Simulation d'expériences aléatoires

Dans ce thème, nous allons utiliser le module `numpy.random` avec l'alias `rd` :

```
>>> import numpy.random as rd
```

Il convient au préalable d'initialiser le générateur de nombres aléatoires :

```
>>> rd.seed()
```

■ Comment simuler une expérience aléatoire ?

SF1 Simuler à partir d'un tirage uniforme

La fonction `random` fournit un nombre « aléatoire » dans $[0,1[$ (selon une loi uniforme).
Plus simplement, `randint(a, b)` fournit un entier de $[\![a, b-1]\!]$ avec équiprobabilité.

Exemples

1 ► Écrivons une fonction simulant le lancer d'une pièce équilibrée donnant *pile* ou *face*.

```
def piece():
    resultat = rd.random()
    if resultat < 0.5:
        return "pile"
    else:
        return "face"
```

2 ► Écrivons une fonction simulant le lancer d'un dé équilibré à 6 faces.

```
def de():
    a = rd.random()
    if a < 1/6:
        return 1
    elif a < 2/6:
        return 2
    elif a < 3/6:
        return 3
    elif a < 4/6:
        return 4
    elif a < 5/6:
        return 5
    else:
        return 6
```

ou de façon plus concise :

```
from math import floor

def de_bis():
    return floor(6 * rd.random() + 1)
```

Mais pour cet exemple, il est naturel de plutôt utiliser directement :

```
>>> rd.randint(1, 7)
3
>>> rd.randint(1, 7)
6
```

3 ► Écrivons une fonction simulant le lancer d'un dé pipé à 6 faces donnant 6 une fois sur deux et donnant les faces 1 à 5 de façon équiprobable.
Cela revient à considérer un dé équilibré à 10 faces avec 6 sur la moitié des faces :

```
def de_ter():
    a = rd.randint(1, 11)
    if a > 6:
        return 6
    else:
        return a
```

SF2 Simuler une expérience complexe

Pour simuler une expérience fondée sur des lancers de pièces ou dés, tirages de boules, etc. on se contente souvent d'écrire des boucles :

- `for` dans le cas de la répétition déterminée à l'avance d'une certaine expérience ;
- `while` dans le cas où l'arrêt de l'expérience est conditionné à un certain résultat.

Exemple

On considère l'expérience aléatoire suivante.
On dispose de deux dés A et B ; le dé A a 4 faces rouges et 2 faces blanches alors que le dé B a 2 faces rouges et 4 faces blanches. On lance une pièce de monnaie truquée telle que la probabilité d'obtenir *pile* soit $\frac{1}{3}$:

- si l'on obtient *pile* alors on décide de jouer uniquement avec le dé A ;
- si l'on obtient *face* alors on décide de jouer uniquement avec le dé B.

Écrivons une fonction de paramètre n simulant cette expérience avec n lancers (du dé déterminé par le résultat du lancer de la pièce) et renvoyant le nombre fois où l'on a obtenu une face rouge.

On commence par le lancer de la pièce truquée.

```
def piece():
    """ sortie: pile (1) une fois sur 3 et face (2) deux fois sur 3 """
    a = rd.randint(1, 4)
    if a == 3:
        return 2
    else:
        return a
```

On définit maintenant deux fonctions simulant les deux dés.

```
def deA():
    a = rd.randint(1, 7)
    if a <= 4:
        return "rouge"
    else:
        return "blanc"

def deB():
    a = rd.randint(1, 7)
    if a <= 2:
        return "rouge"
    else:
        return "blanc"
```

Notons que l'on aurait pu se contenter d'une seule fonction donnant un résultat dans un tiers des cas et un autre dans les autres cas et l'employer dans trois contextes différents.

On peut alors simuler l'expérience.

```
def simulation(n):
    c = 0  # compteur
    p = piece()
    if p == 1:  # si on tombe sur pile
        for k in range(n):
            lancer = deA()
            if lancer == "rouge":
                c = c + 1
    else:
        for k in range(n):
            lancer = deB()
            if lancer == "rouge":
                c = c + 1
    return c
```

Exercice 1

1. Écrire une fonction simulant un lancer de dé à 4 faces non pipé.
2. Écrire une fonction simulant un lancer de dé à 6 faces pipé donnant 6 avec une probabilité de $\frac{1}{3}$, 5 avec une probabilité de $\frac{1}{2}$ et donnant chacun des numéros de 1 à 4 de façon équiprobable.
3. Un considère un dé équilibré à 6 faces numérotées de 1 à 6. On lance le dé jusqu'à obtenir un numéro impair. Écrire une fonction simulant cette expérience et qui renvoie le numéro impair obtenu.

