Introduction aux notebooks Jupyter

February 5, 2020

1 Table of Contents

1 Introduction aux notebooks Jupyter

1.1 Présentation de Jupyter

1.1.1 Qu'est-ce que Jupyter ?

1.1.2 Qu'est-ce qu'un notebook Jupyter ?

1.1.3 Qu'est-ce que l'écosystème Jupyter ?

1.1.4 Quels problèmes résolvent les notebooks Jupyter ?

1.2 Comment installer Jupyter

1.3 Comment utiliser Jupyter pour écrire des documents simples

1.3.1 Aperçu de l'interface de Jupyter : comment lancer Jupyter

1.3.2 Aperçu de l'interface de Jupyter : un notebook vide

1.3.3 Aperçu de l'interface de Jupyter : un notebook en train d'être édité

1.4 Des exemples en Python

1.4.1 Premier contact avec Python

1.4.2 Manipulation de tableaux numpy et affichage avec Matplotlib

1.4.3 Pavage de Penrose avec un joli programme

1.5 Présentant ma propre utilisation de notebooks Jupyter

1.5.1 Quelques micro exemples très pratiques

1.5.2 Correction interactive de TP en prépa (PSI)

1.5.3 Cours d'algorithmique en L3 informatique

1.5.4 Entraînement à l'oral de modélisation d'agrégation (maths option info)

1.6 Conclusion : conseils pour devenir un-e expert-e en Jupyter

1.7 Références et liens

1.7.1 Liens

1.7.2 Publications universitaires

2 Introduction aux notebooks Jupyter

Ce tutoriel durera 60 minutes, le mercredi 05 février, à la conférence Didapro #8, à Lille.

1. Suivez la présentation sur votre écran : `$ regarder_ecran nom_ma_machine`,
2. Ensuite il y aura une activité *à faire par vous même*,
3. Nous terminerons par des exemples de ma propre utilisation de Jupyter et des démonstrations.

Par Lilian Besson, de l'ENS de Rennes (département Informatique).
Liens : perso.crans.org/besson & GitHub.com/Naereen

- Ces ressources sont disponibles en ligne sur frama.link/Atelier-Jupyter-Didapro8
- A FAIRE MAINTENANT : Ouvrez un navigateur web, et allez sur ce lien. Cliquez sur le lien "Binder" et laissez cette page charger.
- N'hésitez pas à m'interrompre et poser des questions, si besoin.

2.1 Présentation de Jupyter

- Qu'est-ce que Jupyter ?
- Qu'est-ce qu'un notebook Jupyter ?
- Qu'est-ce que l'écosystème Jupyter ?
- Quels problèmes résolvent les notebooks Jupyter ?

2.1.1 Qu'est-ce que Jupyter ?

Un environnement de développement intégré (IDE) "WYSIWYG" (What-you-see-is-what-you-get) pour (presque) tous les langages de programmation, que l'on utilise depuis un navigateur Internet.

Par exemple, il peut être utilisé pour des langages dynamiques interprétés, tels que Python, OCaml, Julia ou Bash, mais aussi pour des langages compilés, tels que C/C++.

C'est un ensemble de logiciels libres et gratuits, installables sur n'importe quel ordinateur moderne et qui sont faciles à prendre en main.

2.1.2 Qu'est-ce qu'un notebook Jupyter ?

C'est un fichier, à l'extension `.ipynb` (*ipy*thon *n*ote*b*ook), qui est un format de texte brut de type JSON.

Ce fichier peut être converti en présentation (slide show) comme celle utilisée aujourd'hui, en page web statique (HTML), en document prêt à être imprimé (PDF), en script (Python ou autre)

Le format de fichier, et tous les logiciels de l'environnement Jupyter, sont gratuits et sous licence libre, comme Python.

Un document contient des cellules de texte (en Markdown ↑), et de code ↓.
Ceci est une cellule de texte.

