01-introduction

August 21, 2020

1 Installation

Pour construire note site, nous avez besoin d'une distribution Python. Miniconda est une solution légère permettant d'installer tous les packages nécessaires.

Une fois installer miniconda en ayant suivi les instructions, ouvrez un terminal sur Linux/Macos ou un anaconda prompt sur Windows.

1.1 Environement conda

```
git clone https://gitlab.math.unistra.fr/navaro/agrocampus
# # cd agrocampus
conda env create
```

documentation.

L'environnement est créé a partir d'un fichier nommé `environment.yml` qui contient la liste des packages dont nous avons besoin.

1.2 Activer l'environment conda

Lorsque vous activer l'environement conda, votre configuration du terminal est modifiée et la version de python qui sera disponible sera celle ou tous nos packages seront installés.

Cette opération doit être renouvelée à chaque fois que l'on ouvre un nouveau terminal

1.3 Jupyter Notebook

Le calepin Jupyter est un outil de rédaction qui permet de rendre lisible une analyse mathématique. Il permet de rassembler du code informatique, du texte, des images et des formules mathématiques dans un seul document. Le code informatique est exécutable, le calepin Jupyter consitue un excellent support pour travailler et surtout pour partager.

Jupyter comporte de nombreuses extensions et supportent un nombre important de langages. De plus c'est un logiciel libre et ouvert totalement gratuit qui fonctionne sur tous les systèmes d'exploitation existants.

Jupyter est un acronyme des 3 langages supportés à l'origine du projet: **JU**lia, **PYT**hon, et **R**

1.4 Installation with conda

```
conda install -c conda-forge jupyter
```

1.5 with pip

```
pip install jupyter
```

1.6 Installation de packages Python dans Jupyter

1.6.1 Avec conda

Pour installer `numpy` du canal *conda-forge*

```
# # %conda install -c conda-forge numpy
```

1.6.2 Avec pip

```
# # %pip install numpy
```

1.7 Raccoucis clavier

- Pour afficher les commandes disponibles: `Cmd + Shift + P`
- `Esc` permet de basculer en mode "commande" et vous pouvez naviguer dans le document avec les flèches de votre clavier.
- En mode "commande":
  - `A` permet d'insérer une nouvelle cellule **au dessus** de la cellule active
  - `B` permet d'insérer une nouvelle cellule **en dessous** de la cellule active
  - `M` bacule en mode texte (markdown), `Y` pour revenir au code
  - `D + D` en pressant deux fois ce caractère vous effacez la cellule courante

1.8 Documentation

- Shift + Tab donne accès à la documentation des focntions

[1]:
```
dict
```

[1]:
```
dict
```

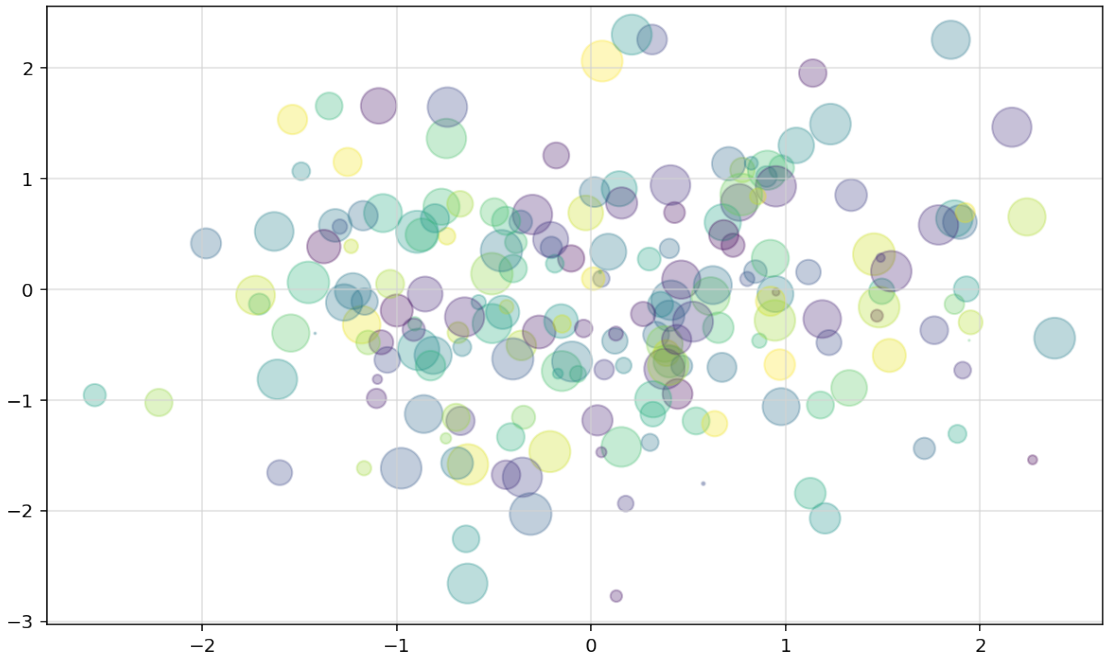
1.9 Graphique (en python)