Exercice 2

Le principe du jeu du craps est le suivant : le joueur lance deux dés ordinaires et non pipés et l'on compte, pour chaque lancer le nombre de points obtenus.

- Le joueur gagne au premier lancer lorsque la somme vaut 7 ou 11.
- Le joueur perd au premier lancer lorsque la somme vaut 2, 3 ou 12.
- Dans les autres cas, le joueur doit relancer les deux dés jusqu'à ce qu'il obtienne la somme initiale ou un 7 :
  - s'il obtient la somme initiale alors il gagne ;
  - s'il obtient un 7 alors il perd.

Écrire une fonction simulant une partie de craps et retournant 1 lorsque la partie est gagnée et 0 sinon. Modifier ensuite la fonction afin qu'elle renvoie également la liste de tous les tirages.

Exercice 3

On considère un jeu en ligne dont la page d'écran affiche une grille à trois lignes et trois colonnes. Une fonction aléatoire place au hasard successivement trois jetons ($\star$) dans trois cases différentes. La partie est gagnée lorsque les trois jetons sont alignés.

	A	B	C
1	$\star$		
2	$\star$		
3		$\star$	

1. Écrire une fonction `grille` créant une grille de façon aléatoire.
2. Écrire une fonction `partie` simulant une partie et renvoyant 1 si la partie est gagnée et 0 sinon.
3. Simuler un grand nombre de parties et estimer la probabilité qu'une grille soit gagnante.

Exercice 4

On considère un entier $n \geqslant 2$ et une urne contenant $2n$ boules numérotées de 1 à n, chaque numéro apparaissant deux fois.
On effectue « au hasard » une succession de tirages simultanés de deux boules de cette urne selon le protocole suivant :

- si les deux boules tirées portent des numéros différents, alors on les remet dans l'urne avant de procéder au tirage suivant ;
- si les deux boules tirées portent le même numéro, alors on s'arrête.

Écrire une fonction simulant une expérience et renvoyant le nombre total de tirages (de deux boules à chaque fois) effectués.

■ Comment étudier des variables aléatoires ?

SF3 Simuler une variable aléatoire

Il s'agit de définir une fonction donnant le résultat d'une expérience aléatoire.

Exemples

1 ► On considère la variable aléatoire X définie de la façon suivante : on effectue deux tirages avec remise dans une urne contenant n boules numérotées de 1 à n et on note X l'écart entre les deux numéros.
Écrivons une fonction `X(n)`, d'argument égal au nombre n de boules, et simulant le résultat d'une expérience.

```
def X(n):
    a = rd.randint(1, n+1)
    b = rd.randint(1, n+1)
    return abs(a-b)
```

2 ► On considère la variable aléatoire X définie de la façon suivante : on effectue des tirages avec remise dans une urne contenant n boules numérotées de 1 à n tant que la suite des numéros obtenus est strictement décroissante (on s'arrête donc lorsque le numéro tiré est supérieur ou égal au précédent) et on note X le nombre total de tirages effectués.
Écrivons une fonction `X(n)`, d'argument égal au nombre n de boules, et simulant le résultat d'une expérience.

```
def X(n):
    a = rd.randint(1, n+1)
    b = rd.randint(1, n+1)
    c = 2  # compteur
    while b < a:
        a = b
        b = rd.randint(1, n+1)
        c = c + 1
    return c
```

On peut aussi choisir de stocker tous les résultats des tirages dans une liste et, par exemple, renvoyer également la liste des tirages.

```
def X_bis(n):
    L = [rd.randint(1, n+1), rd.randint(1, n+1)]
    while L[-1] < L[-2]:
        L.append(rd.randint(1, n+1))
    return L, len(L)
```

Exercice 5

On considère un entier $n \geqslant 2$ et on dispose de deux urnes U et V, l'urne U contenant une boule blanche et $(n-1)$ boules noires et l'urne V contenant une boule noire et $(n-1)$ boules blanches.
Un joueur choisit une urne au hasard pour le premier tirage puis il effectue des tirages d'une boule avec remise de cette boule dans cette urne.
On note X le numéro du tirage où l'on obtient, pour la première fois, une boule noire.
Programmer une fonction simulant la variable aléatoire X.

