10 Must-Know Jupyter Notebook Tricks

Save time and increase your productivity

Artturi Jalli Follow
Jun 18 · 4 min read ★

Image by the author.

Here is a list of useful Jupyter Notebook tricks. The list is in no particular order.

. . .

1. Shell Commands

Do you exit your notebook to run a shell command?

You can run shell commands in your Jupyter Notebook by placing an exclamation mark before a command.

For example:

```
!pip install Tkinter
```

. . .

2. View a List of Shortcuts

Learning useful shortcut keys for working with Jupyter Notebooks can streamline your workflow over time. The list of shortcuts is huge and there is no way to remember everything at once. This is where viewing a list of shortcuts comes in handy:

1. Open up a Jupyter Notebook.
2. Activate the command mode (press `Esc`).
3. Press the `H` key.
4. See the list of all the shortcuts.

. . .

3. Magic Commands

In Jupyter, there are a bunch of *magic commands* to make your life easier. A magic command is a shortcut to solve common problems, such as listing all the files in the current directory. A magic command is useful, as it can be embedded directly into Python code. A magic command has a `%` prefix.

Here are a bunch of useful magic commands:

```
# Print the current working directory
%pwd

# Show all the files in the current directory
%ls

# Change the working directory
%ls [PATH_TO_DIR]

# List all the variables
%Who
```

MagicCommands.py hosted with ♡ by GitHub view raw

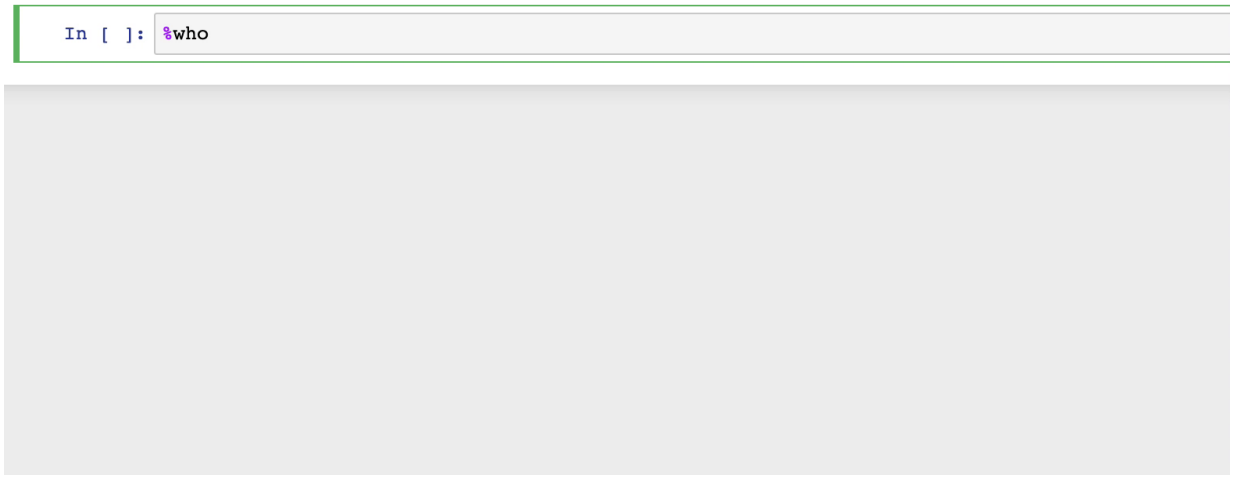
View all the magic commands

The list of useful magic commands is long. View the list of all the available magic commands with:

```
%lsmagic
```

Help with magic commands

To get more info about a specific magic command, highlight it and press `Shift + Tab`:



. . .

4. Measure Cell Execution Time

Use `%%time` to get the elapsed time of running a cell of code.

```
%%time
nums = [i ** 2 for i in range(1, 1_000_000)]
```

