

prise_en_main_jupyter

January 2, 2021

Calcul scientifique (INF2B2) Prise en main Jupyter Notebook Exercices de prise en main
Youssef CHAHIR

- Ces ressources sont disponibles en ligne sur [ecampus](#)

Cet tutoriel a pour objectif de vous faire rapidement explorer les **fonctionnalités de base** de l'interpréteur interactif **jupyter** en mode **Notebook**.

Mais tout d'abord **félicitations** d'être déjà parvenu à ouvrir ce notebook !

Table of Contents

1 Jupyter Notebook

1.1 Installer Jupyter

1.2 Lancer Jupyter

1.3 Cellules

1.3.1 Cellules de code

1.3.2 Cellules de texte

1.4 Sauvegarder et imprimer un notebook

1.5 Créer et exécuter des fichiers Python depuis jupyter

1.5.1 Créer un script :

1.5.2 Exécution d'une commande :

1.6 Des exemples en Python

1.6.1 Ecrire des programmes et les exécuter depuis Jupyter

1.6.2 Récupération des entrées / sorties précédentes

1.6.3 Système d'exploitation

1.6.4 Mesure du temps et profilage de programmes

1.7 Graphiques dans un notebook

1.7.1 Intégrer une image :

1.7.2 Intégrer une vidéo youtube :

1.7.3 Intégrer une page web :

1.8 Autres fonctionnalités utiles

1.9 Exemple : Tableaux avec numpy et affichage avec Matplotlib

1.9.1 Quelques micro exemples très pratiques

1.10 Conseils

2 Annexes : Quelques éléments de Markdown pour Jupyter

2.0.1 Italique

2.0.2 Gras

2.0.3 Strike-through

- 2.0.4 Police de taille fixe
 - 2.0.5 URLs
 - 2.0.6 Nouveaux paragraphes
 - 2.0.7 Verbatim
 - 2.0.8 Tableaux
 - 2.0.9 Ligne horizontale
 - 2.0.10 Headings
 - 2.0.11 Block quote
 - 2.0.12 Listes non ordonnées
 - 2.0.13 Listes ordonnées
 - 2.0.14 Image
 - 2.0.15 LaTeX
 - 2.0.16 Références et liens
-

1 Jupyter Notebook

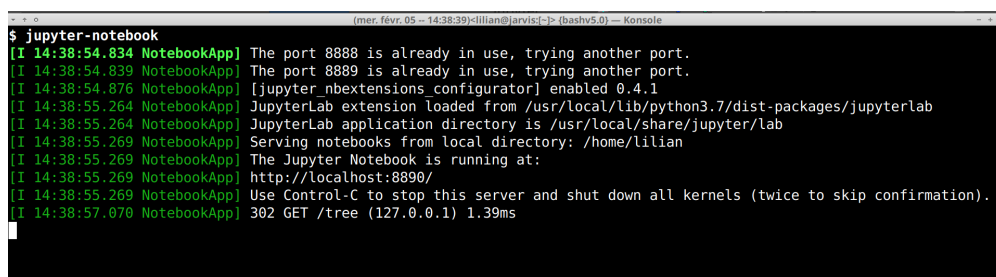
1.1 Installer Jupyter

En suivant le tutoriel en ligne depuis jupyter.org/install.html, il est facile d'installer tout l'écosystème Jupyter sur tout ordinateur avec Python et un gestionnaire de paquets (pip ou conda) installé.

Sur Windows ou Mac OS X, ou même la plupart des systèmes GNU/Linux, un installateur gratuit appelé Anaconda (www.anaconda.com/distribution/) installe tout ça en un clic !

1.2 Lancer Jupyter

- Sous Mac ou GNU/Linux, dans un terminal : `$ jupyter-notebook`,
- Sous Windows, avec Anaconda, il y a un lanceur graphique installé.



```
(mer. févr. 05 - 14:38:39) lilian@jarvis:~$ (bashv5.0) — Konsole
$ jupyter-notebook
[I 14:38:54.834 NotebookApp] The port 8888 is already in use, trying another port.
[I 14:38:54.839 NotebookApp] The port 8889 is already in use, trying another port.
[I 14:38:54.876 NotebookApp] [jupyter_nbextensions_configurator] enabled 0.4.1
[I 14:38:55.264 NotebookApp] JupyterLab extension loaded from /usr/local/lib/python3.7/dist-packages/jupyterlab
[I 14:38:55.264 NotebookApp] JupyterLab application directory is /usr/local/share/jupyter/lab
[I 14:38:55.269 NotebookApp] Serving notebooks from local directory: /home/lilian
[I 14:38:55.269 NotebookApp] The Jupyter Notebook is running at:
[I 14:38:55.269 NotebookApp] http://localhost:8890/
[I 14:38:55.269 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[I 14:38:57.070 NotebookApp] 302 GET /tree (127.0.0.1) 1.39ms
```

