P12

SIMULATIONS D'EXPÉRIENCES ALÉATOIRES

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

Exercice 1 On considère une urne contenant 5 boules rouges (numérotées 1,2,3,4,5) et 3 boules jaunes (numérotées 6,7,8).

1. Simuler un tirage dans cette urne.

2. On réalise 7 tirages avec remise dans cette urne. On note X la variable aléatoire qui vaut 0 si aucune boule jaune n'est tirée et, sinon, est égale au numéro du tirage où l'on a obtenu la première boule jaune. Écrire une fonction `SimulationX()` qui réalise une simulation de X.

3. Construire une liste qui contient 20 simulations différentes de X.

Exercice 2 On considère une urne contenant 5 boules rouges (numérotées 1,2,3,4,5) et 3 boules jaunes (numérotées 6,7,8). On va cette fois-ci réaliser 7 tirages sans remise dans cette urne. On note Y la variable aléatoire égale au rang de la première boule jaune (il y en a forcément eu une).

1. Écrire une fonction `SimulationY()` qui réalise une simulation de Y.

2. Construire une liste qui contient 20 simulations différentes de Y.

Exercice 3 Une urne contient initialement 1 boule rouge et 1 boule blanche. On effectue une succession d'épreuves, chaque épreuve étant constituée des trois étapes suivantes :

- on pioche une boule au hasard dans l'urne,
- on replace la boule tirée dans l'urne,
- on rajoute dans l'urne une boule de la même couleur que celle qui vient d'être piochée.

Après n épreuves, l'urne contient donc $n+2$ boules.
Pour tout $n \in \mathbb{N}^*$, on note X_n le nombre de boules rouges qui ont été **ajoutées** dans l'urne (par rapport à la composition initiale) à l'issue des n premières épreuves.

1. Compléter la fonction suivante, qui simule le tirage d'une boule dans une urne contenant x boules rouges et y boules blanches et qui retourne la valeur 0 si la boule est rouge et 1 si elle est blanche.

```
def tirage(x,y):
    alea =
    if                      :
        return 0
    else:
        return 1
```

2. Écrire une fonction `experience(n)` qui réalise une simulation de X_n.

```
def experience(n):
    x = 1
    y = 1
```

3. Réaliser 10 000 simulations de X_5 puis tracer le diagramme en barres correspondant.

```
# Simulations
X = np.array([                    for k in range(         )])

# Diagramme en barres
x =                     # abscisses = valeurs possibles pour X5

y = [0]*len(x)
for i in range(len(y)):
    y[i] = np.sum(X ==       ) # nombre de fois où X = ...

plt.bar(x,y)
plt.show()
```

4. Conjecturer la loi de X_5.