TP11 - Simulation de variables aléatoires finies

In [2]:
```
# import des bibliothèques
import numpy.random as rd
import numpy as np
import matplotlib.pyplot as plt
```

1. Simulation de lois usuelles avec numpy.random (rd)

- rd.random() : nombre aléatoire de $[0,1[$;
- rd.randint(a,b) : loi uniforme sur $[[a,b[[$ (b exclu) ;
- rd.binomial(n,p) : loi binomiale de paramètres n et p ;

In [17]:
```
rd.random()
```

Out[17]:
```
0.5077228366766777
```

In [6]:
```
rd.randint(1,7)
```

Out[6]:
```
6
```

In [10]:
```
rd.binomial(10,0.2)
```

Out[10]:
```
4
```

Pour en simuler plusieurs d'un coup, on ajoute le nombre de simulations voulu

In [13]:
```
rd.randint(1,7,10)
```

Out[13]:
```
array([3, 5, 6, 1, 4, 5, 3, 3, 3, 4])
```

In [14]:
```
rd.binomial(10,0.2,10)
```

Out[14]:
```
array([5, 3, 2, 2, 2, 3, 2, 4, 0, 2])
```

Exercice 1
On considère une urne contenant 5 boules rouges (numérotées 1,2,3,4,5) et 4 boules jaunes (numérotées 6,7,8,9). On réalise 8 tirages avec remise dans cette urne.
On note X_k la variable aléatoire donnant le numéro obtenu au k-ième tirage.

On note R la variable aléatoire égale au nombre total de boules rouges obtenues.

On note J la variable aléatoire égale au nombre total de boules jaunes obtenues.

1. Réaliser une simulation de $X_1, X_2, \ldots, X_8$.
2. Réaliser une simulation de R.
3. Réaliser 10 simulations de J.

In [3]:
```
rd.randint(1,10,8)
```

Out[3]:
```
array([4, 3, 8, 7, 6, 3, 2, 3])
```

In [6]:
```
rd.binomial(8,5/9)
```

Out[6]:
```
3
```

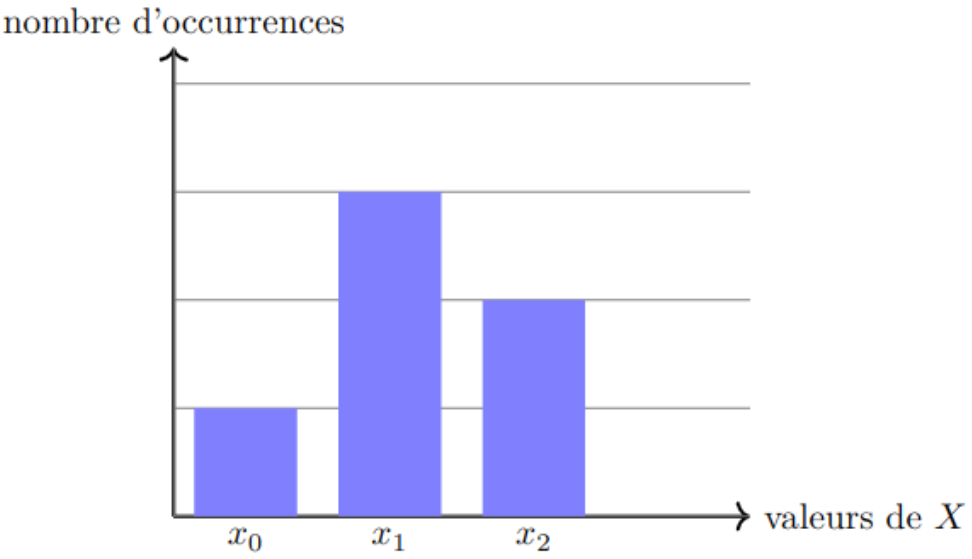
In [7]:
```
rd.binomial(8,4/9,10)
```

Out[7]:
```
array([3, 4, 1, 2, 3, 3, 4, 5, 4, 4])
```

2. Diagramme en barres

On va maintenant tracer les diagrammes en barres obtenus quand on réalise un grand nombre de simulations des lois précédentes.

Un diagramme en barres pour une variable aléatoire X est un diagramme de la forme :



On utilisera la commande

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

où x est la liste des valeurs de X et y est la liste des hauteurs de barres correspondantes.

Si $x = [x_0, x_1, \ldots, x_n]$, alors $y = [y_0, y_1, \ldots, y_n]$ avec

$$y_i = \text{" nombre de fois où } X \text{ vaut } i \text{ "}.$$

Exercice 2 - Diagramme en barres pour X_1

1. Réaliser 1000 simulations de X_1. On appelera la liste créée `X1`.

In [31]:
```
X1 = rd.randint(1,10,10000)
print(X1)
```

```
[8 3 2 ... 9 9 2]
```

2. Définir la liste x des valeurs de X (abscisses du diagramme).

In [28]:
```
x = range(1,10) #[1,2,3,4,5,6,7,8,9]
```

3. Écrire une fonction `nombre(L,k)` qui, étant donnée une liste L et un nombre k, renvoie le nombre de fois où k apparaît dans L.