Notebook Jupyter :

- Un fichier, à l'extension `.ipynb` (*ipython notebook*), qui est un format de texte brut de type [JSON](#).
- Un outil facile à utiliser pour les débutants et puissant pour les utilisateurs experts.
- Un outil unique pour rédiger des petits morceaux de code (exercices, TP, tutoriel, analyse de données etc), avec du texte ou une documentation, les résultats de l'exécution du code, des figures etc,

- Un fichier .ipynb (slideshow) sui peut être converti en présentation en page web statique (HTML), en document pdf (PDF) ou en script (Python ou autre)
- Permet une bonne compatibilité avec les gestionnaires de version tels que git.

On peut utiliser **Jupyter Lab**, la nouvelle version de l'interface utilisateur de Jupyter (plus moderne).

[jupyterlab](#)

1.3 Cellules

Jupyter est nommé d'après Jupiter, et pour *Julia*, *python*, et *R*, les trois premiers langages pour lesquels Jupyter était disponible.

Initialement, il a commencé sous le nom [ipython](#) car il a été conçu pour être utilisé pour le langage de programmation Python.

- Jupyter est un environnement de développement intégré (IDE) "WYSIWYG" (What-you-see-is-what-you-get).
- Il est utilisé pour différents langages de programmation (que l'on utilise depuis un navigateur Internet) tels que des langages dynamiques interprétés comme Python mais aussi pour des langages compilés, tels que C/C++.
- Tous les logiciels de l'environnement Jupyter, sont gratuits et sous licence libre, comme Python.

Un notebook jupyter consiste en **cellules** individuelles pouvant contenir du **code** Python, du **texte** (avec formatage Markdown, HTML et LaTeX) ou des **graphiques**.

1.3.1 Cellules de code

Commençons par les cellules de type Code. (c'est-à-dire préfixée par **In[]:**)

On est par défaut en mode **édition de code** Python !

1) Placez le curseur dans la 1ère cellule **In[]:** ci-dessous puis frappez <enter>.

Pour **exécuter le code** Python d'une cellule, il faut en effet frapper <maj-enter> (ou cliquer sur le bouton , ou encore utiliser le menu **Cell>Run and Select Below...**). Essayez !

Une ligne **Out[]:** devrait être apparue en dessous de la 1ère expression, affichant le résultat de celle-ci ! Le curseur est alors automatiquement passé dans la cellule d'en-dessous que vous pouvez exécuter à son tour.

*Notez qu'il est aussi possible de frapper <ctrl-enter> (identique à **Cell>Run**). Dans ce cas, le curseur reste dans la cellule après l'exécution du code.*

2) Pour détruire une cellule, sélectionnez-la puis cliquez sur le bouton (ou faites **Edit>Delete Cell**). Exercez-vous en détruisant la cellule ci-dessous. Puis rétablissez-la avec **Edit>Undo Delete Cell**

3) On peut **splitter** une cellule : pour cela *double-cliquez* d'abord dans la cellule pour l'éditer, puis placez le curseur entre les 2 lignes, et faites finalement **Edit>Split Cell**

4) On peut **fusionnement** plusieurs cellules de texte avec **Edit>Merge Cell Above** ou **Edit>Merge Cell Below**

1.3.2 Cellules de texte

Il existe 2 types de cellules de texte, selon le choix effectué à l'aide du menu déroulant de droite ou avec `Cell>Cell Type>...` : - **Raw NBConvert** (texte) : cellules de texte sans aucun formatage, directement éditables sans avoir à *double-cliquer* dedans ; - **Markdown** : cellules dans lesquelles le texte est soumis à un formatage selon la syntaxe Markdown ; les éventuelles balises HTML et le code LaTeX sont également interprétés...

5) S'agissant des cellules de texte au format **Markdown**, cela suppose quelques notions relatives au langage de balisage Markdown :

Et il est même possible d'introduire du **code LaTeX** \LaTeX^2 (placé entre deux caractères `$`) dans une cellule de type Markdown :

$$f(x) = \frac{1}{2}x^2 - x^{\frac{\pi}{4}}$$

L'interface graphique des notebooks Jupyter permet d'éditer des cellules de texte ou de code, et exécuter des cellules.

