

Using IPython/Jupyter Notebook with PyCharm

Before you start

Prior to executing the tasks of this tutorial, make sure that the following prerequisites are met:

- You have a Python project already [created](#). In this tutorial the project **C:/SampleProjects/py/JupyterNotebookExample** is used.
- In the [Project Interpreter](#) page of the **Settings/Preferences** dialog, you have:
 - [Created a virtual environment](#). For this tutorial a virtual environment based on Python 3.6 has been created.
 - [Installed the following package](#):
 - **jupyter**

- `matplotlib`
- `sympy`


Note that PyCharm automatically installs the dependencies of these packages.

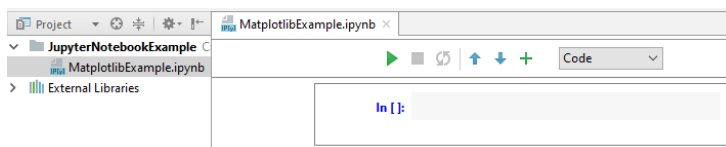
Creating a Jupyter Notebook file

In the [Project Tool Window](#), click `Alt+Insert`.

Then, on the pop-up menu that appears, choose the option **Jupyter Notebook** and type the file name (here it is `MatplotlibExample.ipynb`).

The newly created file now shows up in the [Project Tool Window](#) and automatically [opens for editing](#).


By now, the new file is empty, but PyCharm recognizes it as a [notebook](#) file. As such, this file is marked with the icon  and features a toolbar, which is a complete replica of the real Jupyter Notebook toolbar:



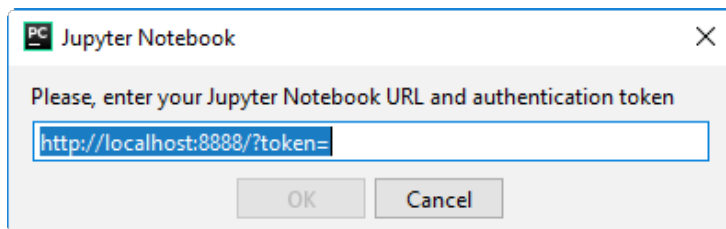
Filling in and running the first cell

This is most easy. Just click the first cell and start typing. For example, in the very first cell type the following code to configure the `matplotlib` package:

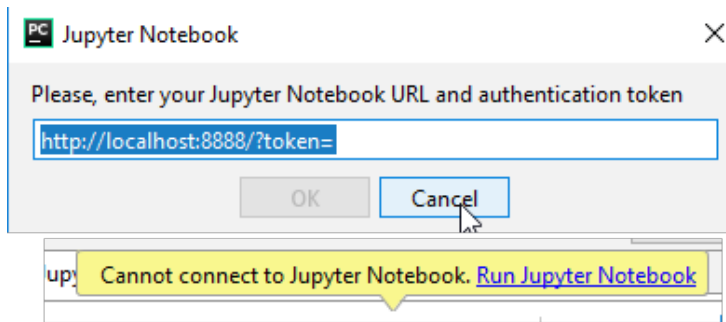
```
%matplotlib inline
```

Next, click the icon  to run the cell (alternatively, you can press `Shift+Enter`).

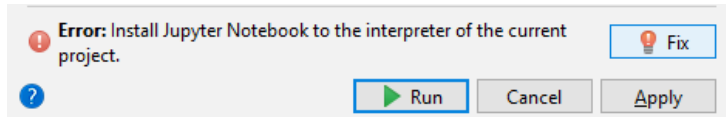
PyCharm shows a dialog box, where you have to specify the URL where the Jupyter Notebook server will run:



In this dialog box, click **Cancel**, and then click the **Run Jupyter Notebook** link:

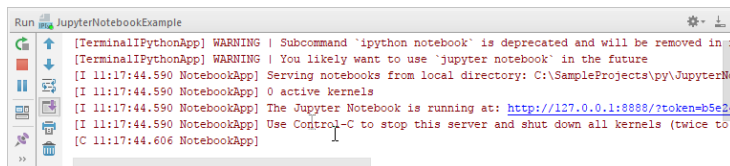


Next, if you didn't install the "Jupyter Notebook" package yet, the run/debug configuration dialog appears showing the error message:

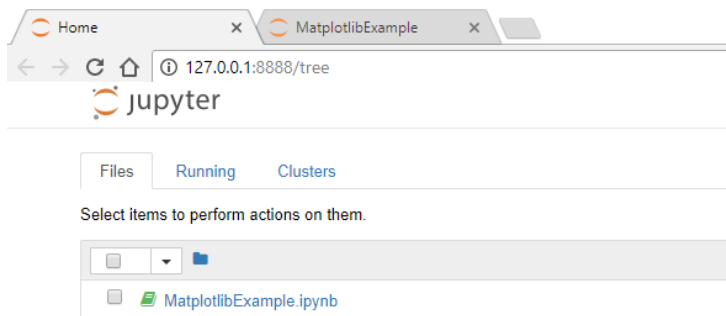


Install the package to fix the problem.

Jupyter server runs in the console:



Follow this address:



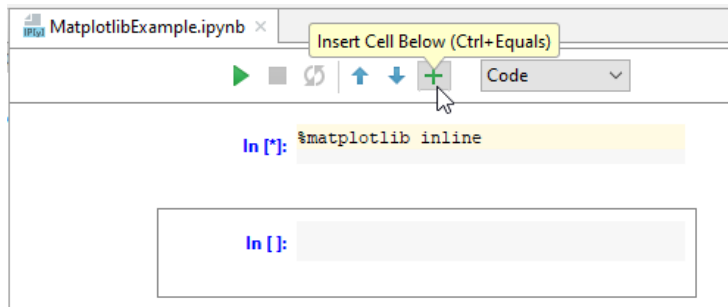
Actually, that's it... From now on you are ready to work with the notebook integration.

Working with cells

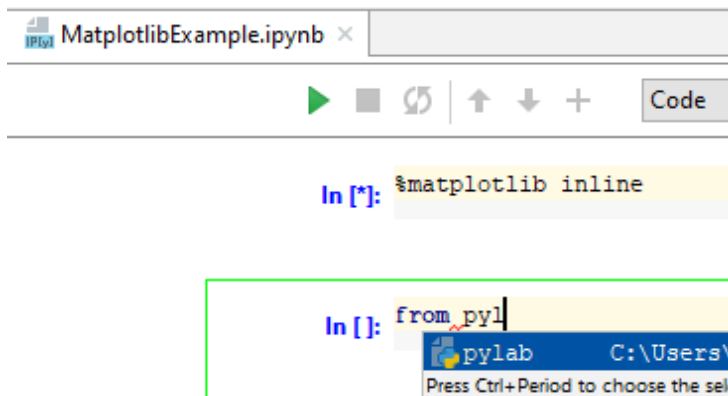
First of all, add the following import statement:

```
from pylab import *
```

This how it's done. To create the next empty cell, click **+** on the toolbar:



Start typing in this cell, and notice [code completion](#):