[6]:
```
# Et celle ci est une celle de code
# (ici dans le terminal, depuis Python dans Jupyter ou IPython, avec !commande ...)
!file "Introduction aux notebooks Jupyter.ipynb"
```

```
Introduction aux notebooks Jupyter.ipynb: HTML document, UTF-8 Unicode text,
with very long lines
```

[7]:
```
!head "Introduction aux notebooks Jupyter.ipynb"
```

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {
    "slideshow": {
     "slide_type": "skip"
    },
    "toc": "true"
   },
```

[10]:
```
!grep "contient des cellules" "Introduction aux notebooks Jupyter.ipynb" | head␣
↪-n1
```

```
    "Un document <span style=\"color:red;\">contient des cellules</span> de
texte (en Markdown ↑), et de code ↓.  \n",
```

2.1.3 Qu'est-ce que l'écosystème Jupyter ?

Il a commencé sous le nom ipython il y a environ 15 ans, conçu pour être utilisé uniquement pour le langage de programmation Python, et de nos jours il a évolué en un écosystème open-source mature.

Jupyter est nommé d'après Jupiter, et pour *Ju*lia, *pyt*hon, et *R*, les trois premiers langages pour lesquels Jupyter était disponible.

L'environnement Jupyter est utilisé par des centaines de milliers de scientifiques du monde entier, allant d'étudiants au lycée, en prépa et à l'université en France et ailleurs, aux meilleures équipes utilisant les sciences des données et du numérique.

Les notebooks Jupyter sont une alternative gratuite et open-source à l'EDI inclus dans les logiciels propriétaires et payants qui dominent le marché :

- MATLAB,
- Wolfram's Mathematica,
- et MapleSoft's Maple.

Parmi ses récentes utilisations réussies, on peut noter :

1. La toute première image d'un trou noir obtenue par Katie Bouman et ses collègues.
   Voir par exemple www.nationalgeographic.com/science/2019/04/first-picture-black-hole-revealed
   et www.bbc.com/news/science-environment-47891902.