In [26]:
```
def nombre(L,k):
    n = 0
    for i in range(len(L)):
        if L[i] == k:
            n = n+1
    return n
```

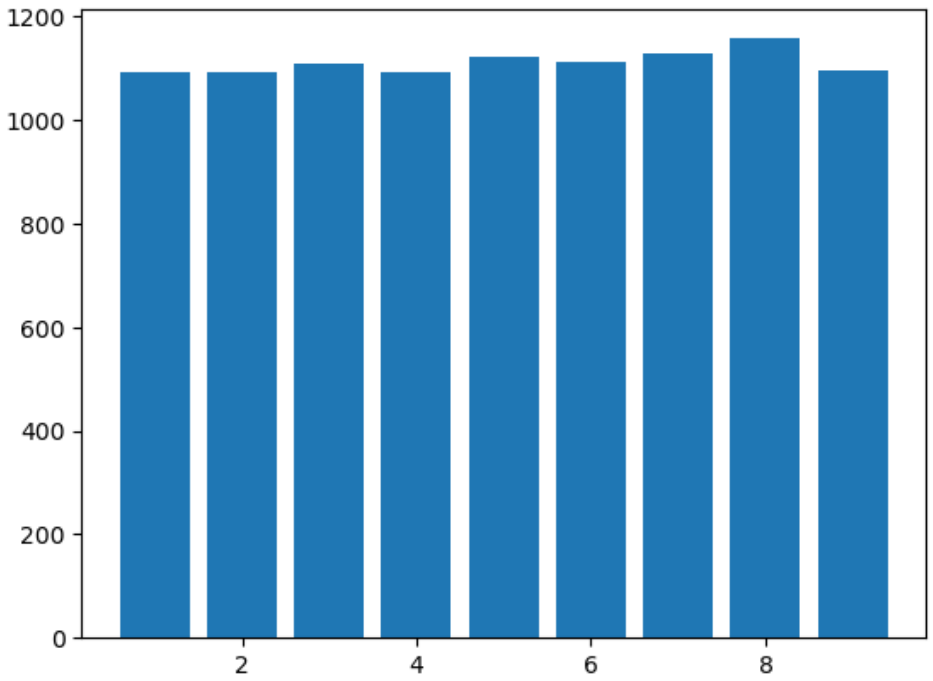
4. Compléter alors le programme suivant pour définir la liste y des hauteurs des barres.

In [32]:
```
y = [0]*len(x) # liste nulle de même taille que x

for i in range(len(y)):
    y[i] = nombre(X1, i+1)
print(y)
```

```
[1092, 1092, 1109, 1092, 1123, 1111, 1127, 1157, 1097]
```

5. Enfin, on trace le diagramme en barres. Reprendre ensuite tout ceci avec 10 000 simulations.

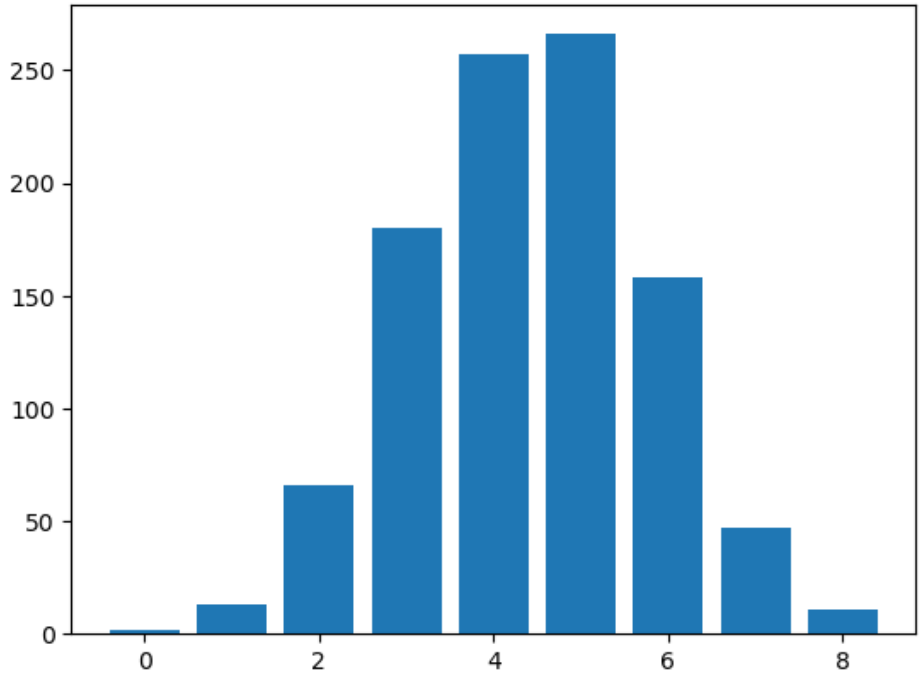
In [33]:
```
plt.bar(x,y)
plt.show()
```