la première ligne de la cellule suivante permet de tracer vos graphiques juste après l"éxécution de la céllule.

[2]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
import matplotlib.pyplot as plt
import numpy as np
```

[3]:
```
plt.rcParams['figure.figsize'] = (10,6)
fig, ax = plt.subplots()
np.random.seed(0)
x, y = np.random.normal(size=(2, 200))
color, size = np.random.random((2, 200))
```

```
ax.scatter(x, y, c=color, s=500 * size, alpha=0.3)
ax.grid(color='lightgray', alpha=0.7)
```



1.10 Les commandes magiques

[4]:
```
%lsmagic
```

[4]:
```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark
%cat  %cd  %clear  %colors  %conda  %config  %connect_info  %cp  %debug  %dhist
%dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %history  %killbgscripts
%ldir  %less  %lf  %lk  %ll  %load  %load_ext  %loadpy  %logoff  %logon
%logstart  %logstate  %logstop  %ls  %lsmagic  %lx  %macro  %magic  %man
%matplotlib  %mkdir  %more  %mv  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc
%pfile  %pinfo  %pinfo2  %pip  %popd  %pprint  %precision  %prun  %psearch
%psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %rehashx
%reload_ext  %rep  %rerun  %reset  %reset_selective  %rm  %rmdir  %run  %save
%sc  %set_env  %store  %sx  %system  %tb  %time  %timeit  %unalias  %unload_ext
%who  %who_ls  %whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%javascript
%%js  %%latex  %%markdown  %%perl  %%prun  %%pypy  %%python  %%python2
%%python3  %%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%timeit
%%writefile
```

```
Automagic is ON, % prefix IS NOT needed for line magics.
```

[5]:
```
%ls
```

```
01-introduction.ipynb    07-jupytext.ipynb        fibonacci.py
02-gitbasics.ipynb       08-exemple-julia.ipynb   images/
03-markdown.ipynb        09-exemple-python.ipynb  ipython_cell_input.py
04-docker.ipynb          10-exemple-r.ipynb       sample.txt
06-binder.ipynb          11-github-action.ipynb   src/
```

[6]:
```
%%file sample.txt

write the cell content to the file sample.txt.
The file is created when you run this cell.
```

```
Overwriting sample.txt
```

[7]:
```
%cat sample.txt
```

```
write the cell content to the file sample.txt.
The file is created when you run this cell.
```

[8]:
```
%%file fibonacci.py

f1, f2 = 1, 1
for n in range(10):
    print(f1, end=',')
    f1, f2 = f2, f1+f2
```

```
Overwriting fibonacci.py
```

[9]:
```
%cat fibonacci.py
```

```
f1, f2 = 1, 1
for n in range(10):
    print(f1, end=',')
    f1, f2 = f2, f1+f2
```

[10]:
```
%run fibonacci.py
```

```
1,1,2,3,5,8,13,21,34,55,
<Figure size 720x432 with 0 Axes>
```

[11]:
```
# %load fibonacci.py

f1, f2 = 1, 1
for n in range(10):
    print(f1, end=',')
    f1, f2 = f2, f1+f2
```

```
1,1,2,3,5,8,13,21,34,55,
```

[12]:
```
%%time
f1, f2 = 1, 1
for n in range(10):
    print(f1, end=',')
    f1, f2 = f2, f1+f2
print()
```