Pour plus d'informations : - [Rédigez en Markdown !](#) - [un Notebook assez complet sur les cellules au format Markdown](#)

Un document contient des cellules de texte (en Markdown), et de code.

Ceci est une cellule de texte.

[45]: 3+2

[45]: 5

1.4 Sauvegarder et imprimer un notebook

- Pour **sauvegarder** un notebook, on peut utiliser le bouton (ou le menu `File>Save and Checkpoint`). Sachez que Jupyter effectue par défaut des sauvegardes à intervalle régulier, et que `File>Revert to Checkpoint` permet de revenir à un état enregistré.
- Pour **imprimer un notebook**, n'utilisez pas la commande d'impression de votre navigateur web depuis la fenêtre de notebook, le résultat sera très mauvais. La bonne méthode consiste à faire au préalable depuis le menu `File>Print Preview`. Mais il est peut-être préférable de télécharger une version numérique du notebook sur l'ordinateur local dans le format qui vous convient comme le HTML pour obtenir une page web statique avec le menu `File>Download as... ou pdf...`

1.5 Créer et exécuter des fichiers Python depuis Jupyter

Documentation sur les différentes commandes : [https://ipython.readthedocs.io/en/stable/interactive/magics.h](https://ipython.readthedocs.io/en/stable/interactive/magics.html)

1.5.1 Créer un script :

```
[46]: %%writefile hello.py
      print("Bonjour de Python!")
```

Overwriting hello.py

1.5.2 Exécution d'une commande :

Pour exécuter une commande shell, on utilise un point d'exclamation (!) qui doit toujours précéder la commande en question.

Par exemple la commande `ls` (ou `dir` sur windows) permet de montrer tous les éléments qui se trouvent au niveau du répertoire actuel ainsi que son emplacement.

Si le résultat tarde à s'afficher, on peut relancer le kernel ou le noyau `IPython` :

Dans Python avec Jupyter, on a accès au shell/terminal

[47]: `%ls -l`

```
total 1912
drwxr-xr-x@ 10 chahir  staff      320  2 jan 13:30  figs/
-rw-r--r--   1 chahir  staff       28  2 jan 19:59  hello.py
-rw-r--r--@   1 chahir  staff  445427  2 jan 19:10  prise_en_main_jupyter.html
-rwxr-xr-x@   1 chahir  staff  353836  2 jan 19:59
prise\_en\_main\_jupyter.ipynb\*
-rw-r--r--   1 chahir  staff  168292  2 jan 19:36  res.pdf
```

[48]: `!ls -l`

```
total 1912
drwxr-xr-x@ 10 chahir  staff      320  2 jan 13:30  figs
-rw-r--r--   1 chahir  staff       28  2 jan 19:59  hello.py
-rw-r--r--@   1 chahir  staff  445427  2 jan 19:10  prise_en_main_jupyter.html
-rwxr-xr-x@   1 chahir  staff  353836  2 jan 19:59
prise\_en\_main\_jupyter.ipynb
-rw-r--r--   1 chahir  staff  168292  2 jan 19:36  res.pdf
```

[49]: `!python3 hello.py`

Bonjour de Python!

[50]: `!python3 --version`

Python 3.7.9

[51]: `# (ici dans le terminal, depuis Python dans Jupyter ou IPython, avec !commande .
↪...)
!file "prise_en_main_jupyter.ipynb"`

prise_en_main_jupyter.ipynb: HTML document text, UTF-8 Unicode text, with very long lines

[52]: `import os
statinfo = os.stat('prise_en_main_jupyter.ipynb')`

```
statinfo
```

```
[52]: os.stat_result(st_mode=33261, st_ino=8621370351, st_dev=16777220, st_nlink=1,
st_uid=502, st_gid=20, st_size=353836, st_atime=1609614006, st_mtime=1609613972,
st_ctime=1609613972)
```

```
[53]: !head "prise_en_main_jupyter.ipynb"
```

```
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {
        "slideshow": {
          "slide_type": "slide"
        }
      },
      "source": [
```

```
[54]: !grep "Jupyter" prise_en_main_jupyter.ipynb | head -n1
```

```
"    <FONT size=\"6\"> Prise en main Jupyter Notebook </FONT><br>\n",
```

1.6 Des exemples en Python

1. Premier contact avec Python,
2. Manipulation de tableaux numpy et affichage avec Matplotlib,
3. Un exemple plus impressionnant : pavage de Penrose.