2. Ou pour l'analyse de données par des économistes, comme le lauréat de prix Nobel, Paul Romer.
   Voir par exemple PaulRomer.net/jupyter-mathematica-and-the-future-of-the-research-paper/

2.1.4 Quels problèmes résolvent les notebooks Jupyter ?

Pourquoi c'est un outil puissant, à la fois facile à apprendre et 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,
- Facilité de conversion vers d'autres formats : scripts exécutables sans Jupyter, fichiers statiques HTML, fichiers imprimables PDF etc
- Encore plus d'autres formats : présentations (slides) interactives,
- Stocké comme des fichiers textes, bonne compatibilité avec les gestionnaires de version tels que `git`.

2.2 Comment 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.

→ `jupyter.org/install.html`

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

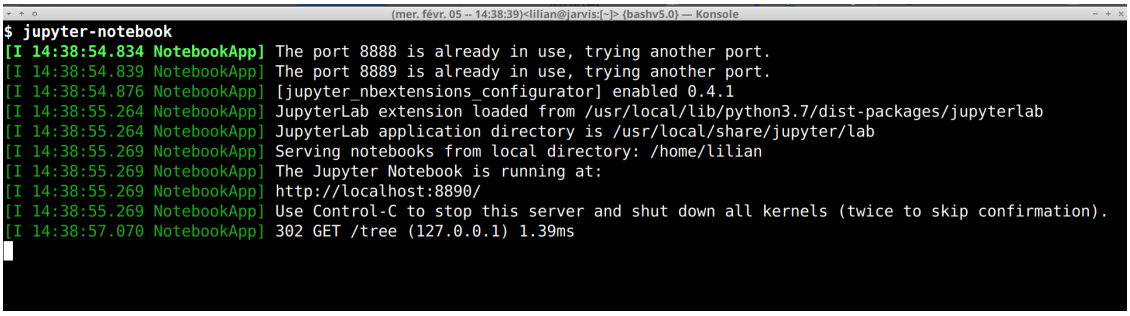
2.3 Comment utiliser Jupyter pour écrire des documents simples

Je vais vous montrer l'interface utilisateur graphique des notebooks Jupyter, et nous allons voir ensemble comment éditer des cellules de texte ou de code, et exécuter des cellules.

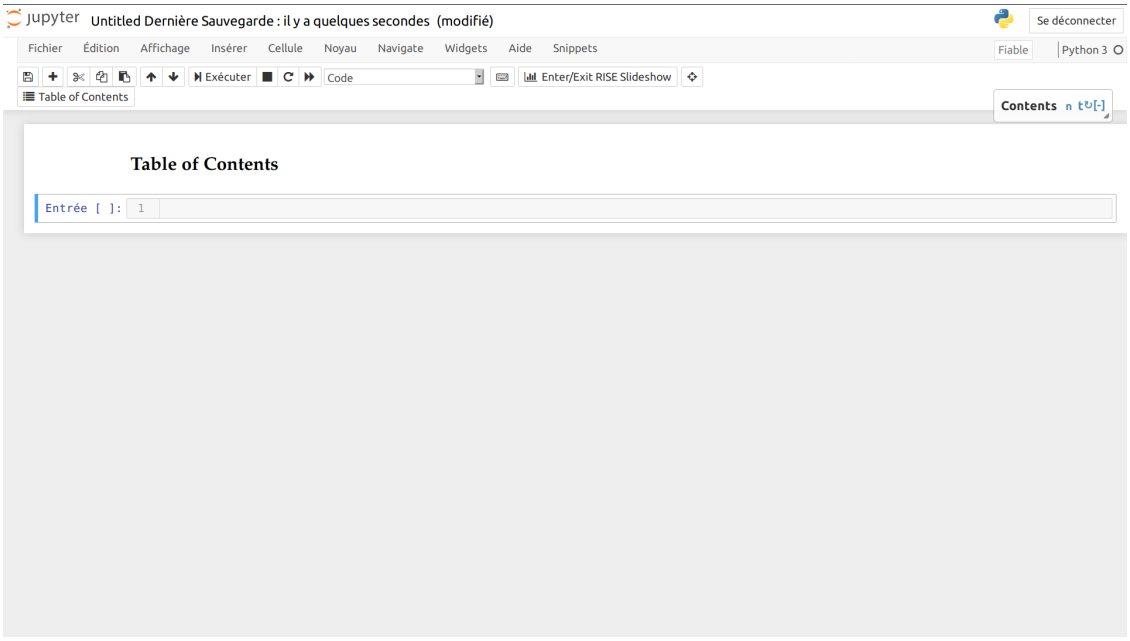
Ceci est une **cellule de texte**. On peut *utiliser* le langage à balise Markdown ! Et LaTeX[2]

2.3.1 Aperçu de l'interface de Jupyter : comment lancer Jupyter

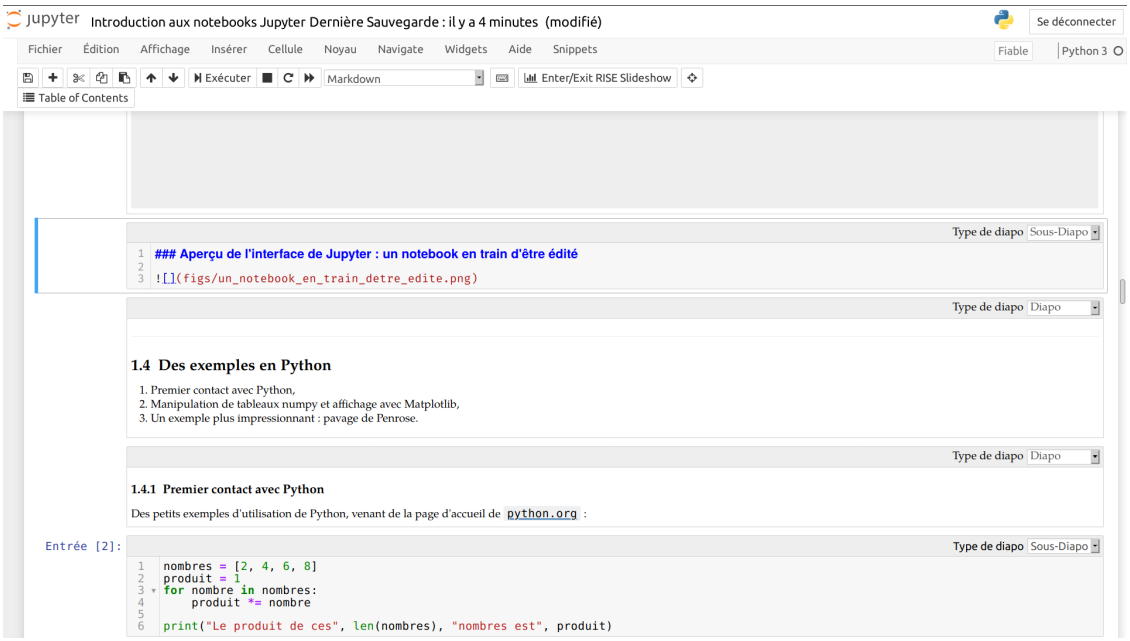
- Sous Mac ou GNU/Linux, dans un terminal : `$ jupyter-notebook`,
- Sous Windows, avec Anaconda, il y a un lanceur graphique installé.



2.3.2 Aperçu de l'interface de Jupyter : un notebook vide



2.3.3 Aperçu de l'interface de Jupyter : un notebook en train d'être édité



2.4 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.

2.4.1 Premier contact avec Python

Des petits exemples d'utilisation de Python, venant de la page d'accueil de python.org :

[2]:
```
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
```

[3]:
```
import datetime

