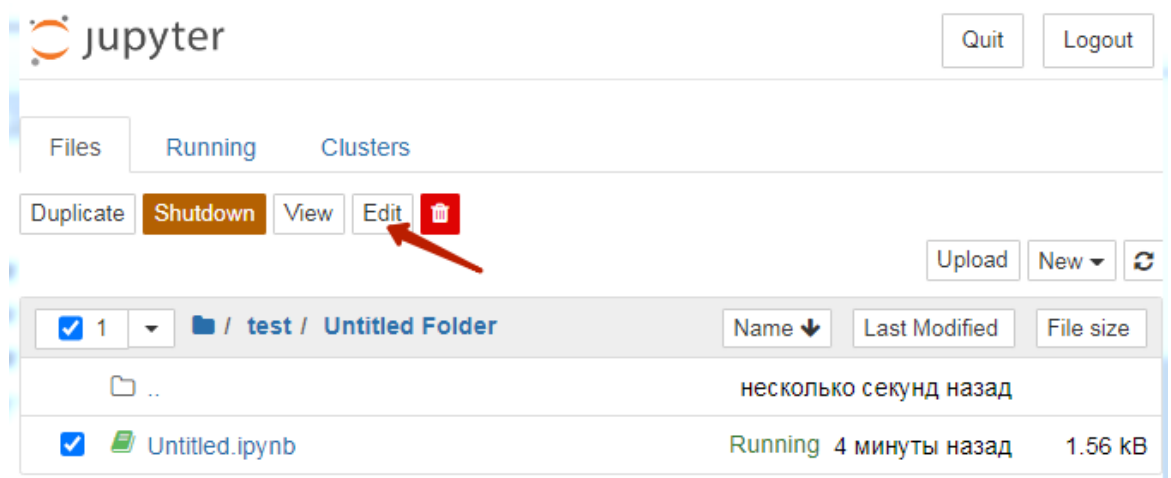


Your first Jupyter Notebook will open in new tab — each notebook uses its own tab because you can open multiple notebooks simultaneously. If you switch back to the dashboard, you will see the new file Untitled.ipynb and you should see some green text that tells you your notebook is running.

What is an ipynb File?

It will be useful to understand what this file really is. Each .ipynb file is a text file that describes the contents of your notebook in a format called JSON. Each cell and its contents, including image attachments that have been converted into strings of text, is listed therein along with some metadata. You can edit this yourself – if you know what you are doing! – by selecting "Edit > Edit Notebook Metadata" from the menu bar in the notebook.

You can also view the contents of your notebook files by selecting **Edit**.



Rename notebook

You can change the file name by double-clicking it.

Rename Notebook

Enter a new notebook name:

My_first_notebook

CancelRename

Kernels and cells

- A kernel is a "computational engine" that executes the code contained in a notebook document.
- A cell is a container for text to be displayed in the notebook or code to be executed by the notebook's kernel.

Cell

Cells form the body of a notebook. In the screenshot of a new notebook in the section above, that box with the green outline is an empty cell. There are two main cell types that we will cover:

- A **code cell** contains code to be executed in the kernel and displays its output below.
- A **Markdown cell** contains text formatted using Markdown and displays its output in-place when it is run.

When you ran the cell, its output will have been displayed below and the label to its left will have changed from In [] to In [1]. The output of a code cell also forms part of the document, which is why you can see it in this article. You can always tell the difference between code and Markdown cells because code cells have that label on the left and Markdown cells do not.

Using list comprehension

In [5]: a = [2**x for x in range(11) if x%2 != 0]
a

no label

label

Out[5]: [2, 8, 32, 128, 512]

The "In" part of the label is simply short for "Input," while the label number indicates when the cell was executed on the kernel — in this case the cell was executed first. Run the cell again and the label will change to In [2] because now the cell was the second to be run on the kernel. It will become clearer why this is so useful later on when we take a closer look at kernels.

Enter the following code in the cell:

```
In [*]: import time
        time.sleep(3) # sleep for 3 seconds
```

This cell doesn't produce any output, but it does take three seconds to execute. Notice how Jupyter signifies that the cell is currently running by changing its label to In [*].

In general, the output of a cell comes from any text data specifically printed during the cell's execution, as well as the value of the last line in the cell, be it a lone variable, a function call, or something else. For example:

```
In [9]: def say_hello(name):
        return 'Hello, {}'.format(name)
        say_hello('Vasya')

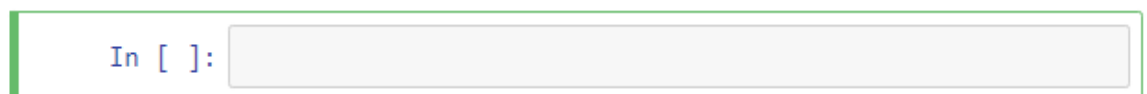
Out[9]: 'Hello, Vasya!'
```

Notebook modes

Jupyter Notebooks have two modes `edit` i `command`.

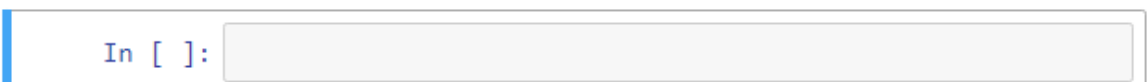
Edit Mode:

To enter edit mode, press **Enter** on your keyboard or click a cell. The edit mode can be defined by the green frame around the cell with the green left field. When you are in edit mode, you can enter text in the cells.