SF4 Représenter la loi d'une variable aléatoire

On peut représenter la loi d'une variable aléatoire par une liste. On répète un grand nombre de fois l'expérience et, à chaque fois, on ajoute un dans la cas adéquate de la liste. Il reste, à la fin, à diviser chaque case par le nombre d'expériences afin d'avoir les fréquences.

Exemple

Reprenons l'exemple précédent de la variable aléatoire X comptant le nombre de tirages jusqu'à avoir pour la première fois un résultat supérieur ou égal au précédent.

```
def loiX(n, N):
    """ entrée : nombre de boules n et nombre de répétitions N
        sortie : liste de longueur n+2 où la case k contient
                 la fréquence d'obtention de k
    """
    t = [0 for k in range(n+2)]
    for i in range(N):
        res = X(n)
        t[res] = t[res] + 1
    for k in range(len(t)):
        t[k] = t[k] / N
    return t
```

On obtient par exemple :

```
>>> loiX(5, 10000)
[0.0, 0.0, 0.6061, 0.3213, 0.0664, 0.006, 0.0002]
>>> loiX(5, 10000)
[0.0, 0.0, 0.6034, 0.3206, 0.0686, 0.0074, 0.0]
>>> loiX(10, 10000)
[0.0, 0.0, 0.5428, 0.3398, 0.0974, 0.0177, 0.0023, 0.0, 0.0, 0.0, 0.0, 0.0]
>>> loiX(10, 10000)
[0.0, 0.0, 0.5438, 0.3331, 0.1006, 0.0192, 0.003, 0.0003, 0.0, 0.0, 0.0, 0.0]
```

Dans cet exemple, les deux premières cases contiennent toujours 0 puisque $X(\Omega) = [\![2, n+1]\!]$.

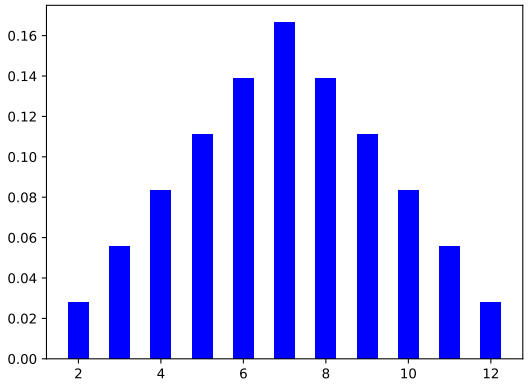
On peut aussi utiliser des diagrammes en bâtons.
Si $x = [x_0, \ldots, x_{n-1}]$ et $y = [y_0, \ldots, y_{n-1}]$ alors on peut construire un diagramme en bâtons avec les coordonnées de X en abscisse et celles de Y en ordonnées à l'aide des instructions présentées dans l'exemple suivant.

Exemple

On lance deux dés usuels équilibrés et on note X la somme des résultats obtenus.
Représentons graphiquement la loi de X.

```
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
x = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
y = [1/36, 2/36, 3/36, 4/36, 5/36, 6/36, 5/36, 4/36, 3/36, 2/36, 1/36]
width = 0.5
plt.bar(x, y, width, color='b')
plt.show()
```

Cela donne la représentation graphique suivante :



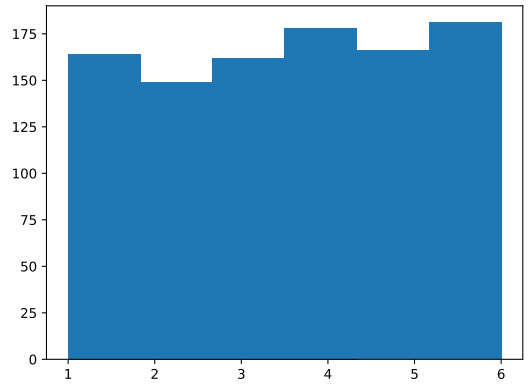
On peut également utiliser `plt.hist` pour obtenir un diagramme en bâtons (on reviendra plus tard sur cette instruction pour tracer des histogrammes).

