

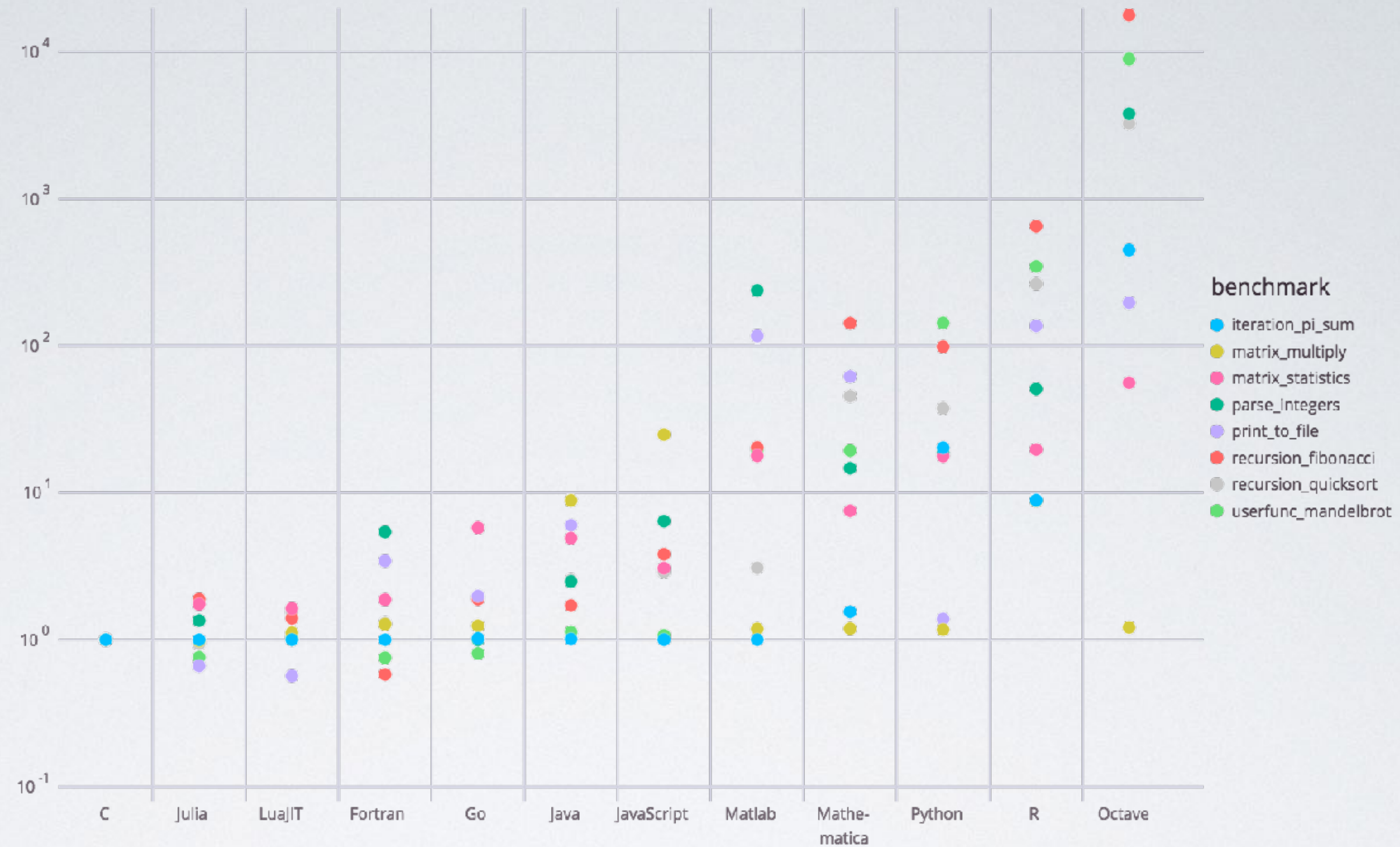
Traitement de données avec PYTHON

POURQUOI PYTHON?

Nous aide à être plus efficace dans notre travail de chercheur
en tirant avantage des outils disponibles

- Bénéficie d'un large écosystème d'outils matures : ex. : Numpy, Pandas, SciPy, Matplotlib, IPython/Jupyter, etc.
- Très large éventail d'outils développés par d'autres utilisateurs offert en code libre

Source



COMPARAISON

avec d'autres langages de programmation

TERMINAL OU NOTEBOOK

If Python is the engine of our data science task, you might think of IPython as the interactive control panel.

— Jake Vanderplast

```

samuelduchesne — bash — 86x34
-s      : don't add user site directory to sys.path; also PYTHONUSERSTF
-S      : don't imply 'import site' on initialization
-u      : unbuffered binary stdout and stderr, stdin always buffered;
         also PYTHONUNBUFFERED=x
         see man page for details on internal buffering relating to '-u'
-v      : verbose (trace import statements); also PYTHONVERBOSE=x
         can be supplied multiple times to increase verbosity
-V      : print the Python version number and exit (also --version)
         when given twice, print more information about the build
-W arg  : warning control; arg is action:message:category:module:lineno
         also PYTHONWARNINGS=arg
-x      : skip first line of source, allowing use of non-Unix forms of #!cmd
-X opt  : set implementation-specific option
file    : program read from script file
-       : program read from stdin (default; interactive mode if a tty)
arg ... : arguments passed to program in sys.argv[1:]

Other environment variables:
PYTHONSTARTUP: file executed on interactive startup (no default)
PYTHONPATH   : ':'-separated list of directories prefixed to the
               default module search path. The result is sys.path.
PYTHONHOME   : alternate <prefix> directory (or <prefix>:<exec_prefix>).
               The default module search path uses <prefix>/pythonX.X.
PYTHONCASEOK : ignore case in 'import' statements (Windows).
PYTHONIOENCODING: Encoding[:errors] used for stdin/stdout/stderr.
PYTHONFAULTHANDLER: dump the Python traceback on fatal errors.
PYTHONHASHSEED: if this variable is set to 'random', a random value is used
               to seed the hashes of str, bytes and datetime objects. It can also be
               set to an integer in the range [0,4294967295] to get hash values with a
               predictable seed.
PYTHONMALLOC: set the Python memory allocators and/or install debug hooks
               on Python memory allocators. Use PYTHONMALLOC=debug to install debug
               hooks.
MacBook-Pro:~ samueld$
```

```

localhost
Notebook
jupyter Notebook
Logout
Trusted | Python 3
File Edit View Insert Cell Kernel Widgets Help
Code
In [6]: 1 import pandas as pd
        2 import numpy as np

In [15]: 1 index = [('California', 2000), ('California', 20
                ('New York', 2000), ('New York', 2010),
                ('Texas', 2000), ('Texas', 2010)]
        2 populations = [33871648, 37253956,
                18976457, 19378102,
                20851820, 25145561]
        3 pop = pd.Series(populations, index=index)
        4 pop

Out[15]: (California, 2000)    33871648
          (California, 2010)    37253956
          (New York, 2000)     18976457
```


Téléchargez
l'exemple

EXEMPLE

BemSolar

Démonstration d'un code simple
permettant de récupérer, traiter et analyser
les données de rayonnement solaire du
National Oceanic and Atmospheric
Administration.