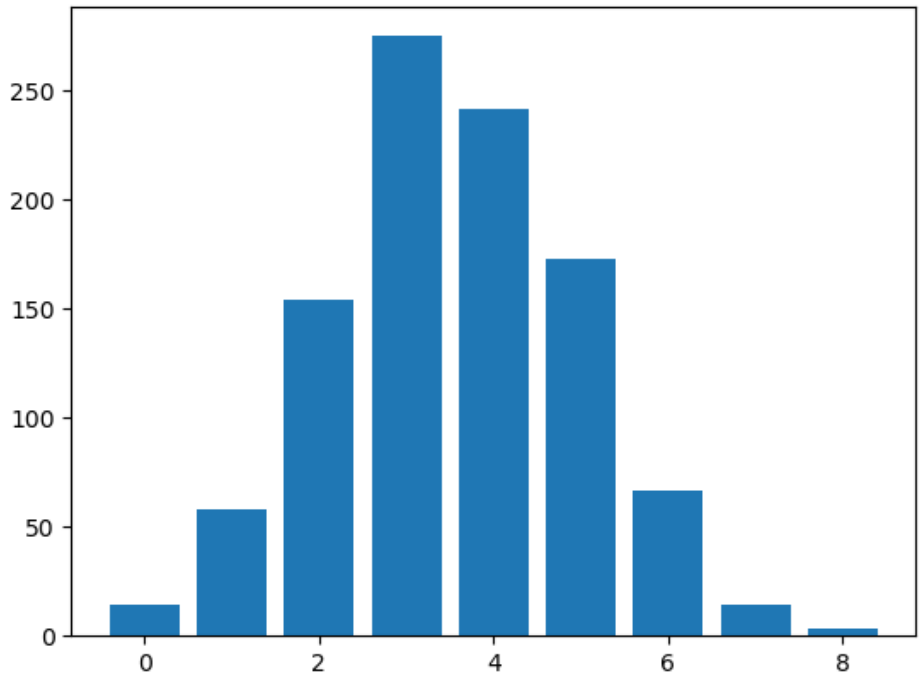
Exercice 3 - Diagramme en barres pour R et J

1. Réaliser 1000 simulations de R puis tracer le diagramme en barres correspondant.
2. Faire de même avec J.

In [37]:
```
R = rd.binomial(8,5/9,1000)
x = range(0,9)

y = [0]*len(x) # liste nulle de même taille que x
for i in range(len(y)):
    y[i] = nombre(R, i)

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



In [36]:
```
J = rd.binomial(8,4/9,1000)
x = range(0,9)

y = [0]*len(x) # liste nulle de même taille que x
for i in range(len(y)):
    y[i] = nombre(J, i)

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



3. Moyenne, variance, écart-type

Ici X désigne une matrice contenant plusieurs simualtions d'une variable aléatoire.

- np.mean(X) : moyenne (proche de l'espérance théorique quand il y a beaucoup de simulations.)
- np.var(X) : variance empirique
- np.std(X) : écart-type empirique

Exercice 4
Calculer la moyenne et la variance empiriques pour X_1, R et J et comparer avec les valeurs théoriques du cours.

In [39]:
```
# X1
# Moyenne empirique
print(np.mean(X1))
# moyenne théorique
print((1+9)/2)
```

```
5.027
5.0
```

In [40]:
```
# R
# Moyenne empirique
print(np.mean(R))
# moyenne théorique
print(8*5/9)
```

```
4.408
4.444444444444445
```

In [41]:
```
# J
# Moyenne empirique
print(np.mean(J))
# moyenne théorique
print(8*4/9)
```

```
3.548
3.5555555555555554
```

In [42]:
```
# X1
# variance empirique
print(np.var(X1))
# variance théorique
print((9**2-1)/12)
```

```
6.640471000000001
6.666666666666667
```

In [44]:
```
# R
# variance empirique
print(np.var(R))
# variance théorique
print(8*5/9*4/9)
```

```
1.9235360000000001
1.9753086419753088
```

In [43]:
```
# J
# variance empirique
print(np.var(J))
# variance théorique
print(8*4/9*5/9)
```

```
2.047696
1.9753086419753088
```

In [ ]:
```

```