A screenshot of a Jupyter Notebook cell in Edit Mode. The cell is outlined with a green border. On the left side of the cell, the text 'In []:' is displayed in blue. The rest of the cell is a large, empty white rectangular area for text input.

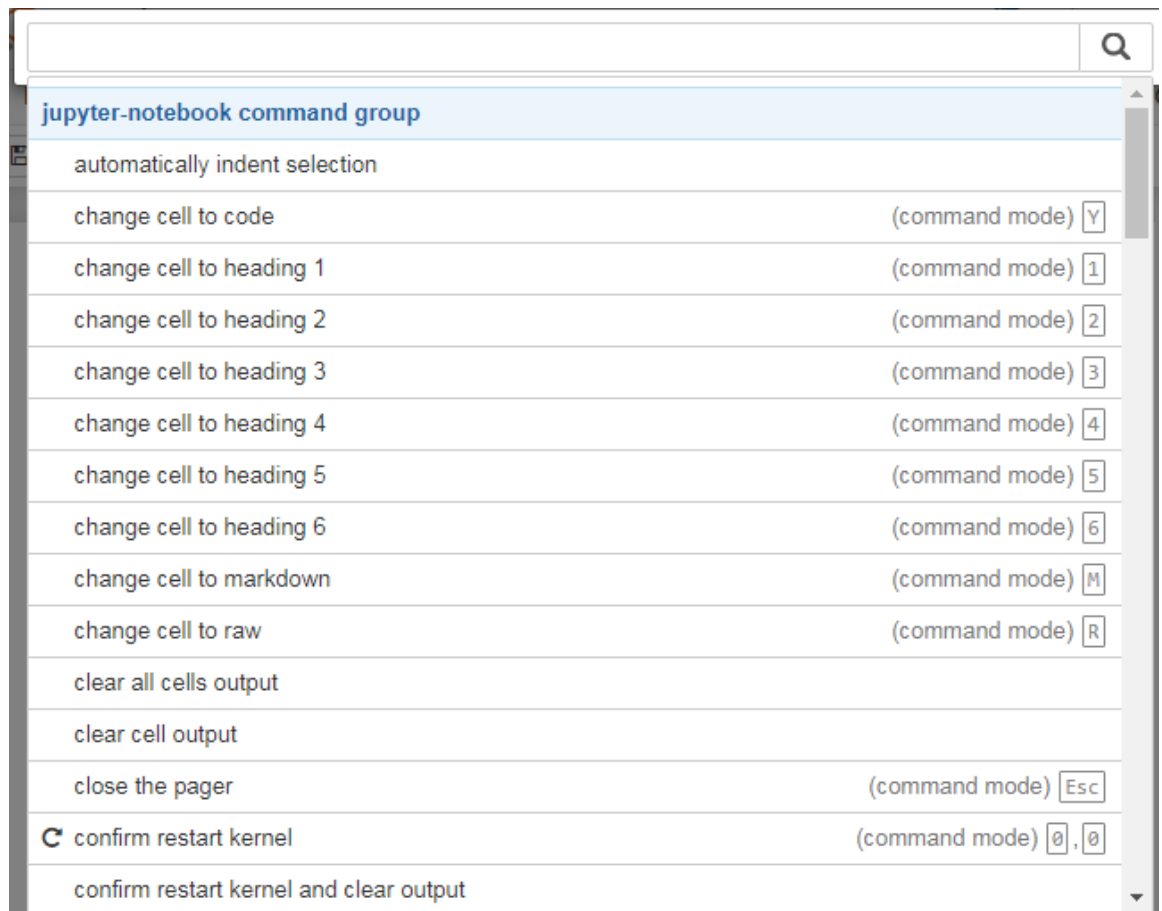
Command mode

To enter command mode, press **Esc** or click anywhere outside the cell. You will see a gray border around the cell with a blue left edge. When you are in command mode, you can edit your notebook, but you cannot enter cells.

A screenshot of a Jupyter Notebook cell in Command Mode. The cell is outlined with a gray border. On the left side of the cell, the text 'In []:' is displayed in blue. The rest of the cell is a large, empty white rectangular area.

Command palette

Pressing the keyboard icon, or Ctrl+Shift +P, opens the keyboard shortcuts palette.



Examples of commands

- Run the selected cell: **Ctrl + Enter**
- Run the cell and insert a new cell below: **Alt + Enter**
- Run the cell and go to the cell below: **Shift + Enter**
- Insert a cell above: **Esc + A**
- Insert a cell below: **Esc + B**
- Cut the selected cells: **Esc + X**
- Copy selected cells: **Esc + C**

Share notebook.

1. Download a notebook, for example on GitHub
- 2a. Enter to browser address bar the path Ввести в адресный рядок https://colab.research.google.com/path_to_your_file_on_GitHub

or

2.6. Go to **nbviewer** (<https://nbviewer.jupyter.org/>) and enter GitHub repository in the appropriate field. In the open repository window, select the desired file.

But nbviewer does not execute notebooks. It only renders the inputs and outputs saved in a notebook document as a web page.

mybinder.org is a separate web service that lets you open notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere. nbviewer shows an *Execute on Binder* icon in its navbar which



Link to Jupyter Notebook with additional materials for this lecture – https://github.com/svniko/data_processing/blob/main/notebooks/Lecture 1 en.ipynb.

Note that not all Out [] cells are rendered properly.

If you open the same laptop in **nbviewer**, everything will be displayed properly - https://nbviewer.jupyter.org/github/svniko/data_processing/blob/main/notebooks/Lecture 1 en.ipynb