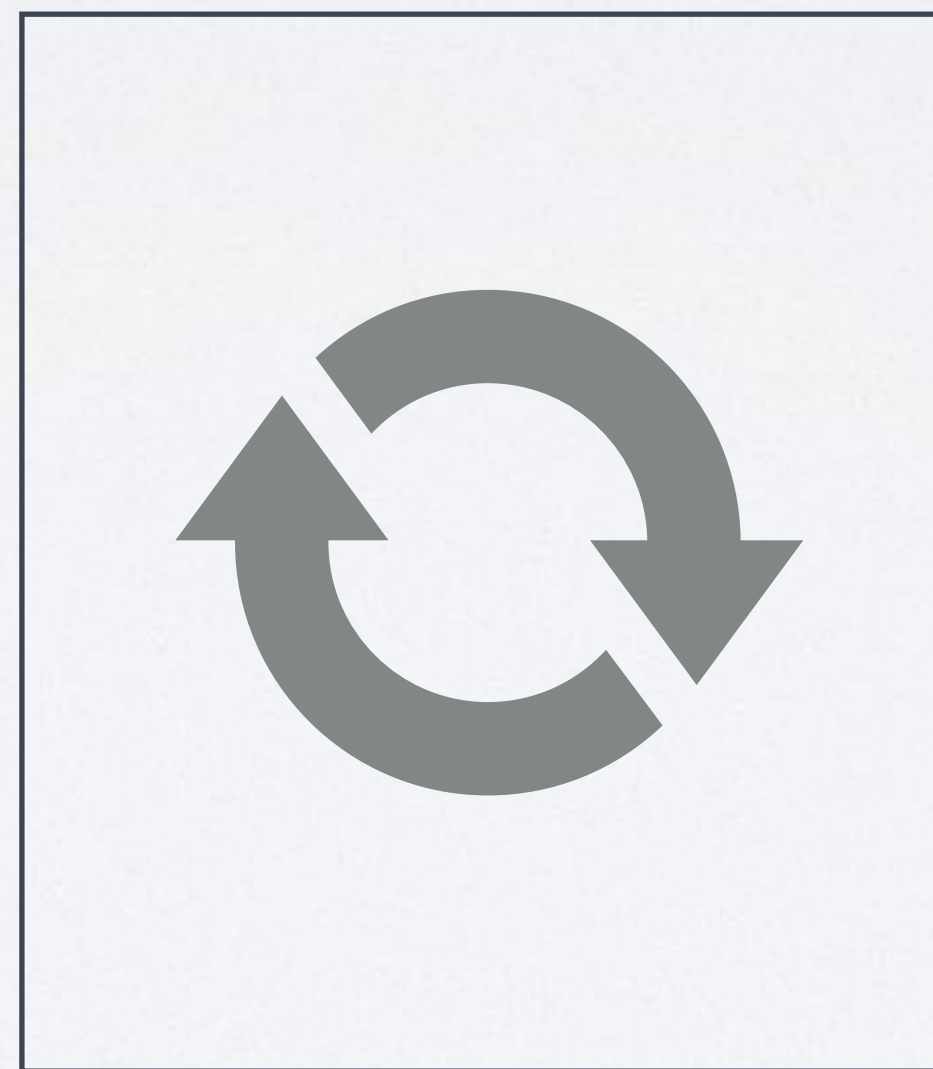
Python, Jupyter, Github

Une stratégie de travail efficace en trois parties

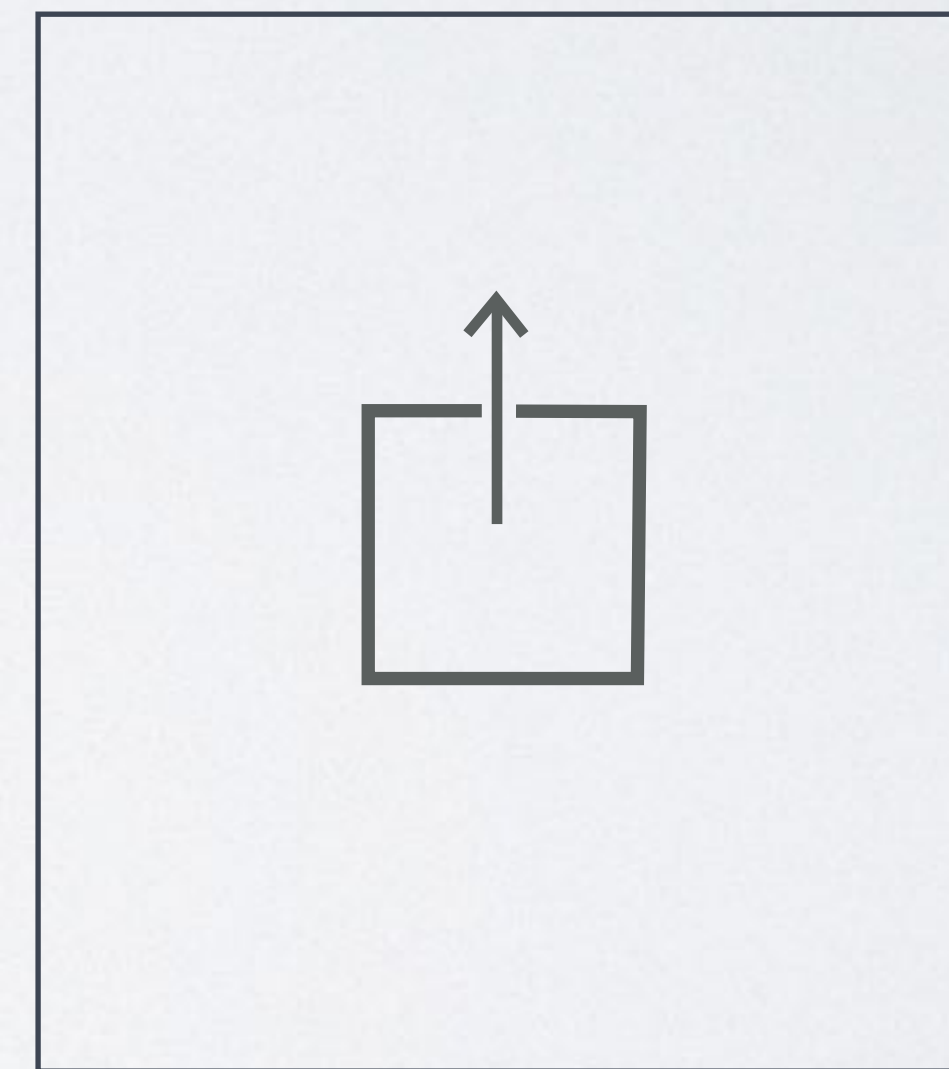
Explorer



Reconstituer



Partager



IMPORTER LES DONNÉES

|

The screenshot shows a Jupyter Notebook titled "1 (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The notebook content is in French and describes the first step of data import: using Pandas' `read_csv()` function and `head()` to inspect data.

Importer les données

La première étape consiste à importer les données. On se sert de Pandas et de la fonction `read_csv()`. La fonction `.head()` permet de visualiser les premières lignes du jeu de données.

```
In [23]: 1 import pandas as pd
          2 import seaborn as sns

In [46]: 1 filename = 'Penn_State_PA_2012.gzip'
          2 df = pd.read_csv(filename, encoding='utf-8', compression='gzip', index_
          3 df.index = df.index.tz_localize('UTC').tz_convert('US/Eastern')
          4 df.head()
```

Out[46]:

| | Year | Day number | Month | Day | Hour | Minute | ct | zen | dw_solar | dw_solar_Flag | ... |
|---------------------------|------|---------------|-------|-----|------|--------|-------|--------|----------|---------------|-----|
| timestamp | | | | | | | | | | | |
| 2012-01-01 00:00:00-05:00 | 2012 | 1 | 1 | 1 | 5 | 1 | 5.017 | 162.07 | -1.5 | 0 | ... |
| 2012-01-01 00:01:00-05:00 | 2012 | 1 | 1 | 1 | 5 | 2 | 5.033 | 162.10 | -1.5 | 0 | ... |
| 2012-01-01 00:02:00-05:00 | 2012 | 1 | 1 | 1 | 5 | 3 | 5.050 | 162.14 | -1.6 | 0 | ... |
| 2012-01-01 00:03:00-05:00 | 2012 | 1 | 1 | 1 | 5 | 4 | 5.067 | 162.17 | -1.6 | 0 | ... |
| 2012-01-01 00:04:00-05:00 | 2012 | 1 | 1 | 1 | 5 | 5 | 5.083 | 162.19 | -1.6 | 0 | ... |

5 rows x 48 columns

In []: 1

VISUALISER LES DONNÉES

2

localhost

jupyter 2 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Visualiser les données

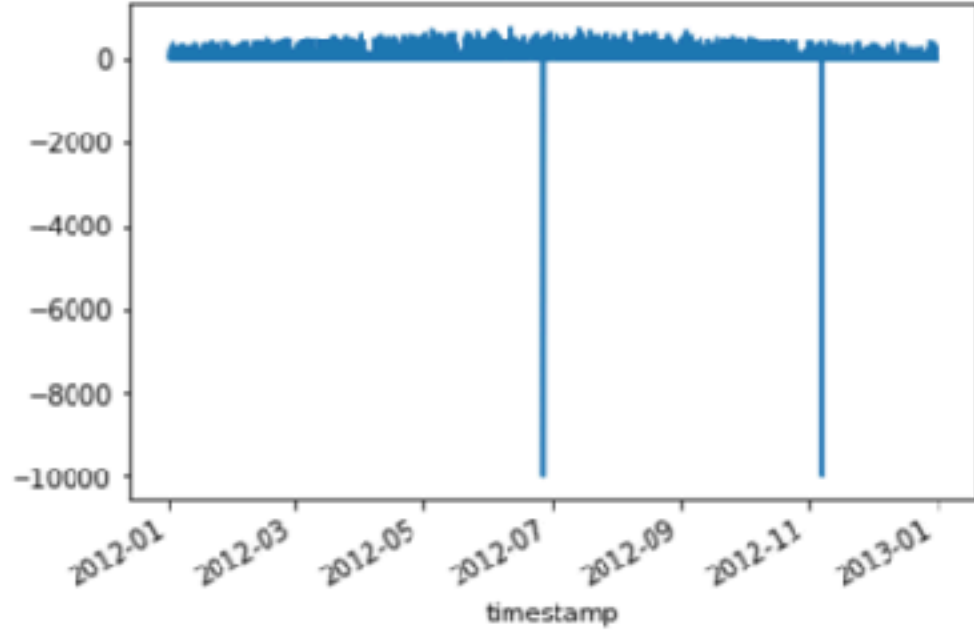
Très rapidement, on peut visualiser les données en appelant la fonction `plot()`

```
In [1]: 1 import pandas as pd
        2 %matplotlib inline

In [9]: 1 filename = 'Penn_State_PA_2012.gzip'
        2 df = pd.read_csv(filename, encoding='utf-8', compression='gzip', index_
        3 df.index = df.index.tz_localize('UTC').tz_convert('US/Eastern')

In [10]: 1 df.diffuse.plot()
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1994f85c0>




On remarque que certaines données sont aberrantes (valeurs -9999.99). Sont le fichier de descriptions des données SURFRAD, elles correspondent à des données manquantes.

Voyons un peu plus en détail à quel moment de l'année elles interviennent.

```
In [11]: 1 df['2012-06'].diffuse.plot()
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x11e742f60>



NETTOYER LES DONNÉES

3

localhost 3

jupyter 3 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

Nettoyer les données

Pour nettoyer le jeux données, nous pouvons tout simplement éliminer les valeurs abbérantes. Chaque colonne du jeux de données contient 'Flag' qui indique si la données est bonnes ou pas. Nous pouvons nous servir de cette information pour nettoyer tout le jeu de données.

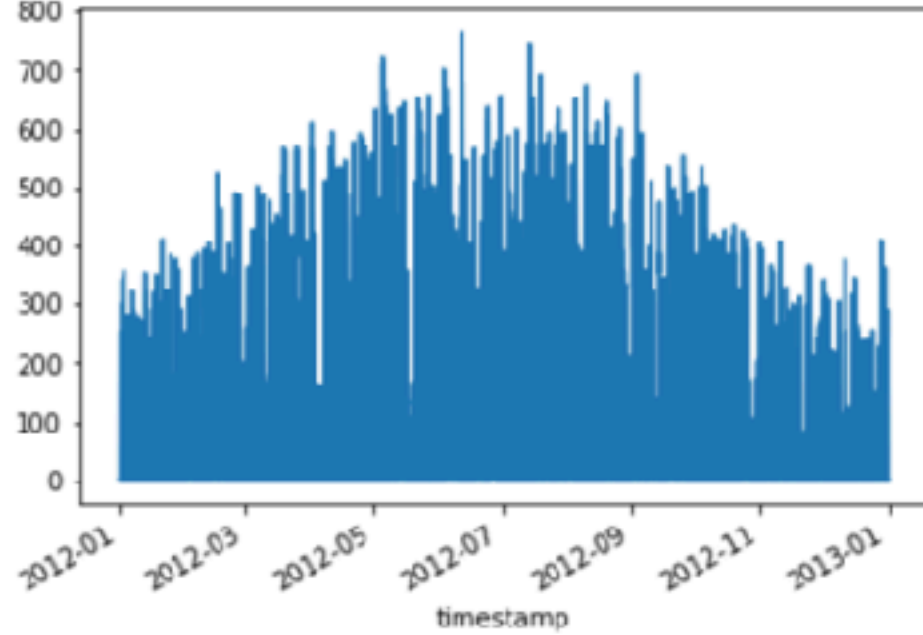
```
In [6]: 1 import pandas as pd
2 import numpy as np
3 %matplotlib inline
```

```
In [4]: 1 filename = 'Penn_State_PA_2012.zip'
2 df = pd.read_csv(filename, encoding='utf-8', compression='gzip', index_
3 df.index = df.index.tz_localize('UTC').tz_convert('US/Eastern')
```

```
In [7]: 1 col_flag = df.columns[df.columns.str.endswith('_Flag')]
2 col_noflag = col_flag.str.replace('_Flag', '')
3 for i in col_noflag.values:
4     df.loc[df[i + '_Flag'] > 0, i] = np.nan
```

```
In [8]: 1 df.diffuse.plot()
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x10a1700b8>
```

