



# Simuler des séries de k lancers PILE ou FACE en six lignes de code.

publié le 05/09/2022

Pour jouer 5000 fois à pile ou face et recommencer 100 fois, six lignes de codes suffisent pour simuler cette situation.

## Descriptif :

Les dés, le jeu de PILE ou FACE permettent d'aborder la notion de fluctuation de fréquence. Pour observer la stabilisation de la fréquence d'une issue, un grand nombre de répétitions est nécessaire. Un tableur permet de réaliser ce type de simulation mais l'augmentation du nombre de cas rend la lecture difficile. Un code python, ne présente pas ce désavantage.

Les conseils de bonnes pratiques de codage en Python sont disponibles dans un œuf de Pâques du langage. Pour le voir, taper :

```
import this
```

La simulation repose sur les modules random et collection. On se restreint à trois fonctions :

```
from random import choice, choices
from collections import Counter
```

## Lancer un dé à douze faces

Un unique lancer d'un dé à douze faces équiprobable est simulé par les deux lignes :

```
faces_dé = [1,2,3,4,5,6,7,8,9,10,11,12]
choice(faces_dé)
```

Répétons les lancers d'un dé équilibré. Pour cela, utilisons la fonction `choices` avec un `s`

- qui permet de donner un poids relatif à chaque issue
- de fixer un nombre de répétitions, ici `k=1`.

Cette fonction renvoie ses résultats dans une liste (d'où les crochets). en suivant deux conseils du [zen du python](#) (taper `import this` dans une cellule ou dans une console) :

- Il devrait y avoir une - et de préférence une seule - manière évidente de le faire.
- Plat est mieux que imbriqué. : on évite d'écrire une boucle si possible.

On va donc écrire pour un unique lancer (`k=1`) d'un dé à douze face :

```
poids = [1,1,1,1,1,1,1,1,1,1,1,1]
choices(faces_dé, poids, k = 1)
```

## Lancer un osselet

Pour jouer aux osselets, on lance un astragale ou sa réplique, dont les quatre faces (dos, ventre, vautour, oreille) ne sont pas équiprobables :

Face visible	Probabilité
--------------	-------------

dos	0.37
ventre	0.37
vautour	0.13
oreille	0.13

Compléter les deux lignes de code ci-dessous pour simuler 20 lancers d'un osselet (Les poids de chaque face seront des nombres entiers exprimant la fréquence en % d'obtention de chacune des faces) :

```
poids = [ , , , ]
choices([ "...", "...", "...", "..."], poids, k = ...)
```

## 1000 épreuves de PILE ou FACE

La fonction *Counter* du module *collections*, donne les effectifs de chaque issues d'une liste. Elle renvoie un dictionnaire, ici *issues*. Un dictionnaire associe des clés à des valeurs :

- les clés sont P ou F correspondants à PILE ou FACE.
- les valeurs sont les effectifs des clés P et F

```
N_Lancers = 1000
faces_pièce = ['P','F']
lancers = choices(faces_pièce, [1,1], k = N_Lancers)
issues = Counter(lancers)
issues
```

On obtient par exemple :

```
Counter({'P': 517, 'F': 483})
```

## Fréquence du côté PILE sur 1000 épreuves de PILE ou FACE

Simulons 1000 épreuves de lancers à PILE ou FACE et calculons la fréquence observée du coté PILE. On triche un peu, le code ci-dessous fait sept lignes (sans compter l'importation des fonctions). On aurait pu faire moins pour atteindre six lignes :

```
N_Lancers = 1000
faces_pièce = ['P','F']
lancers = choices(faces_pièce, [1,1], k = N_Lancers)
issues = Counter(lancers)
N_P = issues['P']
f_P = N_P / N_Lancers
print('La fréquence des côté PILE est:',f_P,' pour ', N_Lancers,' lancers.')
```

Une sortie de ce code est par exemple :

```
La fréquence des côté PILE est: 0.482 pour 1000 lancers.
```

## Faire 20 séries de 100 épreuves PILE ou FACE

A priori, on ne va pas couper à l'usage d'une boucle *for* répéter 20 fois les épreuves de lancers. Le code est le suivant :

```
N_séries = 20
N_Lancers = 100
for lancer in range(N_séries):
    Lancers = choices(['P','F'], k = N_Lancers)
    issues = Counter(Lancers)
```

```

N_P = issues['P']
f_P = N_P / N_Lancers
print('Série:',lancer,' . La fréquence des côté PILE est:',f_P,' pour ', N_Lancers,' lancers.')

