Réflexe :
Importer les librairies nécessaires à la génération de l'aléatoire.

Entrée[2]:
```
import numpy as np
import numpy.random as rd
```

TP Python 10 : lois de variables aléatoires discrètes usuelles

Correction

Nous avons vu précédemment que, sans davantage de précision, tout appel de la fonction `rand` renvoyait un nombre réel au hasard entre 0 et 1. Dans ce TP, on s'intéresse à la simulation de variables aléatoires finies abordées en cours.

Partie I : Loi uniforme

Exercice 1
Ecrire à l'aide de la fonction `random` une fonction `unif_segment` prenant en arguments deux entiers relatifs a et b ($a < b$), simulant une loi uniforme sur $[a, b]$.
On pourra s'inspirer de l'exercice du TP précédent qui consistait à simuler un dé équilibré à six faces!

Entrée[122]:
```
def unif_segment(a,b):
    if a>=b :
        print("Erreur, a doit être strictement plus petit que b"
        return
    else :
        x = rd.random()*(b-a) +a
    return x
```

Exercice 2
Ecrire une fonction `tirage` qui étant donnés deux entiers naturels N et k, simule les tirages successifs avec remise de k boules dans une urne contenant N boules numérotées de 1 à N, stocke le résultat de chaque tirage dans une liste `resultats` et renvoie cette liste.

Entrée[131]:
```
def tirage(N,k):
    resultats = []
    # on effectue k lancers
    for j in range(k):
        resultats.append(rd.randint(1,N+1))

    return resultats

tirage(5,23)
```

Sortie[131]:
```
[2, 3, 5, 4, 1, 2, 3, 5, 4, 1, 2, 1, 4, 2, 3, 5, 2, 5, 2, 5, 1, 1, 1]
```

Partie II : Loi de Bernoulli, loi binomiale

Il est possible de simuler une variable aléatoire suivant une loi de Bernoulli de paramètre $p \in]0,1[$ à l'aide de la fonction `rd.random` .

A vous de jouer :
Compléter le programme suivant afin de simuler une variable aléatoire X suivant une loi de Bernoulli de paramètre $p \in]0,1[$.

Entrée[ ]:
```
def Bernoulli(p):
    X = 0
    if rd.random() .... :
        X = 1
    return X
```

→ Cliquez-ici pour la réponse ←

La fonction `rd.binomial` permet de simuler une variable suivant une loi binomiale de paramètres $(n,p) \in \mathbb{N}^* \times]0,1[$. Pour ce faire, on utilise la commande `rd.binomial(n,p)` .

Exercice 3

Dans cet exercice, utilisera le résultat suivant, que nous verrons bien plus tard dans l'année : la somme de n variables aléatoires indépendantes $X_1, \dots, X_n$ suivant la même loi de Bernoulli de paramètre p suit la loi binomiale de paramètres (n,p).

1. Ecrire une fonction `Binomiale1` qui simule une variable aléatoire de loi binomiale de paramètre $(n,p) \in \mathbb{N}^* \times]0,1[$ en utilisant la fonction `rd.random` .

2. Ecrire une fonction `Binomiale2` qui simule une variable aléatoire de loi binomiale de paramètre $(n, p) \in \mathbb{N}^* \times]0, 1[$ en utilisant cette fois-ci la commande `rd.binomial(1,p)` .

Entrée[163]:

```
def Binomiale1(n,p):
    S = 0
    for k in range(n):
        X = 0
        if rd.random()<=p:
            X = 1
        S = S+ X

    return S

def Binomiale2(n,p):
    S = 0
    for k in range(n):
        S = S + rd.binomial(1,p)

    return S
```

Exercice 4
A l'aide de la commande `rd.binomial(1,p)` , écrire une fonction qui prend en argument un réel $p \in]0, 1[$, simule des réalisations de Bernoulli de paramètre p jusqu'au premier succès, et retourne le rang N de ce premier succès. Tester la fonction quelques fois.

On dit qu'une telle variable aléatoire suit une loi géométrique de paramètre p. Nous la reverrons plus tard au second semestre.

Entrée[7]:

```
def geometrique(p):
    N = 0
    while rd.binomial(1,p) !=1 :
        N = N +1
    return N

# testons la fonction vingt fois
for k in range(20):
    print(geometrique(0.3),geometrique(0.7))
```

```
4 0
1 0
5 2
3 1
0 2
0 1
3 0
1 1
4 1
3 0
0 2
3 0
7 0
3 0
4 0
4 0
7 0
6 1
2 0
5 0
```

Exercice 5 : Déplacement d'un mobile

On considère un mobile se déplaçant sur l'axe (Oy). À l'instant $n = 0$, le mobile est situé en $(0, 0)$. Le mobile se déplace selon la règle suivante : à l'instant $n + 1$, le mobile se déplace avec probabilité $\frac{1}{2}$ d'un cran vers le haut, et probabilité $\frac{1}{2}$ d'un cran vers le bas. On note Y_n son ordonnée à l'étape n, $n \in \mathbb{N}$.

1. Écrire une ligne de code qui, à l'aide d'une variable aléatoire suivant une loi de Bernoulli de paramètre $\frac{1}{2}$, simule la réalisation d'une variable aléatoire Y vérifiant $\mathbb{P}(Y = 1) = \frac{1}{2}$ et $\mathbb{P}(Y = -1) = \frac{1}{2}$.
2. À l'aide de la question précédente, écrire une fonction `mobile` prenant en argument un entier naturel n et qui renvoie la liste des ordonnées successives après une réalisation de l'expérience à l'étape n.

