Efficient Python Tricks and Tools for Data Scientists - By Khuyen Tran

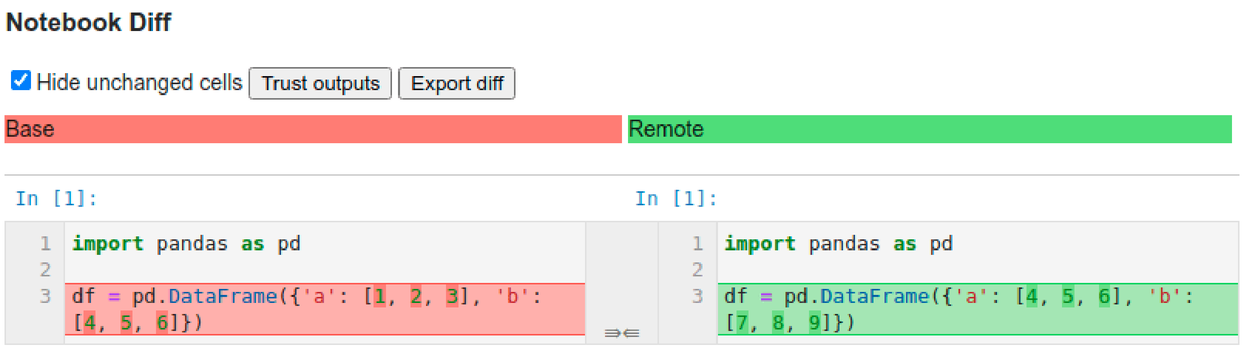
Jupyter Notebook

GitHub View on GitHub Book View Book

This section covers some tools to work with Jupyter Notebook.

nbdime: Better Version Control for Jupyter Notebook

If you want to compare the previous version and the current version of a notebook, use nbdime. The image below shows how 2 versions of a notebook are compared with nbdime.



```
import pandas as pd

df = pd.DataFrame({'a': [1, 2, 3], 'b':
[4, 5, 6]})
```

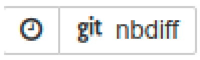
```
import pandas as pd

df = pd.DataFrame({'a': [4, 5, 6], 'b':
[7, 8, 9]})
```

To install nbdime, type:

```
pip install nbdime
```

After installing, click the little icon in the top right corner to use nbdime.



Link to nbdime.

display in IPython: Display Math Equations in Jupyter Notebook

If you want to use latex to display math equations in Jupyter Notebook, use the display module in the IPython library.

```
from IPython.display import display, Math,
Latex

a = 3
b = 5
print("The equation is:")
display(Math(f'y= {a}x+{b}'))
```

```
The equation is:
```

$$y = 3x + 5$$

Reuse The Notebook to Run The Same Code Across Different Data

Have you ever wanted to reuse the notebook to run the same code across different data? This could be helpful to visualize different data without changing the code in the notebook itself.

Papermill provides the tool for this. Insert the tag `parameters` in a notebook cell that contains the variable you want to parameterize.

Then run the code below in the terminal.

```
$ papermill input.ipynb output.ipynb -p
data=data1
```

`-p` stands for parameters. In this case, I specify the data I want to run with `-p data=<name-data>`

Link to papermill

watermark: Get Information About Your Hardware and the Packages Being Used within Your Notebook

If you want to get information about your hardware and the Python packages being used within your notebook, use the magic extension watermark.

The code below shows the outputs of the watermark in my notebook.

```
%load_ext watermark
```

```
%watermark
```

```
Last updated: 2021-09-12T09:58:22.438535-05:00

Python implementation: CPython
Python version       : 3.8.10
IPython version      : 7.27.0

Compiler    : GCC 9.4.0
OS          : Linux
Release     : 5.4.0-81-generic
Machine     : x86_64
Processor   : x86_64
CPU cores   : 16
Architecture: 64bit
```

We can also use watermark to show the versions of the libraries being used:

```
import numpy as np
import pandas as pd
import sklearn
```

```
%watermark --iversions
```

```
sklearn: 0.0
pandas : 1.3.2
numpy  : 1.19.5
```

Link to watermark.

Generate requirements.txt File for Jupyter Notebooks Based on Imports

`pip freeze` saves all packages in the environment, including ones that you don't use in your current project. To generate a `requirements.txt` based on imports in your Jupyter Notebooks, use pipreqsnb.

For example, to save all packages in your current project to a `requirements.txt` file, run:

```
$ pipreqsnb .
```

```
pipreqs  .
INFO: Successfully saved requirements file in
./requirements.txt
```

Your `requirements.txt` should look like below:

```
pandas==1.3.4
numpy==1.20.3
ipython==7.30.1
scikit_learn==1.0.2
```

Usage of pipreqsnb:

```
Usage:
    pipreqsnb [options] <path>

Options:
    --use-local              Use ONLY local
package info instead of querying PyPI
    --pypi-server <url>   Use custom PyPi
server
    --proxy <url>         Use Proxy, parameter
will be passed to requests library. You can
also just set the
                          environments
parameter in your terminal:
                          $ export
HTTP_PROXY="http://10.10.1.10:3128"
                          $ export
HTTPS_PROXY="https://10.10.1.10:1080"
    --debug               Print debug
information
    --ignore <dirs>...    Ignore extra
directories (sepparated by comma no space)
    --encoding <charset>  Use encoding
parameter for file open
    --savepath <file>     Save the list of
requirements in the given file
    --print               Output the list of
requirements in the standard output
    --force               Overwrite existing
requirements.txt
    --diff <file>         Compare modules in
requirements.txt to project imports.
```

```
    --clean <file>        Clean up
requirements.txt by removing modules that are
not imported in project.
    --no-pin              Omit version of
output packages.
```

Link to pipreqsnb

To generate requirements.txt for Python scripts, use pipreqs instead.