```
1,1,2,3,5,8,13,21,34,55,
CPU times: user 805 µs, sys: 445 µs, total: 1.25 ms
Wall time: 901 µs
```

[13]:
```
%who int
```

```
f1       f2       n
```

[14]:
```
import numpy as np
%timeit np.random.normal(size=100)
```

```
20.1 µs ± 2.17 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

[15]:
```
from time import sleep
def fibonacci(n):
    f1, f2 = 1, 1
    res = []
    for i in range(n):
        f1, f2 = f2, f1+f2
        res.append(f1)
    return res
```

[22]:
```
import IPython.core
IPython.core.page = print
```

[28]:
```
%prun -q -T prof.txt fibonacci(10)
```

```
*** Profile printout saved to text file 'prof.txt'.
```

[29]:
```
%cat prof.txt
```

```
         14 function calls in 0.000 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.000    0.000 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 <ipython-
input-15-fbbc9226da40>:2(fibonacci)
       10    0.000    0.000    0.000    0.000 {method 'append' of 'list'
objects}
        1    0.000    0.000    0.000    0.000 <string>:1(<module>)
        1    0.000    0.000    0.000    0.000 {method 'disable' of
'_lsprof.Profiler' objects}
```

[18]:
```
import sys
%pip install -q py-heat-magic
```

```
WARNING: You are using pip version 20.2.1; however, version 20.2.2 is
available.
You should consider upgrading via the '/usr/local/opt/python@3.8/bin/python3.8
-m pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

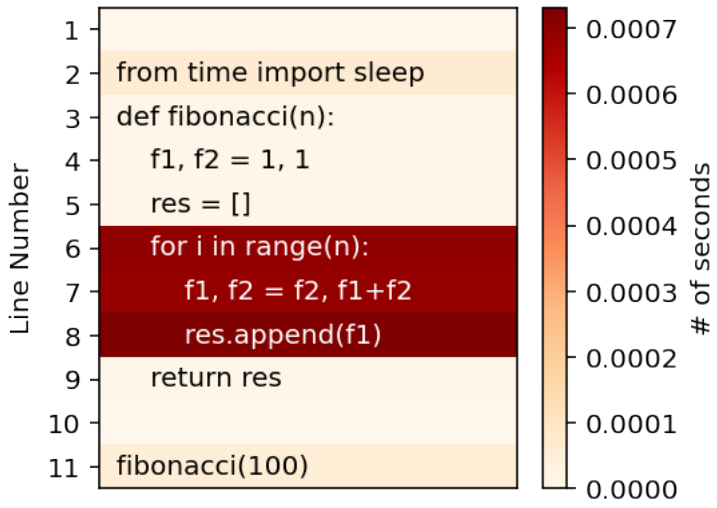
[19]:
```
%load_ext heat
```

[20]:
```
%%heat

from time import sleep
def fibonacci(n):
    f1, f2 = 1, 1
    res = []
    for i in range(n):
        f1, f2 = f2, f1+f2
        res.append(f1)
    return res

fibonacci(100)
```

```
/usr/local/lib/python3.8/site-packages/pyheat/pyheat.py:158: UserWarning:
FixedFormatter should only be used together with FixedLocator
  self.ax.set_yticklabels(row_labels, minor=False)
```



[21]:
```
from tqdm.notebook import tqdm
from time import sleep

n = 10
res = [1]

for x in tqdm(range(2, n)):
    sleep(0.2)
    for i in range(2, x):
        if (x % i) == 0:
            break
        else:
            res.append(x)
            break

res
```

```
HBox(children=(FloatProgress(value=0.0, max=8.0), HTML(value='')))
```

[21]:
```
[1, 3, 5, 7, 9]
```

from ipywidgets import interact

@interact(x=True, y=1.0) def g(x, y): return (x, y)

@interact(tau=(0.01, 0.2, 0.01)) def f(tau): plt.figure(2) t = np.linspace(0, 1, num=1000) plt.plot(t, 1 - np.exp(-t/tau), t, np.exp(-t/tau)) plt.xlim(0, 1) plt.ylim(0, 1)