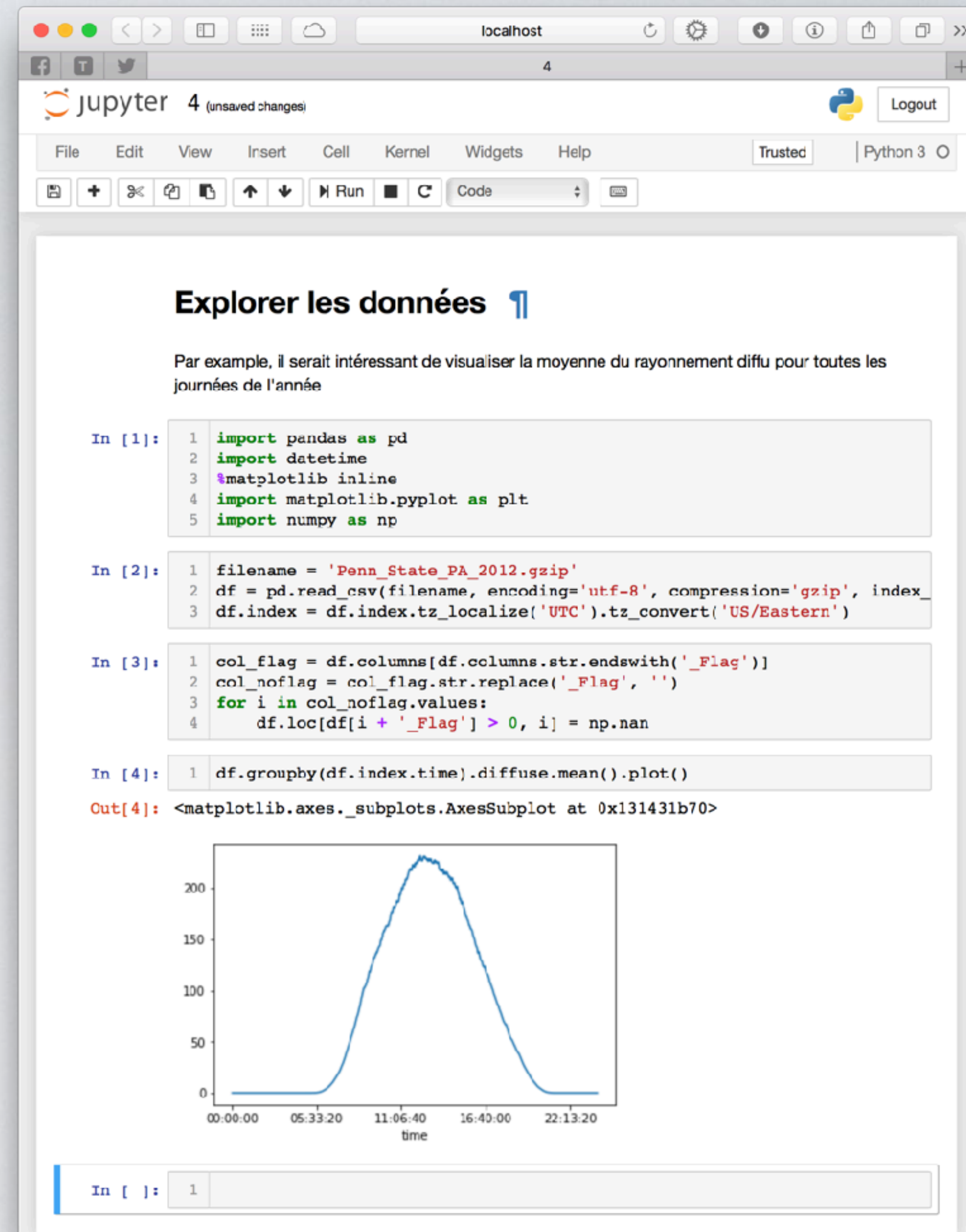


Les valeurs abbérantes ont été enlevées et remplacées par NaN.

```
In [9]: 1 df['2012-06'].diffuse.plot()
```


EXPLORER LES DONNÉES

4



EXPLORER LES DONNÉES (SUITE)

5

localhost 5

jupyter 5 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

Explorer les données (Suite)

Il serait également intéressant de visualiser ces données pour chaque jour de l'année dans son ensemble

```
In [1]: 1 import pandas as pd
2 %matplotlib inline
3 import numpy as np
```

```
In [2]: 1 filename = 'Penn_State_PA_2012.gzip'
2 df = pd.read_csv(filename, encoding='utf-8', compression='gzip', index_
3 df.index = df.index.tz_localize('UTC').tz_convert('US/Eastern')
```

```
In [3]: 1 col_flag = df.columns[df.columns.str.endswith('_Flag')]
2 col_noflag = col_flag.str.replace('_Flag', '')
3 for i in col_noflag.values:
4     df.loc[df[i + '_Flag'] > 0, i] = np.nan
```

```
In [4]: 1 pivoted = df.pivot_table('diffuse', index=df.index.time, columns=df.index
2 pivoted.head()
```

Out[4]:

| | 2012-01-01 | 2012-01-02 | 2012-01-03 | 2012-01-04 | 2012-01-05 | 2012-01-06 | 2012-01-07 | 2012-01-08 | 2012-01-09 | 2012-01-10 | ... | 2012-12-22 | 2012-12-23 | 2012-12-24 |
|----------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|-----|------------|------------|------------|
| 00:00:00 | 0.1 | 0.6 | 0.5 | 0.2 | 0.4 | 0.2 | 0.5 | 0.1 | 0.2 | 0.1 | ... | -0.2 | -0.2 | -0.2 |
| 00:01:00 | 0.1 | 0.7 | 0.5 | 0.2 | 0.4 | 0.1 | 0.4 | 0.1 | 0.2 | 0.2 | ... | -0.2 | -0.2 | -0.2 |
| 00:02:00 | 0.1 | 0.7 | 0.5 | 0.2 | 0.2 | 0.1 | 0.4 | 0.2 | 0.2 | 0.2 | ... | -0.2 | -0.2 | -0.2 |
| 00:03:00 | 0.1 | 0.7 | 0.6 | 0.3 | 0.2 | 0.1 | 0.4 | 0.1 | 0.2 | 0.2 | ... | -0.2 | -0.3 | -0.3 |
| 00:04:00 | 0.0 | 0.7 | 0.8 | 0.3 | 0.2 | 0.1 | 0.3 | 0.1 | 0.2 | 0.2 | ... | -0.2 | -0.2 | -0.2 |

5 rows x 366 columns

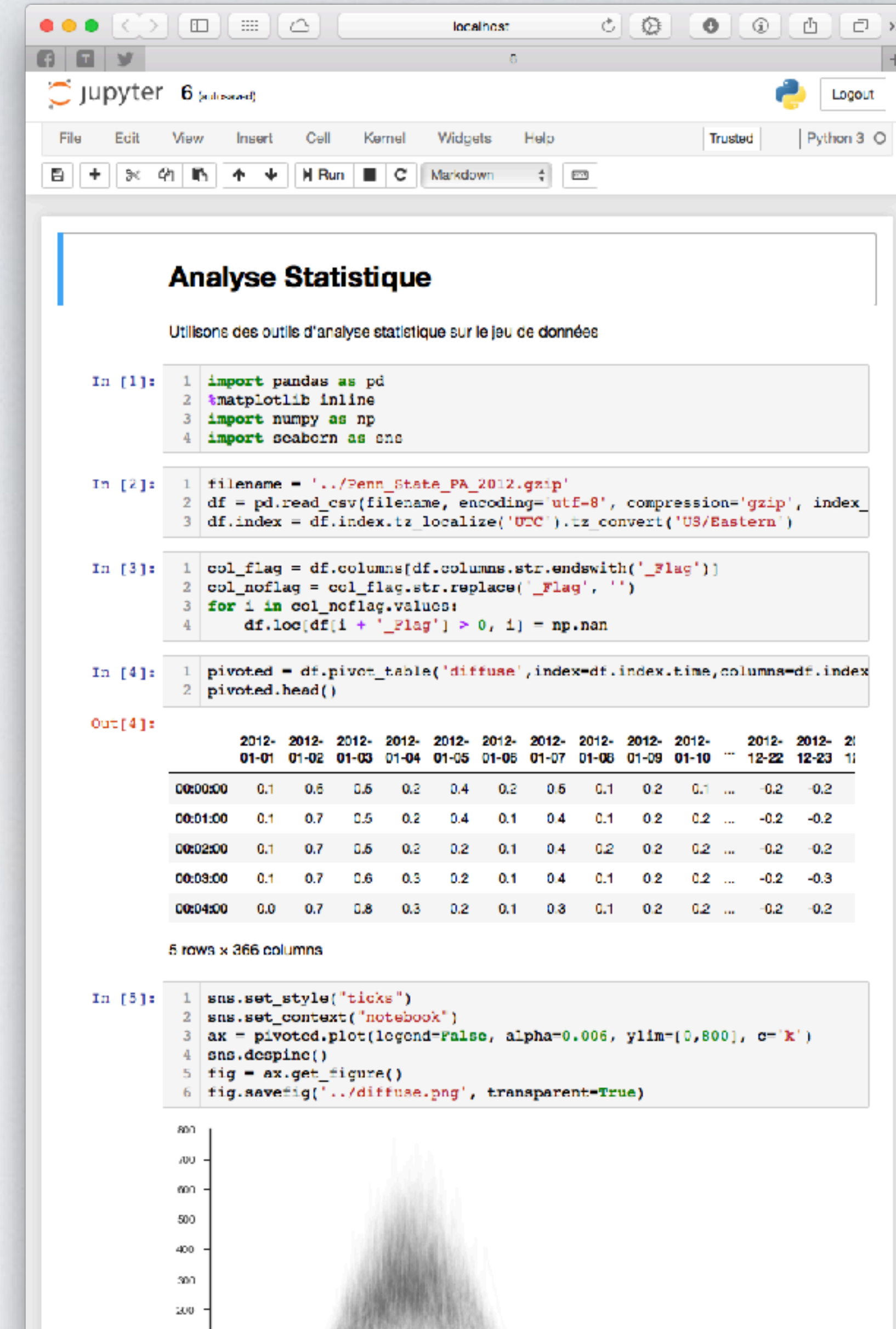
```
In [5]: 1 pivoted.plot(legend=False, alpha=0.006, c='k', title='Rayonnement diffu
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x118c66240>

Rayonnement diffus journalier

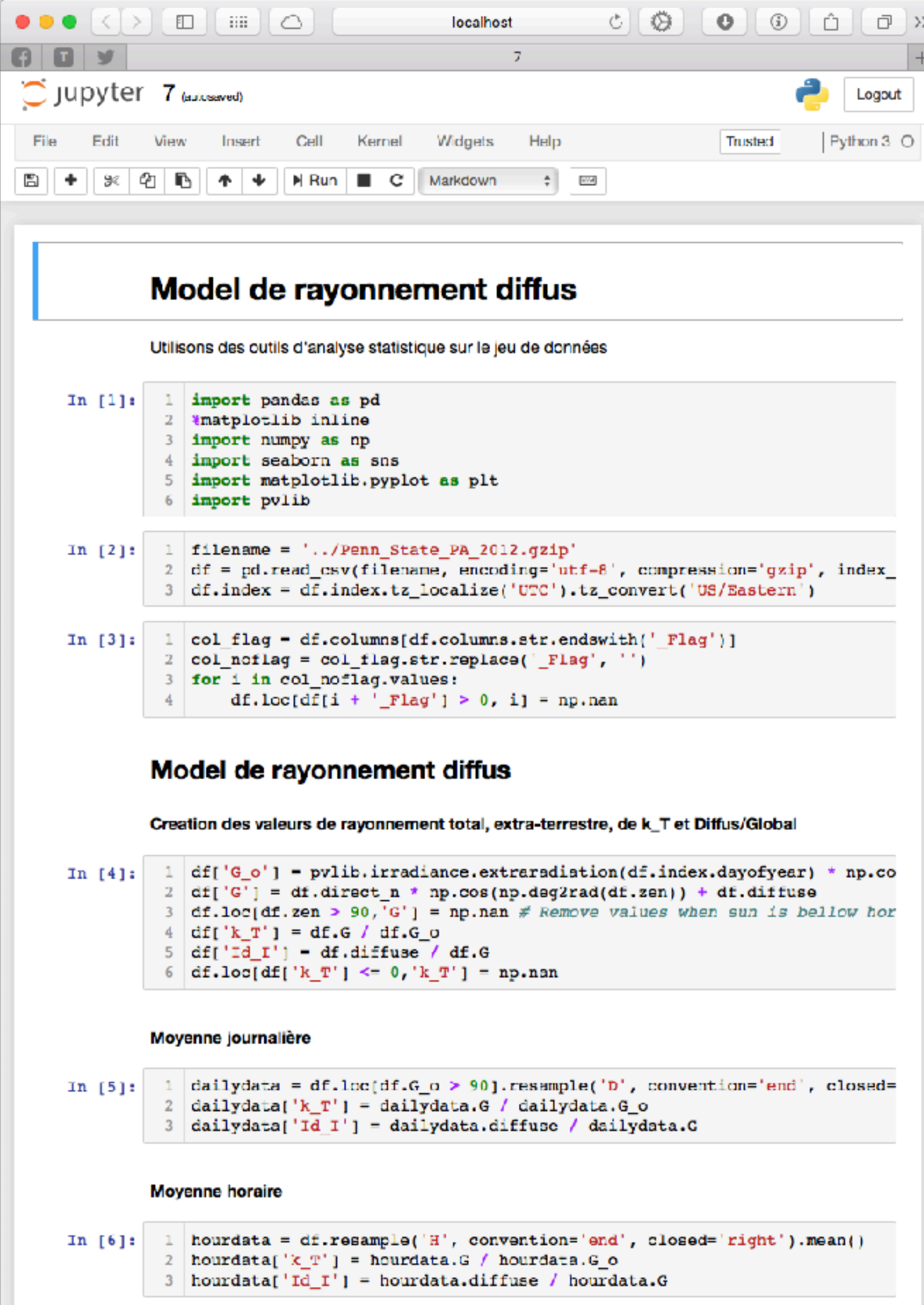
ANALYSE STATISTIQUE

6



MODÈLE DE RAYONNEMENT DIFFUS

7



The screenshot shows a Jupyter Notebook interface with a browser window at localhost. The notebook has a title bar with 'jupyter 7 (auto-saved)' and a 'Logout' button. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar shows icons for file operations and a 'Run' button. The notebook content is as follows:

Model de rayonnement diffus

Utilisons des outils d'analyse statistique sur le jeu de données