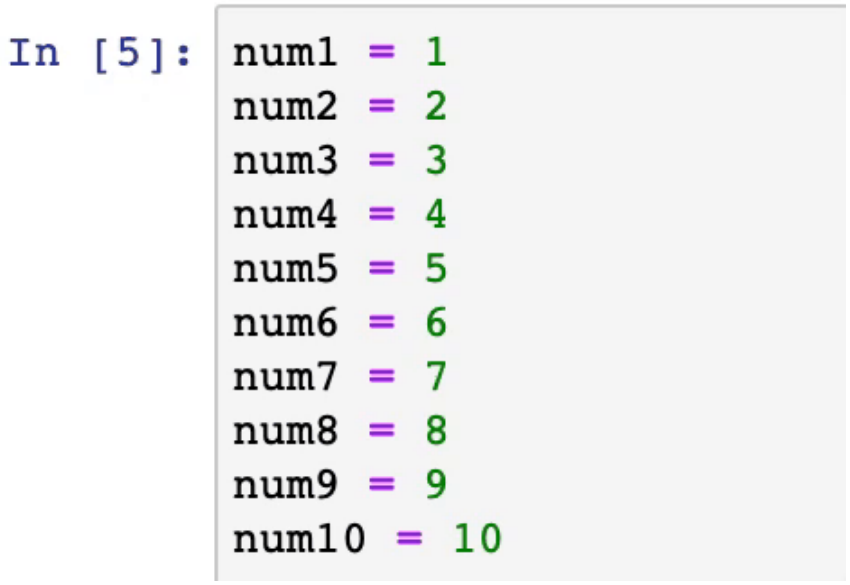
```
CPU times: user 280 ms, sys: 12.8 ms, total: 292 ms
Wall time: 292 ms
```

. . .

5. Add Multiple Cursors

Save time when editing code by working with multiple cursors.

- Windows: Hold `alt` + left-click and drag your cursor.
- Mac: Hold `option` + left-click and drag your cursor.



. . .

6. Set Alarm for Program Completion

Ring an alarm when your program has completed execution.

Windows

On Windows, you can produce a beeping alarm. For example, let's set an alarm of 440HZ for one second (1000ms):

```
import winsound

duration = 1000
freq = 440

winsound.Beep(freq, duration)
```

Mac

Instead of a beeping alarm, you can use the built-in `say` command to make your Mac say something when your program completes:

```
import os
os.system('say "Your program has now finished"')
```

. . .

7. Extensions to Jupyter Notebooks

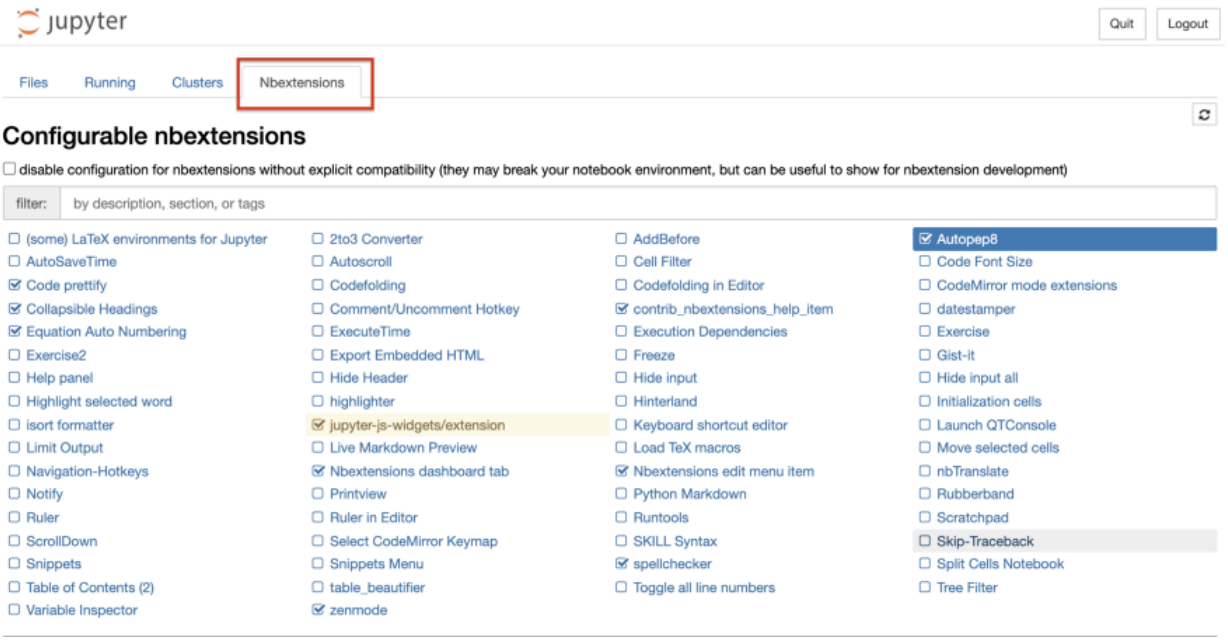
Jupyter Notebook is a great tool, but a bare notebook lacks useful features. This is where the extensions kick in.