Exemple

Représentons la répartition des résultats d'un grand nombre de lancers d'un dé usuel équilibré.

```
plt.figure()
N = 1000
t = []
for k in range(N):
    t.append(rd.randint(1, 7))
plt.hist(t, bins = 6)
plt.show()
```

Cela donne la représentation graphique suivante :



SF5 Estimer l'espérance d'une variable aléatoire

Soit X une variable aléatoire réelle, son espérance sous réserve d'existence est définie par :

$$\mathbb{E}(X) = \sum_{x \in X(\Omega)} x\mathbb{P}(X = x).$$

Une première approche pour estimer l'espérance d'une variable aléatoire consiste à simuler la loi de cette variable (donc à répéter un grand nombre de fois l'expérience et à stocker dans une liste les différentes fréquences d'obtention des valeurs) puis à calculer l'espérance à partir de ce tableau.

Bien entendu, le calcul de la variance s'obtient par la même méthode et la formule de König-Huygens :

$$\mathbb{V}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2.$$

Exemple

Considérons à nouveau l'expérience suivante : on effectue deux tirages avec remise dans une urne contenant n boules numérotées de 1 à n et on note X l'écart entre les deux numéros.

On a vu que la fonction suivante permettait de simuler une expérience :

```
def X(n):
    a = rd.randint(1, n+1)
    b = rd.randint(1, n+1)
    return abs(a-b)
```

On simule la loi de X :

```
def loiX(n, N):
    t = [0 for k in range(n)]  # cf. X(Omega) = [0, n-1]
    for k in range(N):  # répétition de N expériences
        res = X(n)      # résultat d'une expérience
        t[res] = t[res] + 1  # on ajoute 1 dans la case adéquate
    return [x/N for x in t]  # on renvoie les fréquences
```

Pour estimer $\sum_{k=0}^{n-1} k\,\mathbb{P}(X = k)$, on multiplie respectivement chacune des cases précédentes par $0, 1, \ldots, (n-1)$ et on somme :

```
def EspX(n, N):
    tab = loiX(n, N)
    esp = 0
    for k in range(n):
        esp = esp + k*tab[k]
    return esp
```

On trouve par exemple :

```
>>> EspX(5, 1000)
1.625
>>> EspX(5, 1000)
1.595
```

La valeur théorique est $\mathbb{E}(X) = \dfrac{(n-1)(n+1)}{3n}$ donc, pour $n = 5$, $\mathbb{E}(X) = 1,6$.

Voici un autre exemple, avec $n = 10$, où l'on est censé obtenir $\mathbb{E}(X) = 3,3$:

```
>>> EspX(10, 10000)
3.3089
>>> EspX(10, 10000)
3.3116
```

Une autre approche repose sur le concept suivant. Considérons une suite de variables aléatoires $(X_n)_{n\in\mathbb{N}^*}$ indépendantes et de même loi, admettant une espérance, notée μ. On appelle *moyenne empirique* de $X_1,\dots,X_n$, la variable aléatoire :

$$\overline{X_n} = \frac{1}{n}(X_1 + \cdots + X_n).$$

La loi faible des grands nombre affirme alors :

$$\forall \varepsilon > 0,\ \mathbb{P}\left(\left|\overline{X_n} - \mu\right| > \varepsilon\right) \xrightarrow[n\to+\infty]{} 0.$$

Concrètement, on obtient donc une valeur approchée de l'espérance d'une variable aléatoire X en calculant la moyenne d'un grand nombre de réalisations de X.

Exemple

Reprenons l'exemple précédent.

```
def EspXbis(n,N):
    esp=0
    for k in range(N)
        esp = esp + X(n)
    return esp / N
```

On obtient par exemple :

```
>>> EspX(5, 1000)
1.614
>>> EspX(5, 1000)
1.559
>>> EspX(5, 1000)
1.542
>>> EspX(10, 1000)
3.323
>>> EspX(10, 1000)
3.303
```

Exercice 6

Déterminer l'espérance du jeu du craps (vu à l'exercice 2).

Exercice 7

Une urne contient n boules numérotées de 1 à n, on en tire trois simultanément et l'on note X le numéro «central» parmi les trois.

1. Écrire une fonction simulant la variable aléatoire X.

2. Simuler et représenter graphiquement, pour quelques valeurs de n, la loi de X.

3. Estimer l'espérance et la variance de X pour quelques valeurs de n.