```
In [1]: 1 import pandas as pd
2 import matplotlib inline
3 import numpy as np
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 import pvlib
```

```
In [2]: 1 filename = '../Penn_State_PA_2012.gzip'
2 df = pd.read_csv(filename, encoding='utf-8', compression='gzip', index_
3 df.index = df.index.tz_localize('UTC').tz_convert('US/Eastern')
```

```
In [3]: 1 col_flag = df.columns[df.columns.str.endswith('_Flag')]
2 col_noflag = col_flag.str.replace('_Flag', '')
3 for i in col_noflag.values:
4     df.loc[df[i + '_Flag'] > 0, i] = np.nan
```

Model de rayonnement diffus

Creation des valeurs de rayonnement total, extra-terrestre, de k_T et Diffus/Global

```
In [4]: 1 df['G_o'] = pvlib.irradiance.extraterrestrial(df.index.dayofyear) * np.co
2 df['G'] = df.direct_n * np.cos(np.deg2rad(df.zen)) + df.diffuse
3 df.loc[df.zen > 90, 'G'] = np.nan # Remove values when sun is below hor
4 df['k_T'] = df.G / df.G_o
5 df['Id_I'] = df.diffuse / df.G
6 df.loc[df['k_T'] <= 0, 'k_T'] = np.nan
```

Moyenne journalière

```
In [5]: 1 dailydata = df.loc[df.G_o > 90].resample('D', convention='end', closed=
2 dailydata['k_T'] = dailydata.G / dailydata.G_o
3 dailydata['Id_I'] = dailydata.diffuse / dailydata.G
```

Moyenne horaire

```
In [6]: 1 hourdata = df.resample('H', convention='end', closed='right').mean()
2 hourdata['k_T'] = hourdata.G / hourdata.G_o
3 hourdata['Id_I'] = hourdata.diffuse / hourdata.G
```

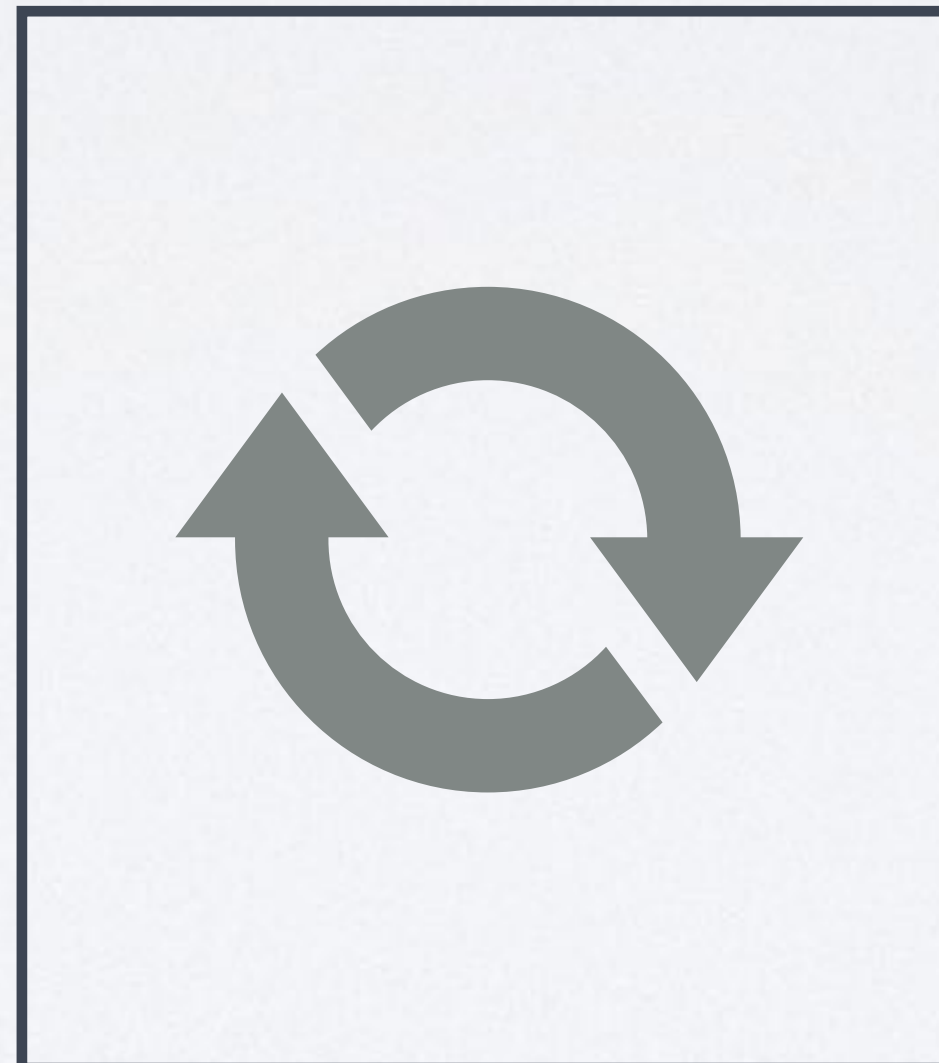

PYTHON

Nous aide à être plus efficace dans notre travail de chercheur
en tirant avantage des outils disponibles

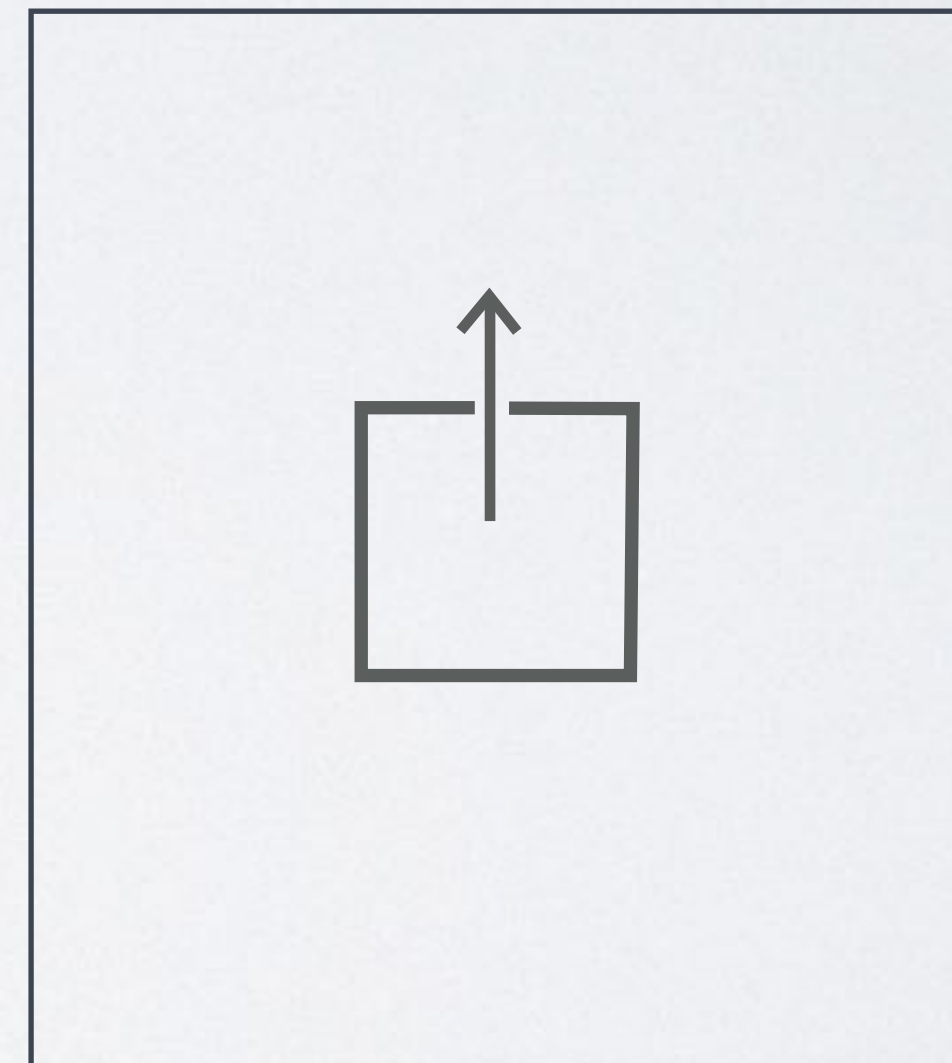
Explorer



Reconstituer



Partager



RÉPÉTER

S'assurer que le code est reproductible

simplement cliquer sur :

Kernel → Restart & Run All

The screenshot displays a Jupyter Notebook titled "Reproductibilité" with a "Last Checkpoint: 5 hours ago (unsaved changes)" status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, cell execution, and zooming. The notebook contains several code cells:

- Cell 1:** Imports necessary libraries: `import pandas as pd`, `import matplotlib inline`, `import numpy as np`, `import seaborn as sns`, `import matplotlib.pyplot as plt`, `import pulp`, and `from pysurfrad.data import get_surfrad_data`.
- Cell 2:** Reads a CSV file: `filename = 'Penn State PA 2012.gzip'`, `df = pd.read_csv(filename, encoding='utf-8', compression='gzip', index_col='timestamp', parse_dates=True)`, and converts the index to UTC: `df.index = df.index.tz_localize('UTC').tz_convert('US/Eastern')`.
- Cell 3:** Processes the data: `col_flag = df.columns[df.columns.str.endswith('_Flag')]`, `col_noflag = col_flag.str.replace('_Flag', '')`, and `df.loc[df[col_flag] > 0, col_noflag] = np.nan`.
- Cell 4:** A text box explaining that adding a question mark to the end of a function name provides help: "En ajoutant le caractère ? à la fin d'une fonction, on obtient le text d'aide".
- Cell 5:** Executes `get_surfrad_data?` to view the function's signature and documentation.
- Cell 6:** A new empty cell with the prompt `In []:`.

Below the code cells, a detailed documentation window for the `get_surfrad_data` function is visible. It includes the function signature: `get_surfrad_data(station_name='Boulder_CO', year='2013', timezone=None, force_reload=False, force_redownload=False, reindex=True, clean_flagged_values=False)`. The documentation describes the function's purpose: "Load, parse and combine the surfrad data for a particular Year". It lists parameters such as `station_name` (Name of weather station), `year` (chosen year), `force_reload` (bool, optional), `force_redownload` (bool, optional), `timezone` (string, optional), `reindex` (bool, optional), and `clean_flagged_values` (bool, optional). It also specifies the return type: `pandas.DataFrame`.

RÉPÉTER

Créer un « python package »

Une librairie de fonction que l'on peut importer dans les premières lignes du carnet