1. Run the following command in the command line to install the extensions:

```
pip install jupyter_contrib_nbextensions && jupyter contrib nbextension install
```

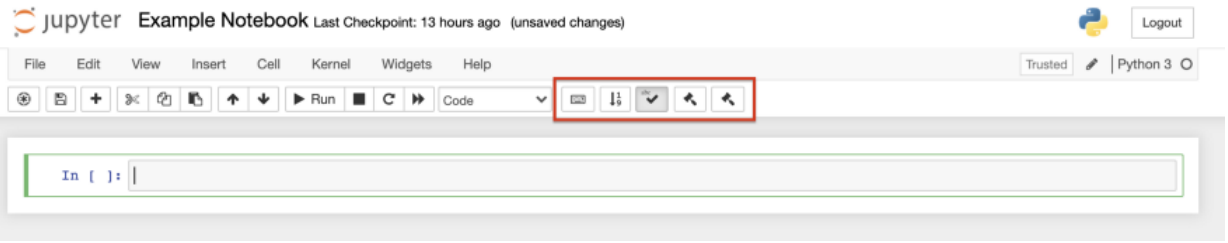
JupyterExtensionInstall.sh hosted with ♡ by GitHub view raw

2. Then start a Jupyter Notebook and go to the "Nbextensions" tab:



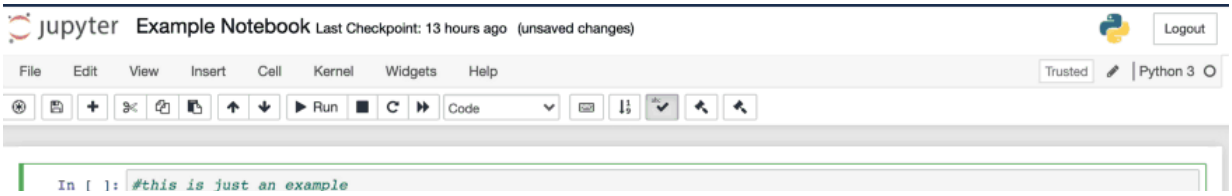
Useful extensions you may want to activate

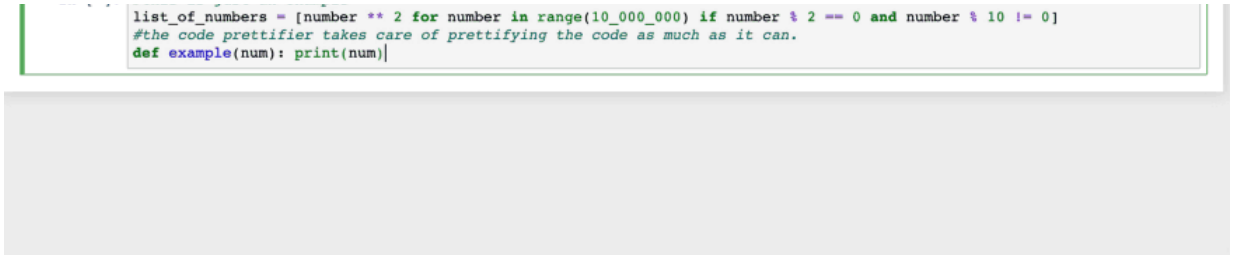
3. Activate the extensions by clicking them. These are added as buttons to the notebook toolbar:



Some activated extensions in the toolbar

For instance, use the Code Prettify extension to make your code look structured and beautiful with one click of a button:



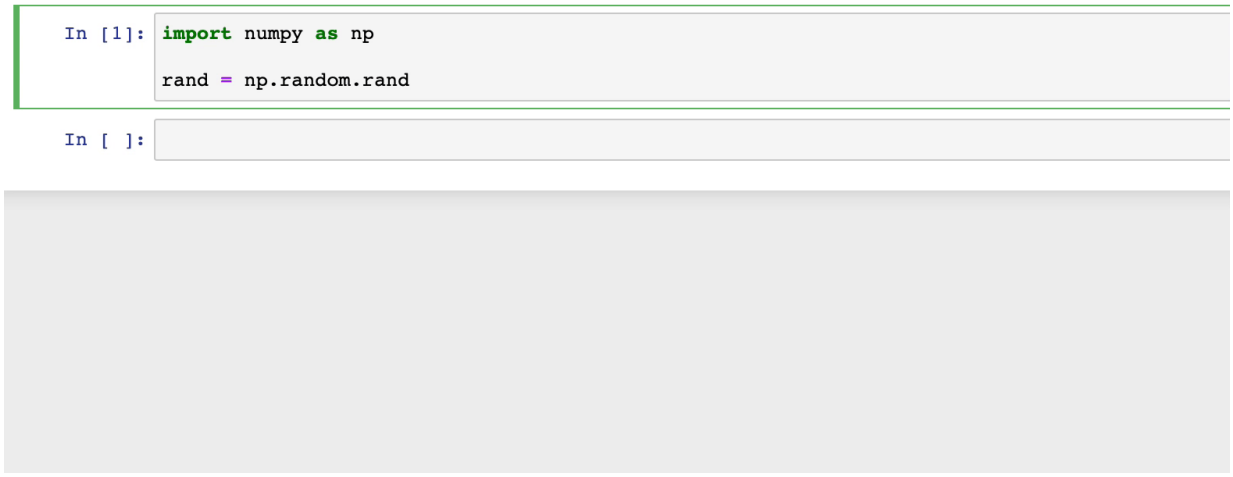


. . .

8. View the Documentation of a Method

To view the documentation of a method, highlight the method and press `Shift + Tab`.

To further expand the modal, press the `+` button on the top right to expand the modal:



. . .

9. Extend the Number of Columns and Rows Shown in pandas

A pandas table can only show a limited number of rows and columns. However, you can change that.

For instance, let's set the max output rows and columns to `1000`:

```
import pandas as pd

pd.set_option('display.max_rows', 1000)
pd.set_option('display.max_columns', 1000)
```

. . .

10. Hide Unnecessary Output

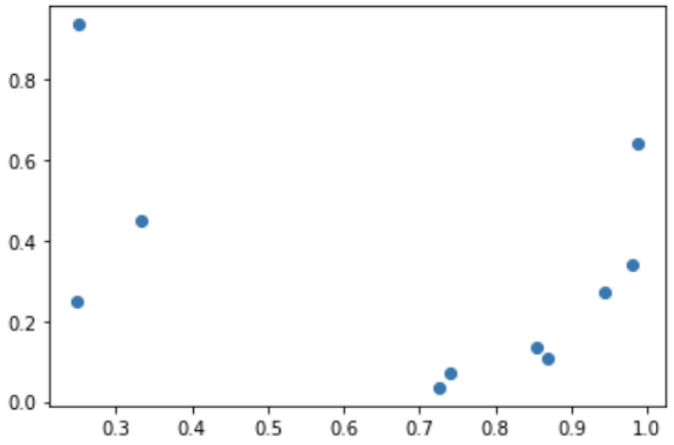
Use a semicolon at the end of a statement to suppress an annoying output. For example, when plotting with Matplotlib, you see a somewhat redundant output before the graph:

```
In [3]: from matplotlib import pyplot as plt
        import numpy as np

        x = np.random.rand(10)
        y = np.random.rand(10)

        plt.scatter(x,y)
```

```
Out[3]: <matplotlib.collections.PathCollection at 0x7fa5f968fa60>
```



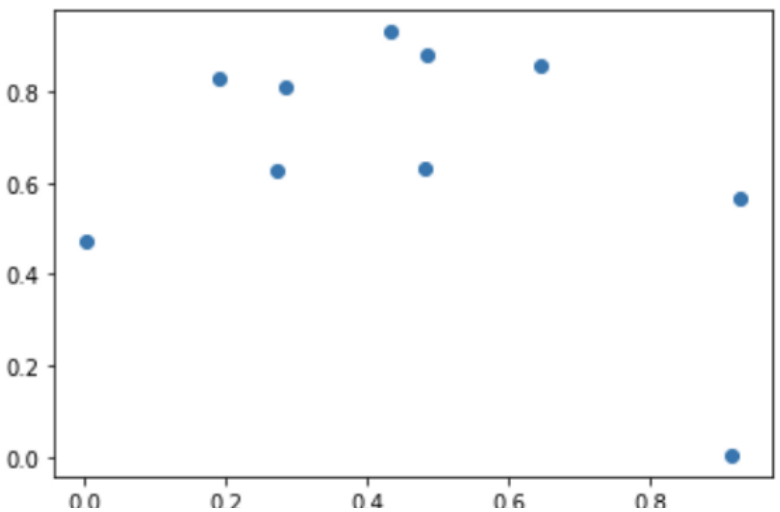
To get rid of this, use a semicolon after the plotting statement:

```
plt.scatter(x,y);
```

```
In [4]: from matplotlib import pyplot as plt
        import numpy as np

        x = np.random.rand(10)
        y = np.random.rand(10)

        plt.scatter(x,y);
```



Looks better, doesn't it?

. . .

Conclusion

That's it for the list. I hope I was able to increase your productivity and save your time with these tricks.

Thanks for reading. Happy coding!

I'd love to join your LinkedIn network. Feel free to connect Artturi Jalli.

. . .

You May Find This Insightful

Thanks to Anupam Chugh.