1.6.1 Ecrire des programmes et les exécuter depuis Jupyter

1. Des petits exemples d'utilisation de Python
2. Utilisation de modules Python

```
[55]: def somme (x,y):
      return (x+y)
```

```
[56]: x = somme(3,4)
      x
```

```
[56]: 7
```

```
[57]: nombres = [2, 4, 6, 8]
      produit = 1
      for nombre in nombres:
          produit *= nombre

      print("Le produit de ces", len(nombres), "nombres est", produit)
```

Le produit de ces 4 nombres est 384

```
[58]: import datetime

print("Date actuelle :")
print(datetime.datetime.now())
```

Date actuelle :
2021-01-02 20:00:22.735349

```
[59]: def fib(n):
      """ Affiche les premières valeurs de la suite de Fibonacci <= n. """
      a, b = 0, 1
      while a < n:
          print(a, end=' ')
          a, b = b, a+b
      print()

fib(1000)
```

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

1.6.2 Récupération des entrées / sorties précédentes

```
[60]: 3 * 3
```

```
[60]: 9
```

```
[62]: #Entrée précédente (60)
```

```
In[60]
```

```
[62]: '3 * 3'
```

```
[63]: #Sortie 62
```

```
Out[62]
```

```
[63]: '3 * 3'
```

```
[99]: # Toutes les entrées
```

```
In
```

```
[100]: # Toutes les sorties
```

```
Out
```

```
[66]: 1+2
```

```
[66]: 3
```

```
[67]: 1+2;
```

```
[68]: x = 2; x
```

```
[68]: 2
```

1.6.3 Système d'exploitation

```
[69]: import platform

plt = platform.system()
print(platform.system())
print(platform.release())
print(platform.version())

if plt == "Windows":
    print("Votre système est Windows")
    # do x y z
elif plt == "Linux":
    print("Votre système est Linux")
    # do x y z
elif plt == "Darwin":
    print("Votre système est MacOS")
    # do x y z
else:
    print("Système non identifié")
```

Darwin

17.7.0

Darwin Kernel Version 17.7.0: Mon Aug 31 22:11:23 PDT 2020;

root:xnu-4570.71.82.6~1/RELEASE_X86_64

Votre système est MacOS

```
[96]: import math
      #Aide sur cos ?
      math.cos?
```

1.6.4 Mesure du temps et profilage de programmes

```
[97]: %timeit fib(10)
```

1.33 μ s \pm 40.9 ns per loop (mean \pm std. dev. of 7 runs, 1000000 loops each)

```
[72]: def fib(N):
      """
      Retourne la liste des N premiers nombres de la suite de Fibonacci.
      """
      f0, f1 = 0, 1
```



```

f = [1] * N
for n in range(1, N):
    f[n] = f0 + f1
    f0, f1 = f1, f[n]

return f

print(fib(10))

```

[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

Profilage des programmes (combien de temps CPU passé sur quelles fonctions)

```
[73]: result = %time fib(50)
```

CPU times: user 11 μ s, sys: 0 ns, total: 11 μ s
 Wall time: 14.1 μ s

1.7 Graphiques dans un notebook

Si vous exécutez la fonction magique `%pylab inline`, les packages **Numpy** et **Matplotlib** sont importés et il sera possible de dessiner des graphiques de façon “inline” (intégrés au notebook).

```
[75]: %pylab inline
```