```
data.py
1 import os
2 import glob
3
4 from datetime import datetime
5 from tqdm import tqdm_notebook
6 from tqdm import tqdm
7 import pandas as pd
8 import numpy as np
9 import pytz
10 import time
11
12 STATIONNAME = 'Boulder_CO'
13 YEAR = '2013'
14
15
16 def get_surfrad_data(station_name=STATIONNAME, year=YEAR, timezone=None, force_reload=False,
17 force_redownload=False, reindex=True, clean_flagged_values=False):
18     """Load, parse and combine the surfrad data for a particular Year
19
20     Parameters
21     -----
22     :param station_name:
23         Name of weather station from which to retrieve data
24     :param year:
25         chosen year
26     :param force_reload: bool (optional)
27         if True, force reload of data
28     :param timezone: string (optional)
29         Any of the tz database time zones (eg. 'America/Denver')
30     :param reindex: bool (optional)
31         reindexes the dataframe to a full year of data (eg. 525600 lines if after 2009 and 175200 if before 2009)
32     :param clean_flagged_values: bool (optional)
33         if flag > 0, replace with NaN for all columns with flags.
34
35     Returns
36     -----
37     :returns: pandas.DataFrame
38         The surfrad data
39
40     More INFO
41     -----
42
43     Parameter      Parameter      Parameter      Parameter
44     -----
45     Year            dt              direct_n       dw_casetemp
46     Day number      zen              direct_n_Flag dw_casetemp_Flag
47     Month           dw_solar        diffuse        dw_domtemp
48     Day             dw_solar_Flag  diffuse_Flag   dw_domtemp_Flag
49     Hour            uw_solar        dw_ir          uw_ir
50     Minute          uw_solar_Flag  dw_ir_Flag     uw_ir_Flag
51     uw_casetemp      par              totalnet       windspd
52     uw_casetemp_Flag par_Flag        totalnet_Flag  windspd_Flag
53     uw_domtemp       netsolar        temp           winddir
54     uw_domtemp_Flag netsolar_Flag  temp_Flag      winddir_Flag
55     uvb              netir           rh              pressure
56     uvb_Flag         netir_Flag      rh_Flag         pressure_Flag
57
58
59     """
60     names = ['Year', 'Day number', 'Month', 'Day', 'Hour', 'Minute', 'dt', 'zen', 'dw_solar',
