TP12 - Simulations d'expériences aléatoires

In [1]:
```
# import des bibliothèques
import numpy.random as rd
import numpy as np
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.

In [2]:
```
rd.randint(1,9)
```

Out[2]: 6

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.

In [70]:
```
def SimulationX():
    for i in range(1,8):
        tirage = rd.randint(1,9)
        if tirage >= 6 :
            return i
    #Si on arrive ici c'est qu'il n'y a jamais eu de jaune
    return 0

SimulationX()
```

Out[70]: 2

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

In [64]:
```
[SimulationX() for k in range(20)]
```

Out[64]: [1, 2, 1, 2, 1, 1, 4, 3, 5, 3, 2, 5, 1, 0, 7, 5, 1, 1, 2, 4]

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.

In [71]:
```
def SimulationY():
    rouges = 5
    jaunes = 3
    for i in range(1,8):
        tirage = rd.randint(1,rouges+jaunes+1)
        if tirage > rouges :
            return i
        else:
            rouges = rouges - 1

SimulationY()
```

Out[71]: 3

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

In [73]:
```
[SimulationY() for k in range(20)]
```

Out[73]: [1, 4, 2, 6, 3, 6, 2, 1, 1, 1, 3, 2, 2, 1, 1, 1, 4, 2, 2, 1]

Exercice 3 - Écricome 2016

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.

In [77]:
```
def tirage(x,y):
    alea = rd.randint(1,x+y+1)
    if alea <= x:
        return 0 # rouge
    else:
        return 1
```

In [78]:
```
tirage(3,4)
```

Out[78]: 0

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

In [84]:
```
def experience(n):
    x = 1
    y = 1
    X = 0
    for k in range(n): # n épreuves
        t = tirage(x,y)
        if t == 0 : # rouge
            x = x + 1
            X = X + 1 # on ajoute une boule rouge
        else: # blanc
            y = y + 1
    return X
    # ou, à la fin, X = x-1 puis return X

experience(5)
```

Out[84]: 3

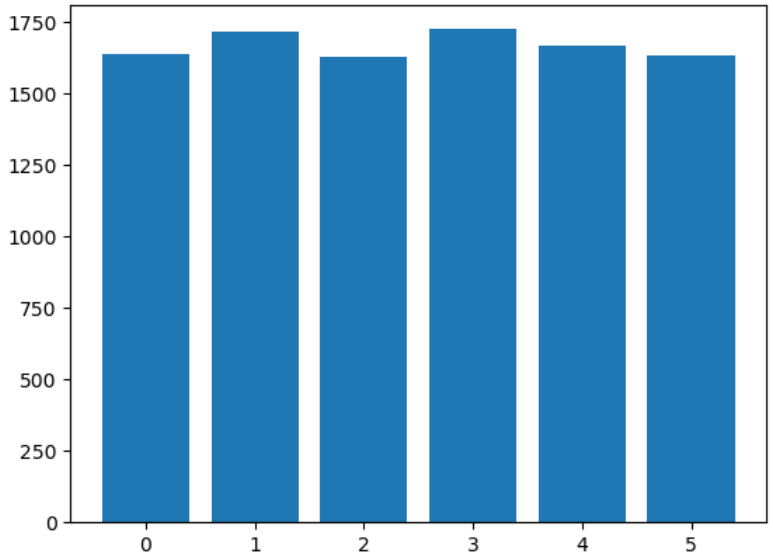
3. Réaliser 10000 simulations de X_5 puis tracer le diagramme en barres correspondant.

In [97]:
```
X = np.array([experience(5) for k in range(10000)])
print(X)
```

```
[3 2 5 ... 1 5 3]
```

In [98]:
```
x = range(0,6)
y = [0]*len(x)
for i in range(len(y)):
    y[i] = np.sum(X == i) # nombre de fois où X=i

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



4. Conjecturer la loi de X_5.