```

La sortie du code est :

```

Série: 0 . La fréquence des côté PILE est: 0.5 pour 100 lancers.
Série: 1 . La fréquence des côté PILE est: 0.45 pour 100 lancers.
Série: 2 . La fréquence des côté PILE est: 0.49 pour 100 lancers.
Série: 3 . La fréquence des côté PILE est: 0.54 pour 100 lancers.
Série: 4 . La fréquence des côté PILE est: 0.5 pour 100 lancers.
Série: 5 . La fréquence des côté PILE est: 0.51 pour 100 lancers.
Série: 6 . La fréquence des côté PILE est: 0.55 pour 100 lancers.
Série: 7 . La fréquence des côté PILE est: 0.57 pour 100 lancers.
Série: 8 . La fréquence des côté PILE est: 0.45 pour 100 lancers.
Série: 9 . La fréquence des côté PILE est: 0.51 pour 100 lancers.
Série: 10 . La fréquence des côté PILE est: 0.47 pour 100 lancers.
Série: 11 . La fréquence des côté PILE est: 0.55 pour 100 lancers.
Série: 12 . La fréquence des côté PILE est: 0.54 pour 100 lancers.
Série: 13 . La fréquence des côté PILE est: 0.56 pour 100 lancers.
Série: 14 . La fréquence des côté PILE est: 0.43 pour 100 lancers.
Série: 15 . La fréquence des côté PILE est: 0.44 pour 100 lancers.
Série: 16 . La fréquence des côté PILE est: 0.55 pour 100 lancers.
Série: 17 . La fréquence des côté PILE est: 0.55 pour 100 lancers.
Série: 18 . La fréquence des côté PILE est: 0.5 pour 100 lancers.
Série: 19 . La fréquence des côté PILE est: 0.49 pour 100 lancers.

```

## Encapsulation du code dans une fonction Python

Dans le code précédent, la sortie est obtenue avec un `print()`, pour visualisation. le numéro d'une série est associé à sa fréquence. Un dictionnaire est indiqué pour enregistrer cette association.

On a intérêt à encapsuler le code simulant la réalisation de **k** séries de **N** lancers PILE ou FACE :

```

def simulation_séries_de_lancés(N_séries = 20, N_lancers = 100):
    fréquence_P_séries = {}
    poids = [1, 1]
    for s in range(N_séries):
        Lancers = choices(['P','F'], poids, k = N_lancers)
        issues = Counter(Lancers)
        N_P = issues['P']
        f_P = N_P / N_lancers
        fréquence_P_séries[s] = f_P
    return fréquence_P_séries

```

On peut alors exploiter cette simulation de 20 séries de 5000 lancers :

```
fréquences_P = simulation_séries_de_lancés(N_séries = 20, N_lancers = 5000)
```

Le résultat est un dictionnaire dont les clés sont les séries et les valeurs : les fréquences :

```
fréquences_P.keys()
```

donne

```
dict_keys([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

et :

```
fréquences_P.values()
```

```
dict_values([0.4944, 0.4964, 0.4966, 0.4874, 0.501, 0.4954, 0.5034, 0.493, 0.4974, 0.5082, 0.5032, 0.4848, 0.4994,
0.4994, 0.495, 0.5026, 0.5016, 0.495, 0.4936, 0.4952])
```

Représentons graphiquement les fluctuations de la fréquence des issues du côté PILE. On importe le module pyplot :

```
from matplotlib import pyplot as plt  
puis :
```

```
plt.figure(figsize = (10,4))  
plt.grid(True)  
plt.xlim(-1, 20)  
plt.xticks(list(fréquences_P.keys()))  
plt.xlabel('Série')  
plt.ylim(0.45, 0.55)  
plt.ylabel('fréquence PILE')  
  
plt.scatter(fréquences_P.keys(), fréquences_P.values())  
plt.show()
```

On obtient (cliquer pour zoomer) :

Une expérience est constituée de k séries de n lancers PILE ou FACE. On réalise trois expériences de 100 séries avec n lancers variant de 50 à 5000 répétitions P/F :

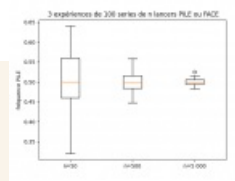


```
experience_1 = simulation_séries_de_lancés(N_séries = 100, N_lancers = 50)  
experience_2 = simulation_séries_de_lancés(N_séries = 100, N_lancers = 500)  
experience_3 = simulation_séries_de_lancés(N_séries = 100, N_lancers = 5000)
```

On affiche les diagrammes en boîte qui correspondent :

avec le code ci-dessous :

<https://stackoverflow.com/questions/52273543/creating-multiple-boxplots-on-the-same-graph-from-a-dictionary>




```
fig, ax = plt.subplots()  
#ax.boxplot(my_dict.values())  
#ax.set_xticklabels(my_dict.keys())  
  
Data = [ list(experience_1.values()), list(experience_2.values()), list(experience_3.values())]  
  
ax.title.set_text(" 3 expériences de 100 series de n lancers PILE ou FACE")  
ax.boxplot(Data)  
ax.set_ylabel('fréquence PILE')  
ax.set_xticklabels(["n=50", "n=500", "n=5 000"])  
  
plt.show()
```

La diminution de la dispersion de la fréquence de l'issue PILE est visible avec l'augmentation du nombre de lancers (50, 500, 5000).

Le notebook est [accessible depuis Capytale](#).

Ou téléchargeable.

 [fre\\_quence\\_de\\_fluctuation\\_dans\\_le\\_jeu\\_pile\\_ou\\_face\\_ipynb](#) (Zip de 115 ko)

Ou consultable ci-dessous :

PILE\_FACE-Fréquences.ipynb hosted with ♥ by GitHub

[view](#)  
[raw](#)

## Portfolio



## Document joint

 [seconde\\_proba\\_03\\_fluctuation\\_et\\_algo](#) (PDF de 423.8 ko)



**Académie  
de Poitiers**

Avertissement : ce document est la reprise au format pdf d'un article proposé sur l'espace pédagogique de l'académie de Poitiers.  
Il ne peut en aucun cas être proposé au téléchargement ou à la consultation depuis un autre site.