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

```
Date actuelle :
2020-02-05 09:59:11.046061
```

[4]:
```
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
```

Depuis un notebook Jupyter, la documentation de n'importe quelle valeur peut être affichée avec la touche `Tab`, ou en utilisant :

[51]:
```
fib?
```

2.4.2 Manipulation de tableaux numpy et affichage avec Matplotlib

> Cet exemple est tiré d'un exercice d'oral de CentraleSupélec pour PSI (166 officiel de la Taupe 2019).

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.

[12]:
```
import numpy as np
import numpy.random as rd

import matplotlib.pyplot as plt
import seaborn as sns
# pour de jolis graphiques
sns.set(context="notebook", style="whitegrid", palette="hls",␣
 ↪font="sans-serif", font_scale=1.3)
```

[13]:
```
case_max = 12

def prochaine_case(case):
    return (case + rd.randint(1, 6+1)) % case_max  # modulo 12

def Yn(duree, depart=0):
    case = depart
    for coup in range(duree):
        case = prochaine_case(case)
    return case
```

Avant de s'en servir pour simuler plein de trajectoirs, on peut vérifier que en un coup, on avance pas plus de 6 cases :

[14]:
```
[Yn(1) for _ in range(10)]  # toutes 1 <= ... <= 6
```

[14]:
```
[5, 6, 6, 5, 4, 4, 5, 5, 5, 3]
```

On peut ensuite réaliser des histogrammes des 12 valeurs possibles pour Y_n, avec par exemple 10000 répétitions iid.

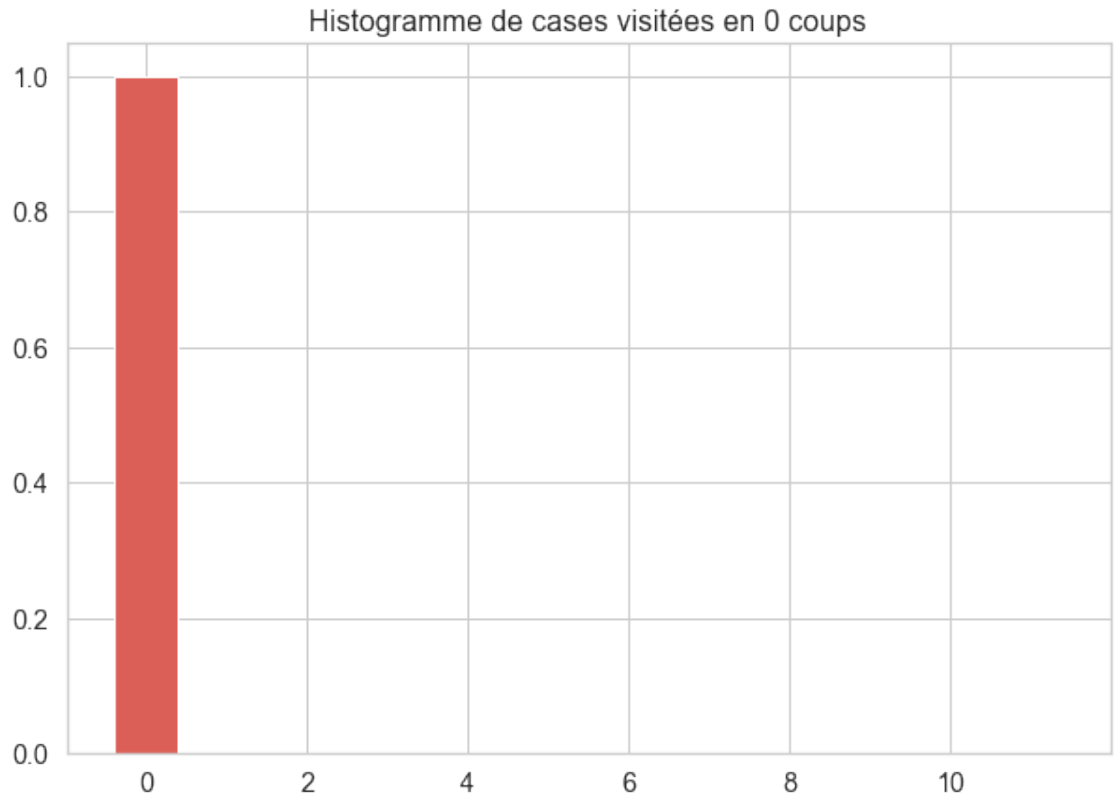
[15]:
```
def histogramme(duree, repetitions=10000):
    cases = [Yn(duree) for _ in range(repetitions)]
    frequences = [0] * case_max
    for case in cases:
        frequences[case] += 1
    return frequences / np.sum(frequences)
```