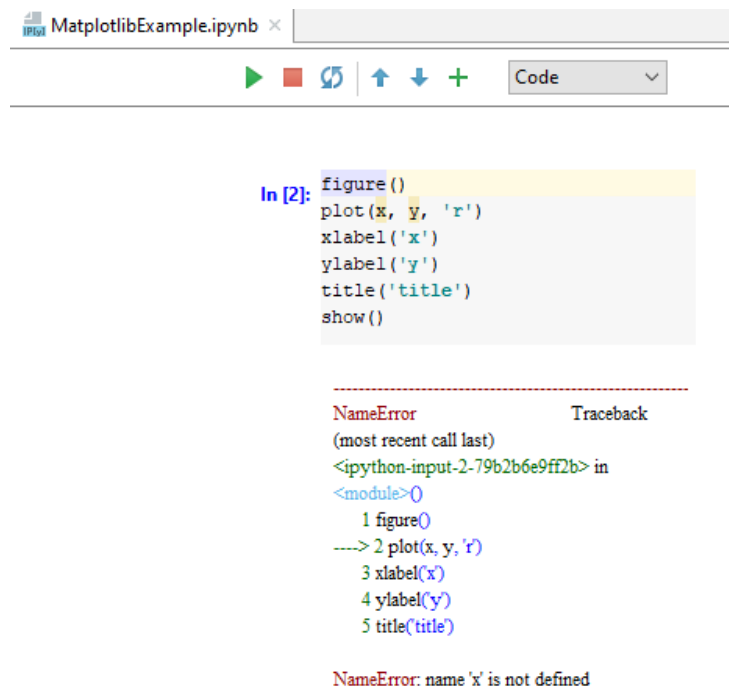


Click **▶** on the toolbar again to run this cell. Note that the cell produces no output, but the next empty cell is created automatically. In this new cell, enter the following code:

```
figure()
plot(x, y, 'r')
xlabel('x')
```

```
ylabel('y')
title('title')
show()
```

Run this cell. Oops! The attempt to run results in an error:



The screenshot shows a Jupyter Notebook interface within PyCharm. The notebook is titled 'MatplotlibExample.ipynb'. The toolbar includes a play button (run), a red square (clear), a refresh icon, and navigation arrows. A dropdown menu is set to 'Code'. The active cell, labeled 'In [2]:', contains the following Python code:

```
figure()
plot(x, y, 'r')
xlabel('x')
ylabel('y')
title('title')
show()
```

Below the code, a red error message is displayed:

```
-----
NameError                                Traceback
(most recent call last)
<ipython-input-2-79b2b6e9ff2b> in
<module>()
      1 figure()
----> 2 plot(x, y, 'r')
      3 xlabel('x')
      4 ylabel('y')
      5 title('title')

NameError: name 'x' is not defined
```

It seems that the variables should be defined first. To do that, add a new cell.

Adding

Since the new cell is added below the current one, click the cell with import statement - its frame becomes green. Then on the toolbar click **+** (or press **Alt+Insert**).

Press **Escape** key to enable the action.

In the created cell, type the import statements

and run them:

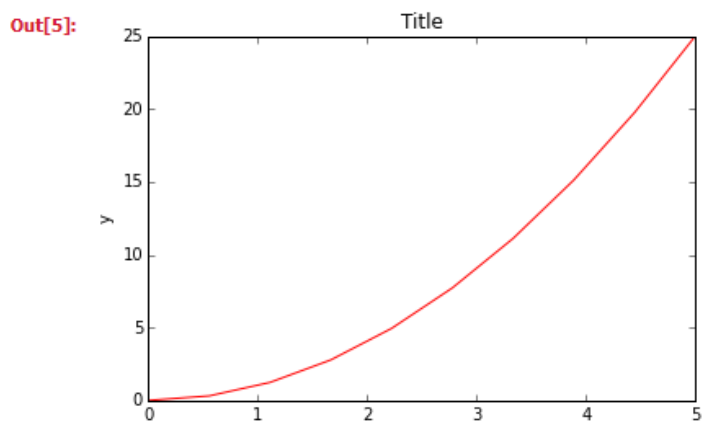
```
In [7]: from pylab import *  
  
In [10]: from matplotlib.pyplot import figure, plot, xlabel, ylabel, title, show
```

The new cell is created automatically. In this cell, type the following code that will define x and y variables:

```
x = linspace(0, 5, 10)  
y = x ** 2
```

Run this cell, and then run the next one. This time it shows the expected output:

```
In [5]: figure()  
        plot(x, y, 'r')  
        xlabel('x')  
        ylabel('y')  
        title('Title')  
        show()
```





Clipboard operations with the cells


You can perform the standard clipboard operations: `Ctrl+C`, `Ctrl+X` and `Ctrl+V`.

Try these shortcuts yourself.

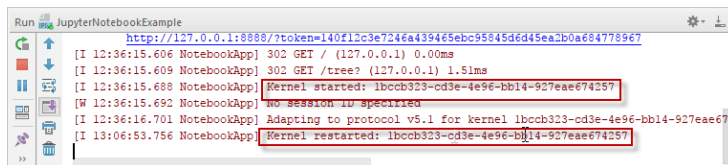
Running and stopping kernels

As you've already learnt, the button  is used to execute a cell.

If calculation of a certain cell takes too much time, you can always stop it. To do that, click  on the document toolbar.


Finally, you can rerun the kernel by clicking  on the document toolbar.

