Python - TP 5
Simulations de phénomènes aléatoires

L'objectif de ce TP est l'utilisation de Python dans la modélisation d'expériences et variables aléatoires. Nous aurons besoin de la bibliothèque `numpy.random` à importer ainsi :

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

Nous aurons également besoin, pour les graphiques, d'importer :

```
import matplotlib.pyplot as plt
```

Pour représenter un histogramme des données contenues dans la liste `Lvaleurs` en fonction de la listes des bornes `Lbornes`, on utilise :

```
plt.hist(Lvaleurs,Lbornes)
```

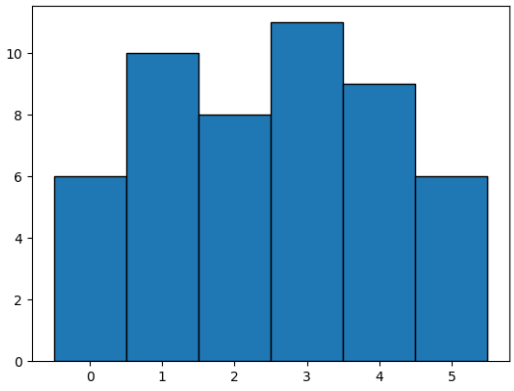
Après la création de l'histogramme, il faut rajouter la ligne `plt.show()` pour le représenter.
Nous aurons besoin des commandes suivantes :

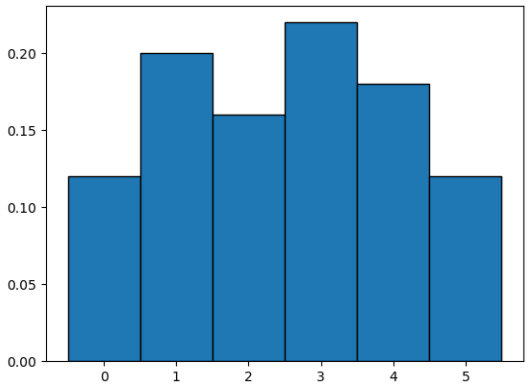
- `rd.random()` renvoie un réel aléatoire de $]0;1[$
- `rd.random(N)` renvoie N réels aléatoires de $]0;1[$ dans un tableau 1 ligne × N colonnes.
- `rd.randint(a,b)` renvoie un entier aléatoire de $[\![a;b[\![$
- `rd.randint(a,b,N)` renvoie N entiers aléatoires de $[\![a;b[\![$ dans un tableau 1 ligne × N colonnes.

L'exécution des lignes ci-dessous renvoie successivement la figure de gauche puis celle de droite.

```
import numpy.random as rd
import matplotlib.pyplot as plt

L=rd.randint(0,6,50)
Labs=[-0.5+k for k in range(0,7)]
plt.hist(L,Labs,edgecolor='k')
plt.show()
plt.hist(L,Labs,edgecolor='k',density=True)
plt.show()
```





Commentaires :

Petite remarque

Par défaut, il s'agit d'un histogramme d'effectifs. Pour obtenir un histogramme de fréquences, on ajoute `density=True` en option dans la commande `plt.hist()`.

✘ Attention !

Comme pour la commande `range(a,b)`, la borne de droite est exclue !

Petite remarque

L'option `edgecolor='k'` permet de tracer le contour des rectangles en noir.

Première expérience.

On lance 3 fois de façon indépendante une pièce donnant PILE avec la probabilité $\frac{1}{3}$. On note X la variable aléatoire égale au nombre de PILE obtenus.

1. Que permet de simuler l'exécution du programme suivant ?

```
import numpy.random as rd

x=rd.random()
if x<1/3:
    piece="PILE"
else:
    piece="FACE"
print(piece)
```

2. Écrire une fonction telle que l'exécution de `simul_X()` renvoie une réalisation de la variable aléatoire X.

3. Réaliser 10000 simulations de la variable aléatoire X et représenter l'histogramme des données ainsi obtenues.

4. De quoi cet histogramme est-il proche ?

5. Comment obtenir une valeur approchée de $\mathbb{E}(X)$?

☞ Pour info...

La **loi faible des grands nombres** stipule que la moyenne d'un grand nombre de réalisations indépendantes d'une variable aléatoire X est proche de $\mathbb{E}(X)$.