[16]:
```
n = 0
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()
```

[16]:
```
<Figure size 720x504 with 0 Axes>
```

[16]:
```
<BarContainer object of 12 artists>
```

[16]:
```
Text(0.5, 1.0, 'Histogramme de cases visitées en 0 coups')
```

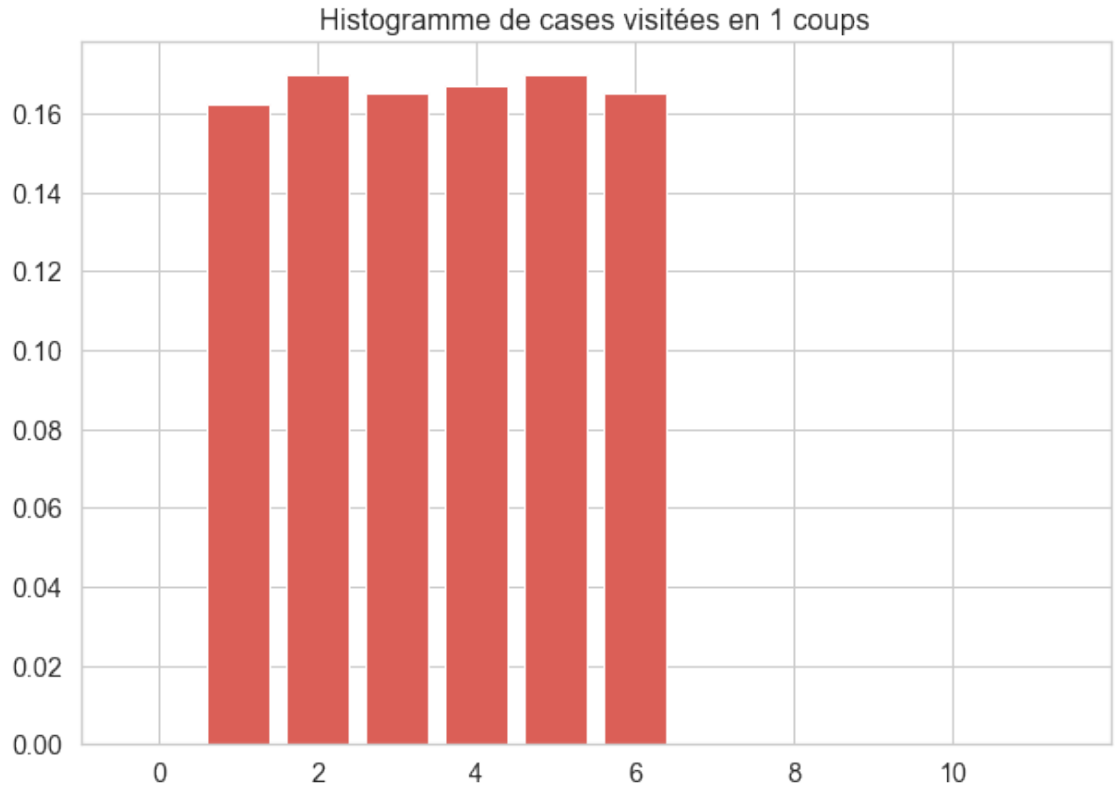


[17]:
```
n = 1
plt.figure(figsize=(10, 7))
plt.bar(np.arange(case_max), histogramme(n))
plt.title("Histogramme de cases visitées en " + str(n) + " coups")
plt.show()
```