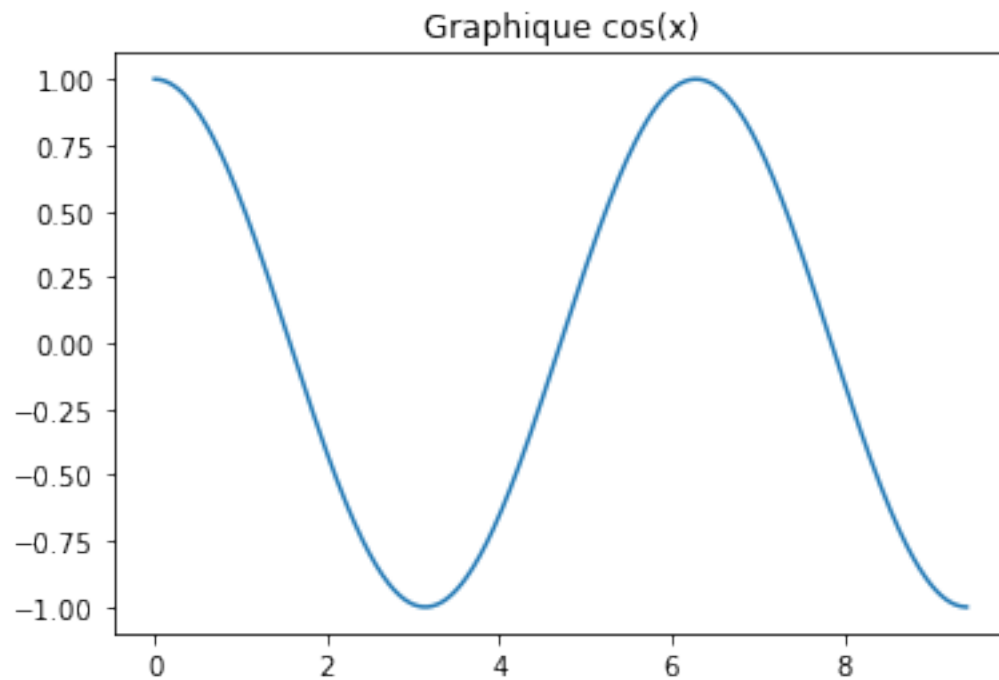
Populating the interactive namespace from numpy and matplotlib

L'exemple de code suivant sera alors exécutable.

```
[76]: # Fait appel à numpy (linspace et pi)
x = linspace(0, 3*pi, 500)

# Fait appel à matplotlib (plot et title)
plot(x, cos(x))
title('Graphique cos(x)')
```

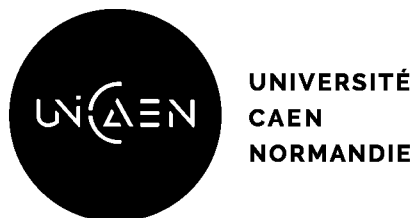
```
[76]: Text(0.5, 1.0, 'Graphique cos(x)')
```



1.7.1 Intégrer une image :

```
[77]: from IPython.display import Image  
Image("figs/UNICAEN-logo-NOIR-horizontal.png", width = 100, height = 100)
```

[77]:



1.7.2 Intégrer une vidéo youtube :

```
[78]: from IPython.display import YouTubeVideo
      YouTubeVideo('Rc4JQWowG5I')
```

[78]:



1.7.3 Intégrer une page web :

```
[79]: from IPython.display import HTML
      HTML('<IFrame src="https://fr.wikipedia.org/wiki/Jupyter" width=700_
      ↪height=350>')
```

[79]: <IPython.core.display.HTML object>

1.8 Autres fonctionnalités utiles

- En mode édition, bordures vertes, lorsque le curseur est en début de ligne ou lorsque vous avez sélectionné du texte, l'appui sur la touche <Tab> (respectivement <Shift-TAB>) indente (respectivement désindente) les lignes correspondantes.

- La touche <ESC> passe en mode commande, les bordures sont alors bleues. Notez bien que dans ce mode, certaines touches du clavier sont associées à des actions sur le notebook. Tenter de saisir du texte dans ce mode peut donc produire des effets inattendus.
- Le bouton [open the command palette] affiche une fenêtre qui liste l'ensemble des commandes.
- Pour **créer vos propres notebooks**, utilisez le bouton [New Notebook] dans la première fenêtre "jupyter Dashboard", ou faites File>New depuis n'importe quelle fenêtre de notebook. N'oubliez ensuite pas de donner un nom de fichier à votre notebook en cliquant dans le champ "Untitled" au haut de la fenêtre du nouveau notebook
- Pour **effacer tous les outputs** d'un notebook, vous pouvez faire Cell>All Output>Clear
- Si jupyter Notebook semble bloqué, c'est sûrement qu'il **tourne indéfiniment** en raison par exemple d'une boucle mal contrôlée (consommant 100% de CPU d'1 core de votre machine). Vous deviez voir le message "Kernel busy" au haut de l'écran. Pour **l'interrompre**, cliquez sur le bouton (ou menu Kernel>Interrupt). Si ça ne fonctionne vraiment pas, il faudra se résoudre à tuer le serveur web avec <ctrl-C>
- Sous Help>Keyboard Shortcuts vous trouverez la liste des nombreux **raccourcis clavier** améliorant grandement l'efficacité d'utilisation de jupyter Notebook !
- Sous Help>Notebook Help vous accéderez à la **documentation** complète de jupyter Notebook.

1.9 Exemple : Tableaux avec numpy et affichage avec Matplotlib

Sur un plateau style Monopoly à 12 cases, un pion part de la case 0, et on joue au dé (à 6 faces) pour avancer à chaque coup de x cases (où $x \sim \mathcal{U}(1, \dots, 6)$).