61             'dw_solar_Flag', 'uw_solar', 'uw_solar_Flag', 'direct_n', 'direct_n_Flag', 'diffuse', 'diffuse_Flag',
62             'dw_ir', 'dw_ir_Flag', 'dw_casetemp', 'dw_casetemp_Flag', 'dw_domtemp', 'dw_domtemp_Flag', 'uw_ir',
63             'uw_ir_Flag', 'uw_casetemp', 'uw_casetemp_Flag', 'uw_domtemp', 'uw_domtemp_Flag', 'uvb', 'uvb_Flag']
64
65     pysurfrad/data.py 101
```

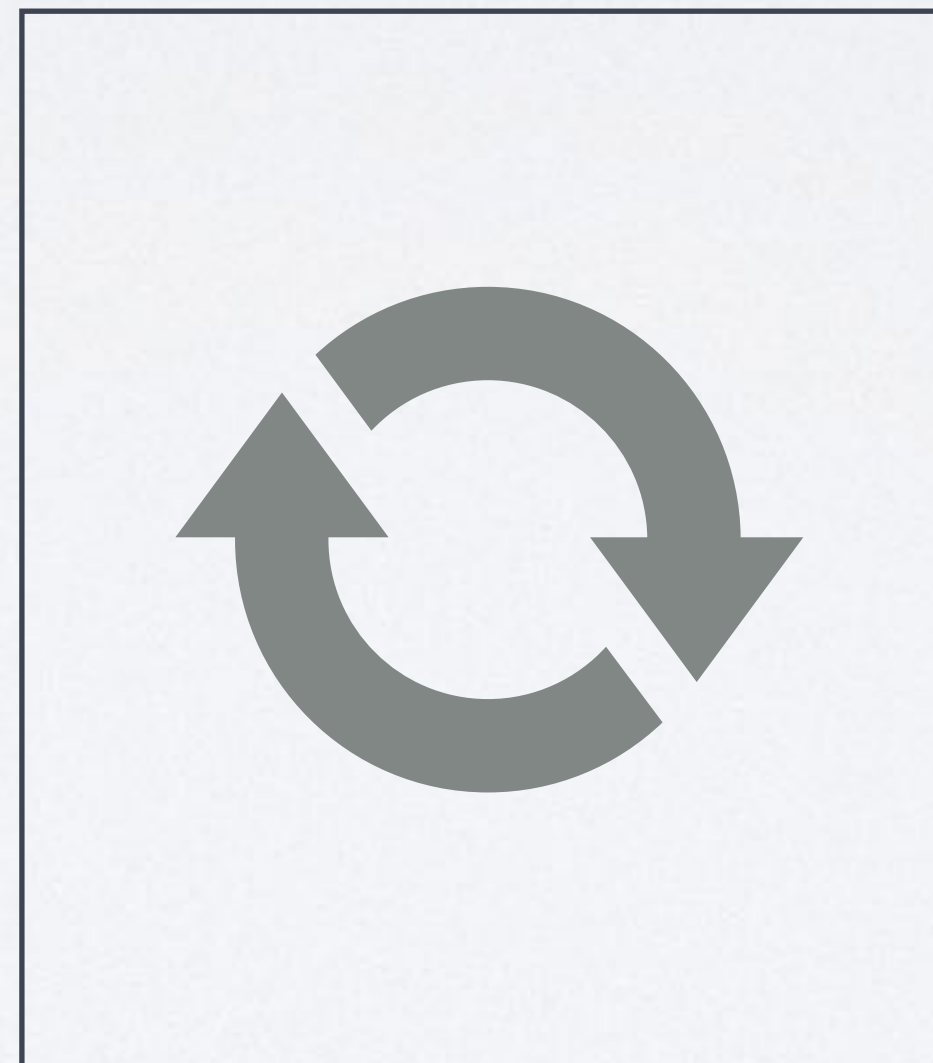

PYTHON

Nous aide à être plus efficace dans notre travail de chercheur
en tirant avantage des outils disponibles

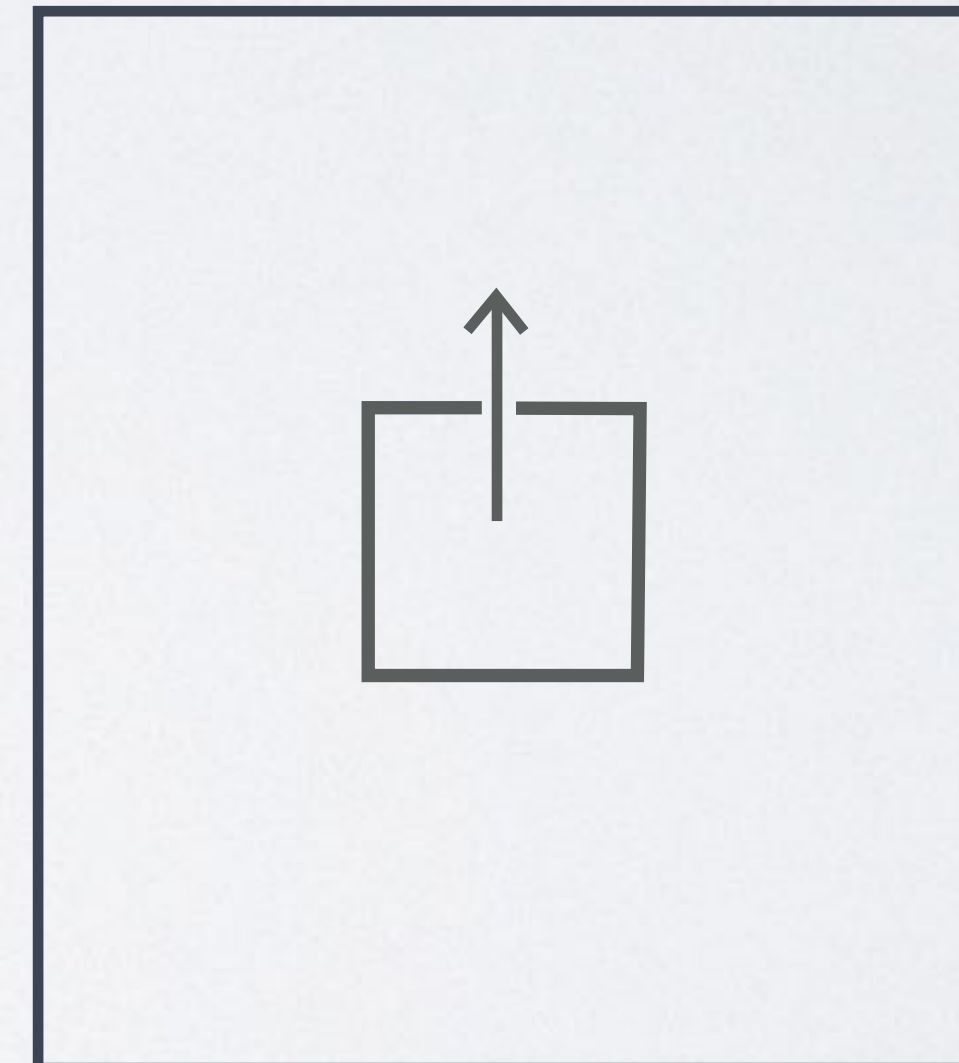
Explorer



Reconstituer



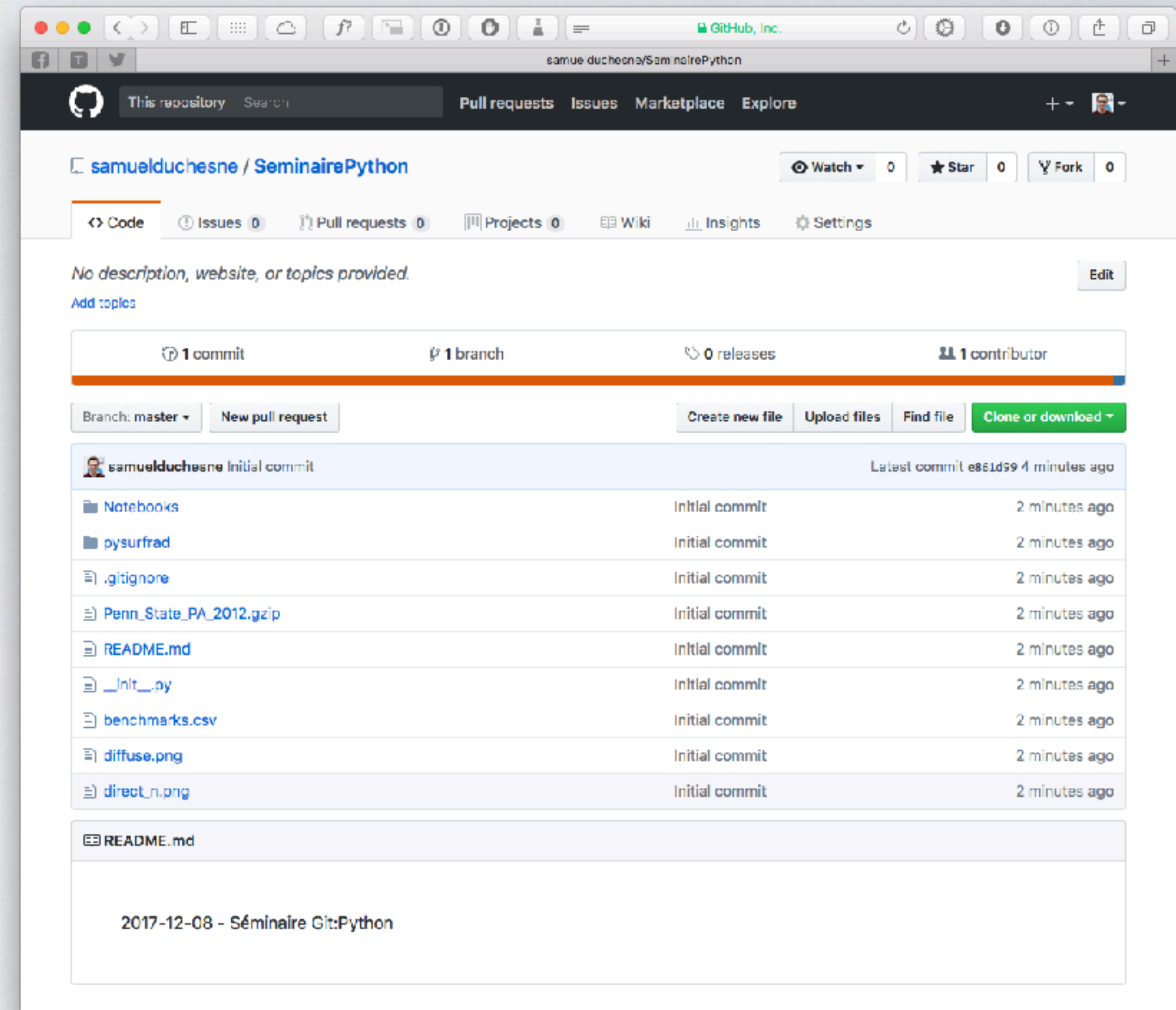
Partager



PARTAGER

Les « Notebooks » sont facilement
partagés en lignes

github, nbviewer, etc.

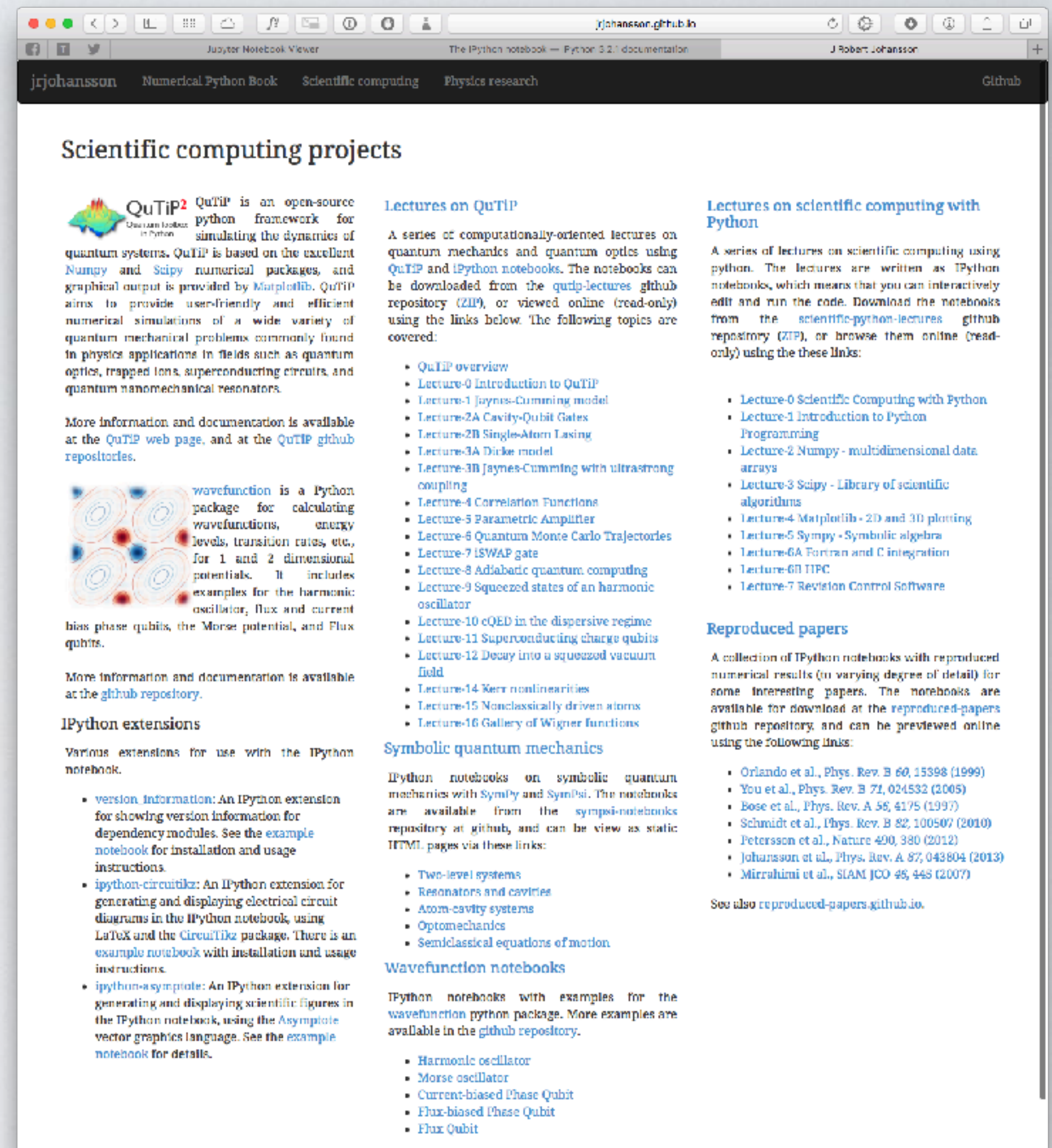


IPython Notebook

Pour des notes de cours en ligne

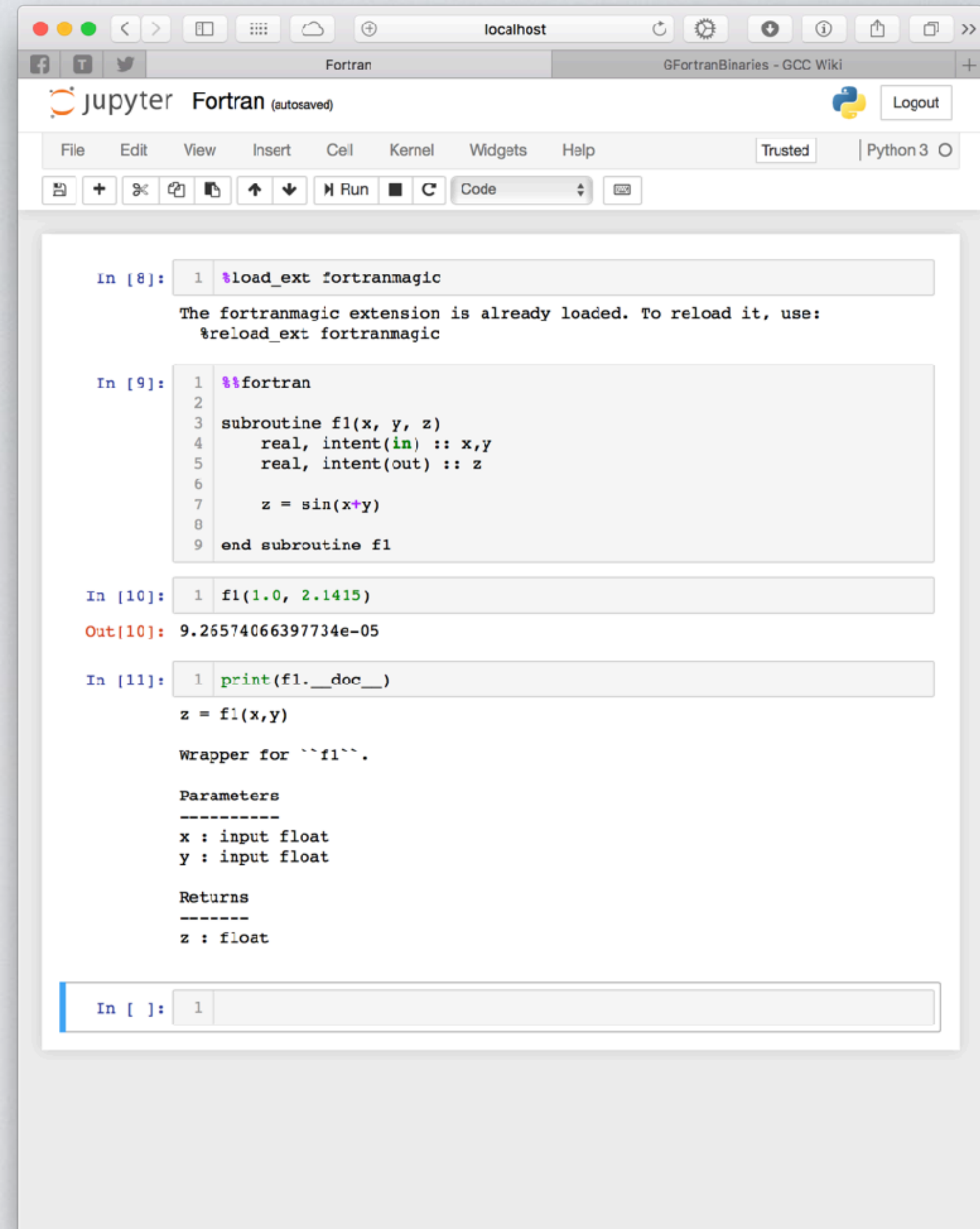
J Robert Johansson, Ph.D.

physique des solides théoriques et de calcul, mécanique quantique dans les circuits électriques supraconducteurs.



FORTRAN

et oui!



The screenshot shows a Jupyter Notebook titled "Fortran (autosaved)" running on a "localhost" browser. The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help" menus. Below the menu is a toolbar with icons for file operations and a "Run" button. The notebook contains three input cells and one output cell. The first input cell (In [8]:) contains the command `%load_ext fortranmagic`, followed by a message: "The fortranmagic extension is already loaded. To reload it, use: %reload_ext fortranmagic". The second input cell (In [9]:) contains a Fortran subroutine definition: `%%fortran`, `subroutine f1(x, y, z)`, `real, intent(in) :: x,y`, `real, intent(out) :: z`, `z = sin(x+y)`, and `end subroutine f1`. The third input cell (In [10]:) contains the function call `f1(1.0, 2.1415)`, which has produced an output (Out[10]:) of `9.26574066397734e-05`. The fourth input cell (In [11]:) contains `print(f1.__doc__)`, which has produced a docstring output: `z = f1(x,y)`, `Wrapper for ``f1``.`, `Parameters`, `-----`, `x : input float`, `y : input float`, `Returns`, `-----`, and `z : float`. The bottom of the notebook shows an empty input cell (In []:).