Correction de tout :

```
import numpy.random as rd
import matplotlib.pyplot as plt

def simul_X():
    X=0
    for k in range(1,4):
        u=rd.random()
        if u<1/3:
            X=X+1
    return X

LX=[simul_X() for i in range(10000)]

Labs=[-0.5+k for k in range(0,5)]
plt.hist(LX,Labs,density=True,edgecolor='k')
plt.show()

print(sum(LX)/len(LX)) #ou np.mean(LX) : valeur approochée de E(X)
```

Seconde expérience.

On lance successivement et de façon indépendante une pièce donnant PILE avec la probabilité $\frac{1}{3}$ jusqu'à obtenir le premier PILE, et on s'arrête dans tous les cas après le $4^{\text{ème}}$ lancer. On note X la variable aléatoire égale au nombre de lancers effectués.

> **Petite remarque**
> C'est le contexte de l'exercice 12 du chapitre 12.

1. Écrire une fonction telle que l'exécution de `simul_X()` renvoie une réalisation de la variable aléatoire X.

2. Réaliser 10000 simulations de la variable aléatoire X et représenter l'histogramme des données ainsi obtenues.

3. Donner une valeur approchée de $\mathbb{P}([X = 1])$.

4. Écrire un programme permettant d'obtenir une valeur approchée de $\mathbb{E}(X)$.

Correction de tout :

```python
import numpy.random as rd
import matplotlib.pyplot as plt

def simul_X():
    for k in range(1,5): #4 lancers max
        u=rd.random()
        if u<1/3:
            return k #renvoie le rang du PILE
    return k #sinon renvoie 4 à la fin

LX=[simul_X() for i in range(10000)]

Labs=[-0.5+k for k in range(0,6)]
plt.hist(LX,Labs,density=True,edgecolor='k')
plt.show()

print(sum(LX)/len(LX)) #ou np.mean(LX) : valeur approochée de E(X)
```

Troisième expérience.

Soit $n \in [\![2;+\infty[\![$. On considère une urne composée de n boules, numérotées de 1 à n, indiscernables au toucher. On tire simultanément deux boules dans cette urne. On note X_n la variable aléatoire égale au plus grand des deux nombres obtenus.

> **Petite remarque**
> C'est le contexte de l'exercice 17 du chapitre 12.

1. Écrire une fonction telle que l'exécution de `simul_X(n)` renvoie une réalisation de la variable aléatoire X_n.

2. Réaliser 10000 simulations de la variable aléatoire X_5 et représenter l'histogramme des données ainsi obtenues.

3. Écrire un programme permettant d'obtenir une valeur approchée de $\mathbb{E}(X_5)$ et de $\mathbb{V}(X_5)$.

Correction de tout :

```
import numpy.random as rd
import matplotlib.pyplot as plt

def simul_X(n):
    a=rd.randint(1,n+1)
    b=rd.randint(1,n+1)
    while b==a:
        b=rd.randint(1,n+1)
    return max(a,b)

n=5
L=[simul_X(n) for k in range(10000)]
L2=[simul_X(n)**2 for k in range(10000)]
Labs=[-0.5+k for k in range(2,n+2)]
plt.hist(L,Labs,edgecolor='k',density=True)
plt.show()
E=sum(L)/len(L)
V=sum(L2)/len(L2)-E**2
print(E,V)
```

Quatrième expérience.

N désigne un entier naturel supérieur ou égal à 3. On dispose d'une urne contenant (N – 1) balles blanches et une balle noire. On effectue des tirages sans remise dans l'urne, jusqu'à l'obtention de la balle noire. On note X_N la variable aléatoire égale au rang d'apparition de la balle noire.

1. Écrire une fonction Python telle que l'exécution de la commande `simul_X(N)` renvoie une réalisation de la variable aléatoire X_N.

2. En déduire un programme permettant d'obtenir l'histogramme des fréquences sur 10000 réalisations de la variable aléatoire X_N, dans le cas où N = 5, puis N = 10.
Que peut-on conjecturer ?

3. Démontrer cette conjecture. Justifier ensuite que X_N possède une espérance et la calculer.

Correction de tout :

```
import numpy.random as rd
import matplotlib.pyplot as plt

def simul_X(N):
    n=N
    a=rd.randint(1,n+1)
    X=1
    while a!=1:
        n=n-1
        a=rd.randint(1,n+1)
        X=X+1
    return X

N=10
L=[simul_X(N) for k in range(10000)]
Labs=[-0.5+k for k in range(0,N+3)]
plt.hist(L,Labs,edgecolor='k',density=True)
plt.show()
E=sum(L)/len(L)
print(E)
```