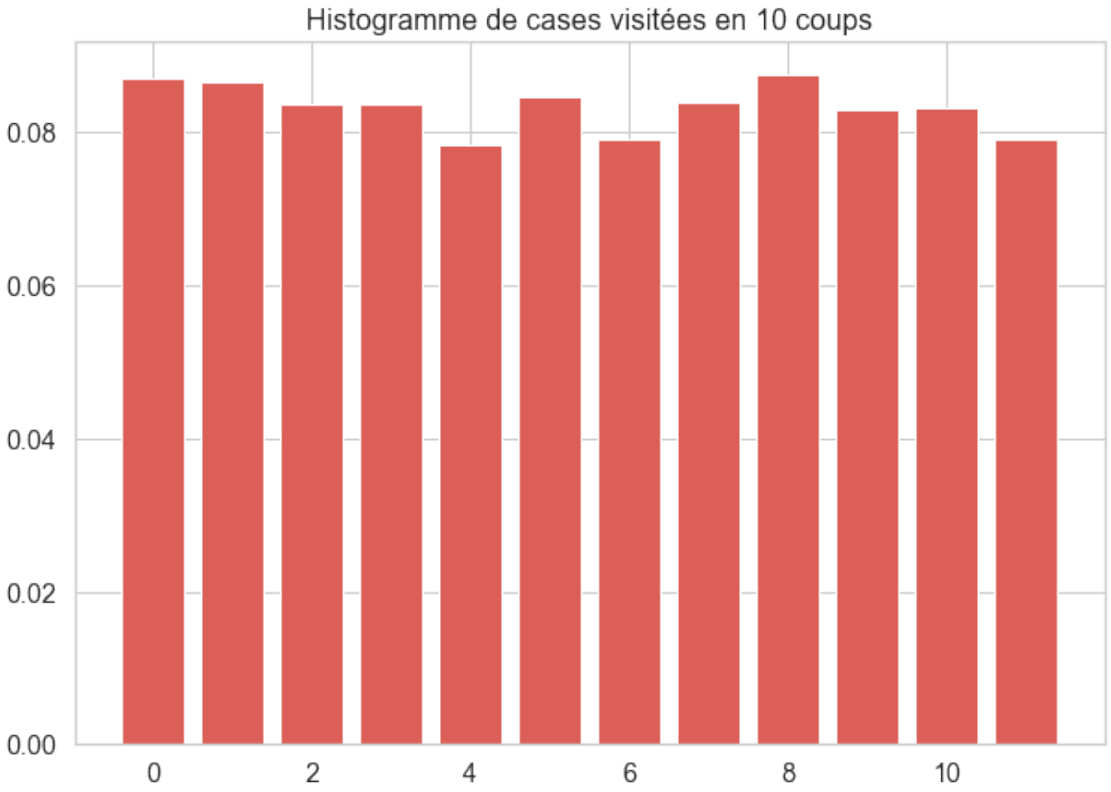
[17]: <Figure size 720x504 with 0 Axes>

[17]: <BarContainer object of 12 artists>

[17]: Text(0.5, 1.0, 'Histogramme de cases visitées en 1 coups')