```
In [8]: 1 %load_ext fortranmagic
The fortranmagic extension is already loaded. To reload it, use:
%reload_ext fortranmagic

In [9]: 1 %%fortran
2
3 subroutine f1(x, y, z)
4     real, intent(in) :: x,y
5     real, intent(out) :: z
6
7     z = sin(x+y)
8
9 end subroutine f1

In [10]: 1 f1(1.0, 2.1415)
Out[10]: 9.26574066397734e-05

In [11]: 1 print(f1.__doc__)
z = f1(x,y)

Wrapper for ``f1``.

Parameters
-----
x : input float
y : input float

Returns
-----
z : float

In [ ]: 1
```


RESSOURCES

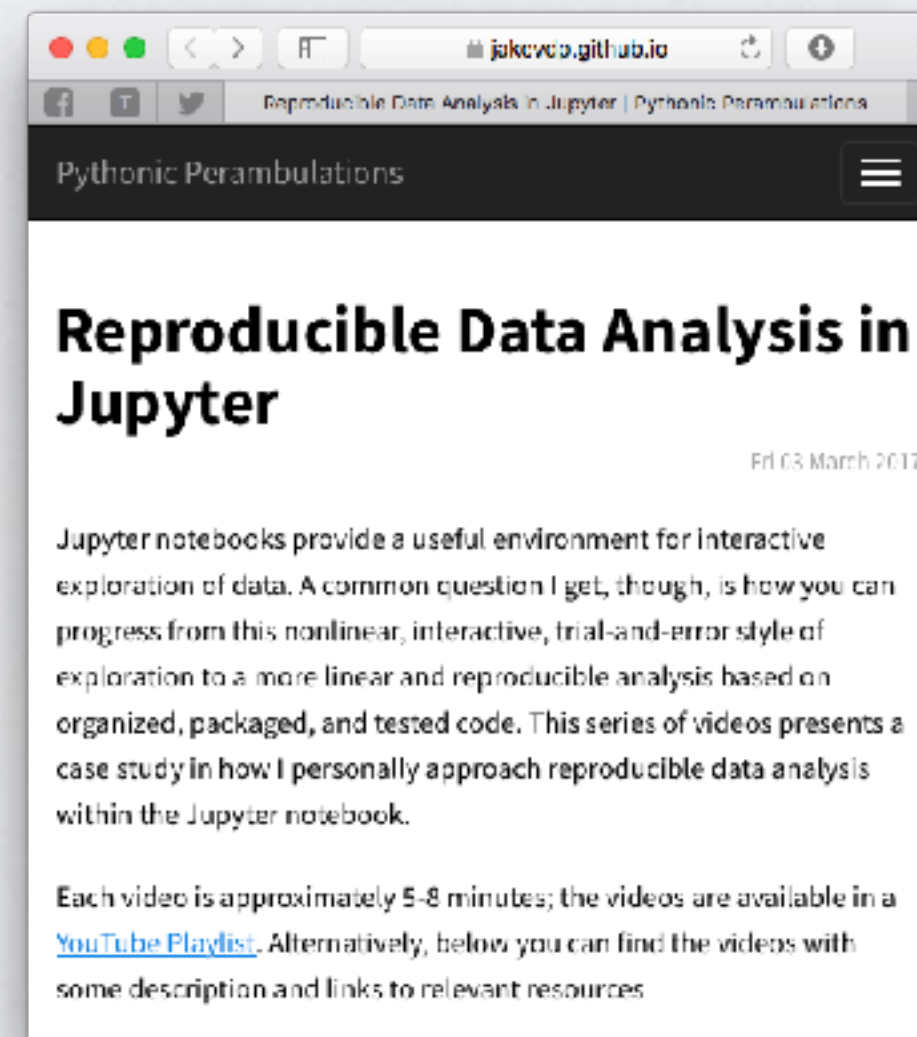
En ligne

Pour installer Python



Anaconda
Installation facile de Python

Pour apprendre



**Reproducible Data Analysis in
Jupyter**