L'exercice demande de simuler la variable aléatoire Y_n représentant la case sur laquelle le pion se trouve après n déplacements.

```
[93]: import numpy as np
import numpy.random as rd

import matplotlib.pyplot as plt
case_max = 10
def case_aleatoire(case):
    return (case + rd.randint(1, 6+1)) % case_max

def Yn(duree, depart=0):
    case = depart
    for coup in range(duree):
        case = case_aleatoire(case)
    return case
[Yn(1) for _ in range(10)] # toutes 1 <= ... <= 6
```

```
[93]: [1, 1, 6, 3, 3, 3, 2, 3, 3, 5]
```

```
[94]: def histogramme(duree, repetitions=10000):
    cases = [Yn(duree) for _ in range(repetitions)]
```

```

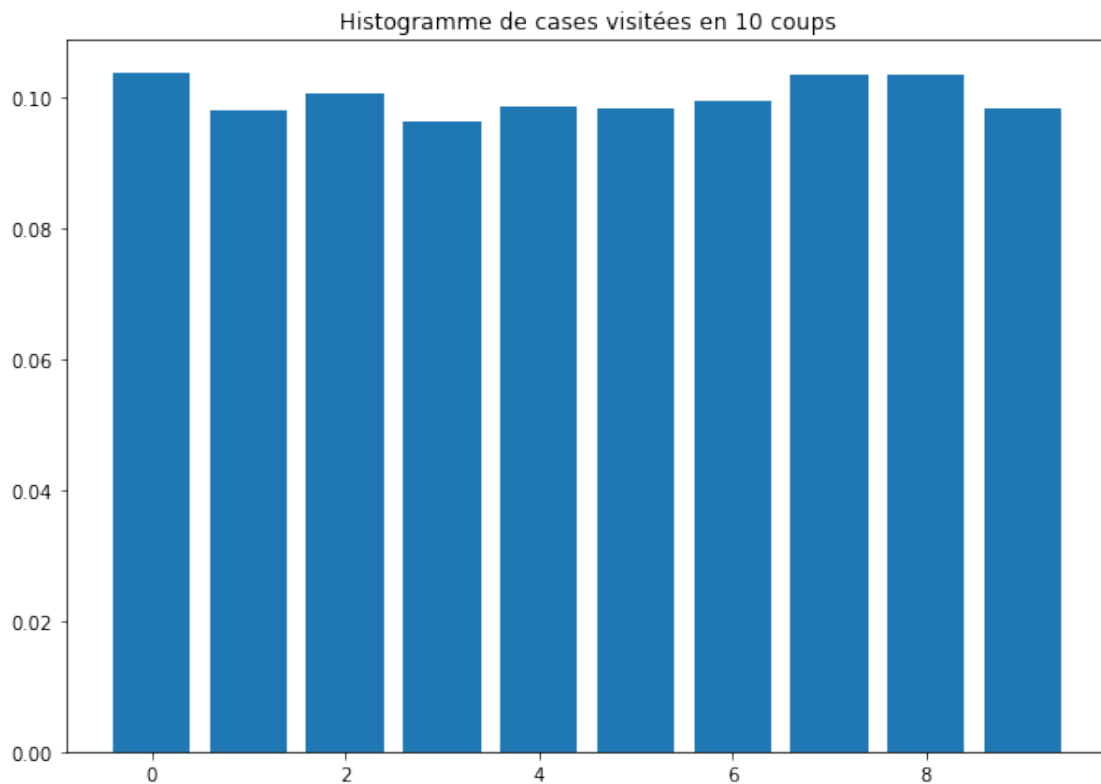
frequences = [0] * case_max
for case in cases:
    frequences[case] += 1
return frequences / np.sum(frequences)

```

```

[95]: n = 10
_ = plt.figure(figsize=(10, 7))
_ = plt.bar(np.arange(case_max), histogramme(n))
_ = plt.title("Histogramme de cases visitées en " + str(n) + " coups")
plt.show()

```



1.9.1 Quelques micro exemples très pratiques

Dans Python avec Jupyter, on a accès au shell/terminal de l'environnement facilement :

1.10 Conseils

- Installer Python et Jupyter sur votre ordinateur personnel ;
- Essayer de rédiger un notebook Jupyter au lieu d'utiliser votre éditeur / IDE préféré ;
- Apprendre par soi-même à utiliser l'interface de Jupyter ;