[18]:
```
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()
```



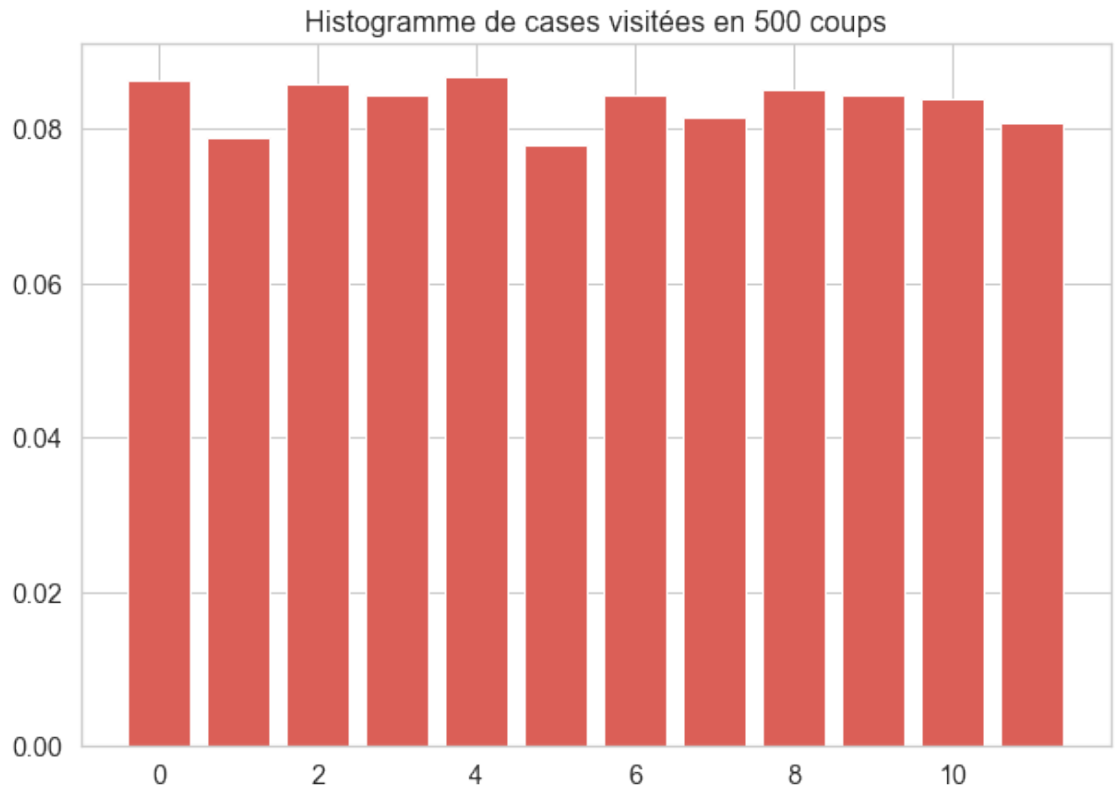
Cela montre qu'avec une trajectoire assez longue, le pion a une probabilité identique de terminer sur chaque position.

[19]:
```
n = 500
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()
```

[19]:
```
<Figure size 720x504 with 0 Axes>
```

[19]:
```
<BarContainer object of 12 artists>
```

[19]:
```
Text(0.5, 1.0, 'Histogramme de cases visitées en 500 coups')
```



2.4.3 Pavage de Penrose avec un joli programme

Ce petit code (qui n'est pas de moi) affiche un pavage (infini) de Penrose et le sauvegarde dans une image PNG de taille 2000×2000 :

[6]:
```
%%time
from functools import reduce
_                                    =\
                                   """if!
                                 1:"e,V=200
                               0,(0j-1)**-.2;
                             v,S=.5/  V.real,
                           [(0,0,4      *e,4*e*
                         V)];w=1          -v"def!
                       E(T,A,               B,C):P
                   ,Q,R=B*w+                 A*v,B*w+C
              *v,A*w+B*v;retur               n[(1,Q,C,A),(1,P
     ,Q,B),(0,Q,P,A)]*T+[(0,C               ,R,B),(1,R,C,A)]*(1-T)"f
or!i!in!_[:11]:S       =sum([E           (*x)for       !x!in!S],[])"imp
  ort!cair               o!as!0;       s=0.Ima              geSurfac
   e(1,e,e)               ;c=0.Con  text(s);              M,L,G=c.
     move_to                ,c.line_to,c.s               et_sour
```

```
      ce_rgb                a"def!z(f,a)                :f(-a.
       imag,a.       real-e-e)"for!T,A,B,C!in[i       !for!i!
         in!S!if!i["""];exec(reduce(lambda x,i:x.replace(chr
          (i),"\n "[34-i:]),   range(   35),_+"""0]]:z(M,A
            );z(L,B);z         (L,C);         c.close_pa
            th()"G             (.4,.3             ,1);c.
            paint(             );G(.7             ,.7,1)
            ;c.fil             l()"fo             r!i!in
            !range             (9):"!             g=1-i/
            8;d=i/          4*g;G(d,d,d,          1-g*.8
            )"!def     !y(f,a):z(f,a+(1+2j)*(     1j**(i
            /2.))*g)"!for!T,A,B,C!in!S:y(M,C);y(L,A);y(M
            ,A);y(L,B)"!c.st            roke()"s.write_t
            o_png('figs/                   penrose.png')
            """                                    ))
```

```
CPU times: user 4.49 s, sys: 0 ns, total: 4.49 s
Wall time: 4.5 s
```

2.5 Présentant ma propre utilisation de notebooks Jupyter

Je souhaite vous montrer différents cas d'utilisation des cahiers Jupyter au quotidien pour mes activités d'enseignement, durant les trois dernières années.