Jake VanderPlas

Pour se rappeler

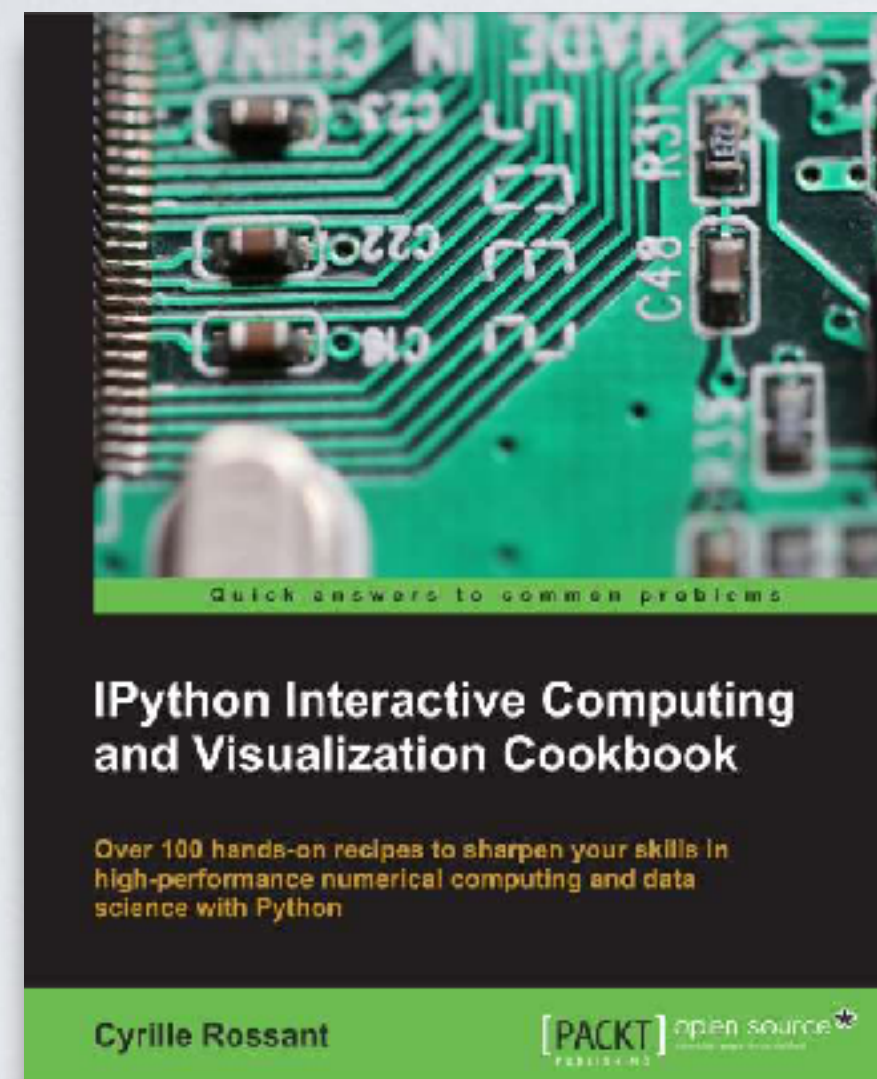


Cheat Sheet

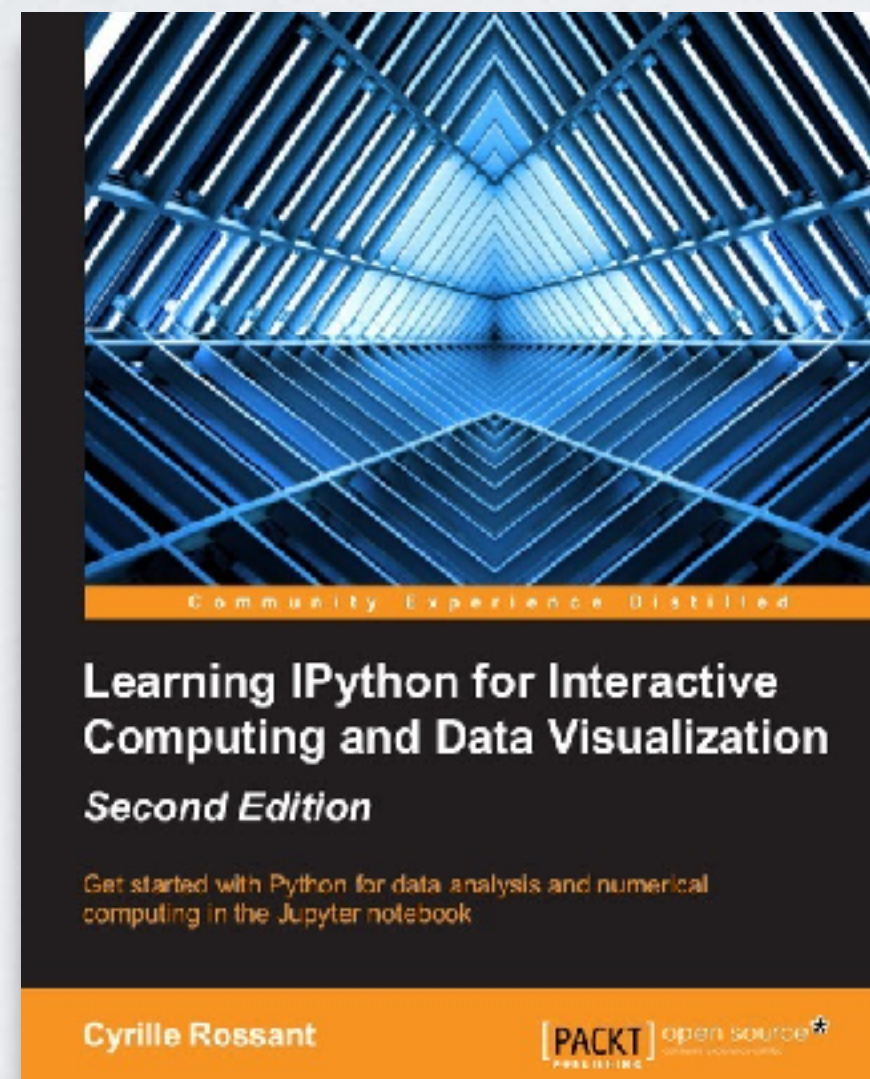
Python Pandas Cheat Sheet
Aide mémoire pour Pandas

RESSOURCES

Livres



IPython Interactive Computing and Visualization Cookbook
Cyrille Rossant

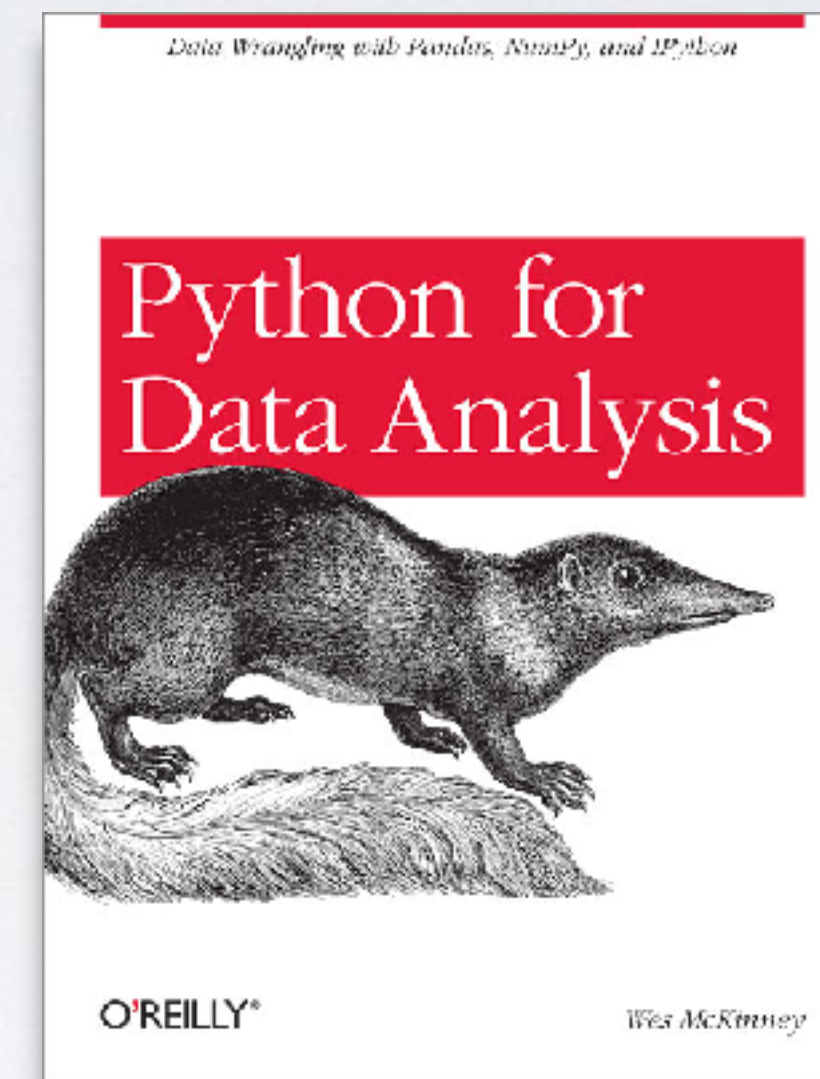


Learning IPython for Interactive Computing and Data Visualization, second edition
Cyrille Rossant

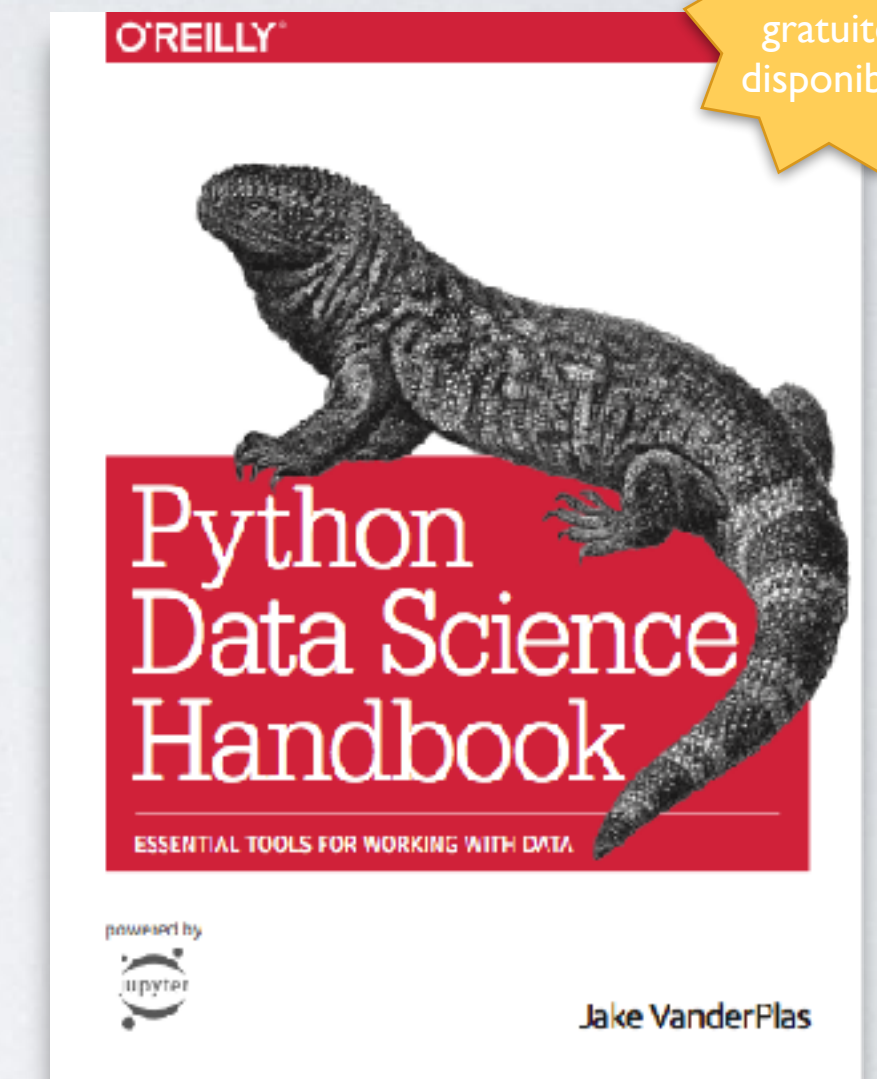


Version gratuite disponible

A Whirlwind Tour of Python
Jake VanderPlas



Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython
William McKinney



Version gratuite disponible

Python Data Science Handbook
Jake VanderPlas