The messages about all these actions show up in the console:



Choosing style

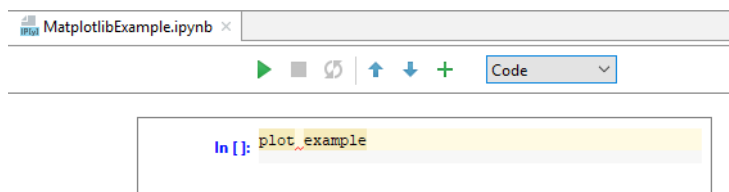
Look at the drop-down list to the right of the document toolbar. It allows you to choose presentation style of a cell. For example, the existing cells are presented as code.


Click the cell with the import statement again, and then click . The new cell appears below. By

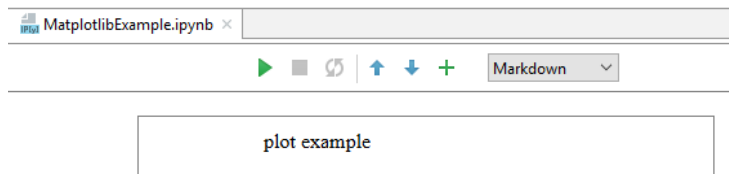
default, its style selector shows **Code**. In this cell, type the following text:

```
plot example
```

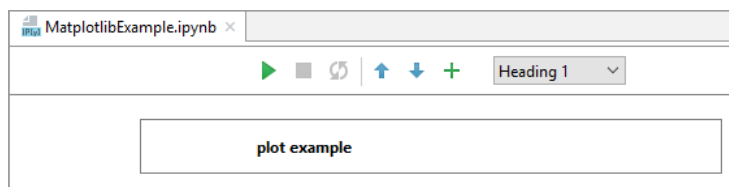
Run this cell and see the error message. Next, click the down arrow, and choose Markdown from the list. The cell changes its view:



Now click  on the toolbar, and see how the cell looks now:




Now you can just select the desired style from the drop-down list, and the view of the cell changes appropriately:

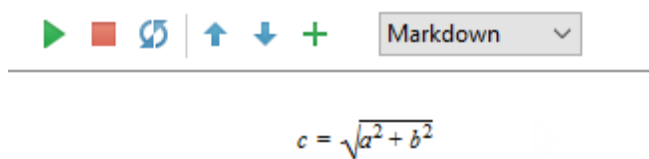


Writing formulae

Add a new cell. In this cell, choose Markdown from the style selector, and type the following text:

```
$$c = \sqrt{a^2 + b^2}$$
```

Click . The result is stunning:



$$c = \sqrt{a^2 + b^2}$$

As you see, PyCharm's Jupyter Notebook integration makes it possible to use [LaTeX notation](#) and render formulae, labels and text.

Next, explore the more complicated case. The expected result - the formula - should appear as the result of calculation. Add a cell and type the following code (taken from [SymPy: Open Source Symbolic Mathematics](#)):

```
from __future__ import division
from IPython.display import display


from sympy.interactive import printing
printing.init_printing(use_latex='math')

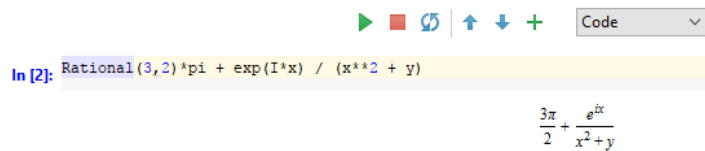
import sympy as sym
from sympy import *
x, y, z = symbols("x y z")
```

```
k, m, n = symbols("k m n", integer=True)
f, g, h = map(Function, 'fgh')
```

Run this cell. It gives no output. Next, add another cell and type the following:

```
Rational(3,2)*pi + exp(I*x) / (x**2 + y)
```

Click  and enjoy:



The image shows a Jupyter Notebook interface. At the top, there is a toolbar with icons for running, clearing, and other actions, along with a dropdown menu labeled 'Code'. Below the toolbar, there is a code cell with the input 'In [2]: Rational(3,2)*pi + exp(I*x) / (x**2 + y)'. The output of this cell is a mathematical expression: $\frac{3\pi}{2} + \frac{e^{ix}}{x^2 + y}$.

Last modified: 29 November 2017