Je présenterai principalement des exemples de ressources produites à partir d'un notebook Jupyter, et comment convertir des notebooks en HTML.

2.5.1 Quelques micro exemples très pratiques

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

[21]:
```
!ls -larth paper/*.png figs/*.png | head -n 10
```

```
-rw-r--r-- 1 lilian lilian  45K oct.  22 09:58 paper/apercu_ENS_agreg_1.png
-rw-r--r-- 1 lilian lilian  51K oct.  22 09:58 paper/apercu_ENS_agreg_2.png
-rw-r--r-- 1 lilian lilian  87K oct.  22 09:58 paper/apercu_ENS_agreg_3.png
-rw-r--r-- 1 lilian lilian 125K oct.  22 09:58 paper/apercu_ENS_agreg_4.png
-rw-r--r-- 1 lilian lilian 108K oct.  22 09:58
paper/interactive_Turing_Machine_simulator_1.png
-rw-r--r-- 1 lilian lilian  94K oct.  22 09:58
paper/interactive_Turing_Machine_simulator_2.png
-rw-r--r-- 1 lilian lilian 129K févr.  5 09:55 figs/apercu_prepa2.png
-rw-r--r-- 1 lilian lilian  85K févr.  5 09:55 figs/apercu_prepa.png
-rw-r--r-- 1 lilian lilian 253K févr.  5 09:55
figs/git_diff_for_jupyter_notebooks.png
-rw-r--r-- 1 lilian lilian  68K févr.  5 09:55 figs/installing_jupyter.png
```

[23]:
```
# Quelle est mon adresse IP ?
!wget --quiet -O - http://monip.org/ | html2text
```

```
IP : 131.254.243.175
   131.254.243.175


Pas de proxy détecté - No Proxy detected
```

2.5.2 Correction interactive de TP en prépa (PSI)

J'ai écrit les solutions pour les sessions pratiques données en tant qu'entraînement pour l'épreuve orale "Mathématiques avec Python" du concours CentraleSupélec (CPGE), avec des notebooks Jupyter, afin de les partager facilement avec les étudiants, de les exposer et de travailler dessus pendant les sessions pratiques. Voir perso.crans.org/besson/notebooks/Oraux_CentraleSupelec_PSI__Juin_2019.html, avec Python et des maths, pour les étudiants d'une classe de PSI (CPGE) en juin 2017, 2018 et 2019.

2.5.3 Cours d'algorithmique en L3 informatique

J'ai écrit des implémentations propres et détaillées de structures de données et d'algorithmes, pour un cours d'algorithmique à l'automne 2019, avec des notebooks Jupyter.
Voir github.com/Naereen/ALGO1-Info1-2019, avec le langage Python, pour des étudiants de L3.

2.5.4 Entraînement à l'oral de modélisation d'agrégation (maths option info)

J'ai utilisé des notebooks Jupyter pour écrire les sujets et les solutions des sessions pratiques utilisées pour entraîner nos étudiants à l'épreuve orale de modélisation de l'agrégation. Voir nbviewer.jupyter.org/github/Naereen/notebooks/tree/master/agreg/TP_Programmation_2017-18/, avec le langage OCaml, pour des étudiants de M2.

2.6 Conclusion : conseils pour devenir un-e expert-e en Jupyter

- 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 ;
- Comment utiliser un dépôt GitHub/Bitbucket/GitLab pour héberger des notebooks Jupyter, et les afficher en ligne en utilisant le site nbviewer.jupyter.org.
  - nbviewer.jupyter.org/github/Naereen/Tutoriel-notebooks-Jupyter-a-Didapro-8-Lille-fevr
  - Voir github.com/Naereen/ALGO1-Info1-2019 et github.com/Naereen/notebooks pour des exemples.
- Comment utiliser Binder, Google Colab ou d'autres outils gratuits en ligne, pour ajouter un lien afin que tout utilisateur consultant vos notebooks Jupyter puisse démarrer un environnement interactif, directement depuis son navigateur Web, pour interagir avec le notebook sans rien avoir à installer.