Entrée[35]:

```
Y = 2*(rd.binomial(1,1/2)-1/2)


def mobile(n):
    # le mobile commence avec une ordonnée y = 0
    # on stocke cette ordonnée la variable pos_y
    pos_y = [0]

    # on effectue n sauts de puces, et on met à jour
    # l'ordonnée du mobile
    for k in range(1,n+1):
        saut = 2*(rd.binomial(1,1/2)-1/2)
        new_pos = pos_y[k-1] + saut
        pos_y.append(new_pos)

    return pos_y

z = mobile(50)
print(len(z))
```

```
51
```

3. On note T le premier rang $n \in \mathbb{N}$ où le mobile retourne à l'origine $(0, 0)$. Compléter le code suivant pour qu'il simule une réalisation de T.

Entrée[ ]:

```
Y = 2*(rd.binomial(1,1/2)-1/2)
T = 1
while ...:
    Y = Y + ...
    T = ...
print(...)
```

Entrée[17]:

```
# On initialise le premier mouvement du mobile
Y = 2*(rd.binomial(1,1/2)-1/2) # 1 si succès, -1 si echec
T = 1
# on continue d'avancer tant que l'ordonnée Y ne revient pas à 0
while Y !=0:
    # Tant que l'ordonnée Y ne revient pas à 0, on simule une iss
    # et on l'ajoute à notre ordonnée précédente.
    Y = Y + 2*(rd.binomial(1,1/2)-1/2)
    # On incrémente T, puisqu'on a répété l'expérience
    T = T + 1
# Une fois que le mobile est retourné en (0;0), on affiche le ra
print(T)
```

```
4
```

4. Sur un même graphique représenter la trajectoire du mobile pour cinq réalisations de cette expérience, jusqu'à l'étape 50.

Entrée[41]:

```
import matplotlib.pyplot as plt

etapes = [k for k in range(51)]
print(len(etapes))
traj1 = mobile(50)
print(len(traj1))
traj2 = mobile(50)
traj3 = mobile(50)
traj4 = mobile(50)
traj5 = mobile(50)

plt.clf()
plt.plot(etapes,traj1)
plt.plot(etapes,traj2)
plt.plot(etapes,traj3)
plt.plot(etapes,traj4)
plt.plot(etapes,traj5)
plt.legend(["trajectoire 1","trajectoire 2","trajectoire 3","traj
plt.show()

```

```
51
51
```

Exercice 6 : Une loi binomiale détraquée
Soit X une variable aléatoire de loi binomiale de paramètres $(n, p) \in \mathbb{N}^* \times]0, 1[$, avec $n \geq 2$. Les résultats de X sont affichés par un compteur détraqué :

- le compteur affiche la valeur correcte de X lorsque X prend une valeur entre 1 et $n - 1$.
- le compteur affiche un nombre entier tiré uniformément entre 1 et $n - 1$ lorsque X prends la valeur 0 ou n.

On note Y la variable aléatoire égale au nombre affiché par le compteur.
Écrire une fonction Python qui prend en argument les paramètres n et p et qui simule la variable aléatoire Y.

Entrée[108]:

```
def binomiale_detra(n,p):
    X = rd.binomial(n,p)
    if 1<=X <=n-1 :
        Y = X
    else :
        Y =rd.randint(1,n)
    return Y

binomiale_detra(7,0.5)
```

Sortie[108]: 3

Partie III : Pour s'entraîner chez soi : Ecricome 2015

Exercice : à partir de Ecricome 2015

N est un entier supérieur ou égal à 3.
Une urne contient une boule noire et $N-1$ boules blanches.
On effectue des tirages **sans remise** dans l'urne jusqu'à l'obtention de la boule noire.
On note X la variable aléatoire qui prend pour valeur le nombre de tirages nécessaires pour l'obtention de la boule noire. On notera pour tout entier naturel i non nul :

- N_i l'événement "on tire une boule noire lors du i-ième tirage" ;
- B_i l'événement "on tire une boule blanche lors du i-ième tirage".

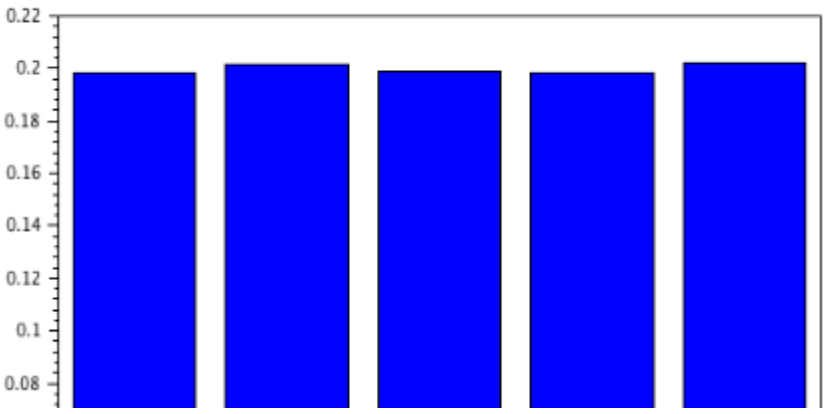
1. On simule 10000 fois cette expérience aléatoire.
   Recopier et compléter le programme Python suivant pour qu'il affiche l'histogramme donnant la fréquence d'apparition du rang d'obtention de la boule noire.

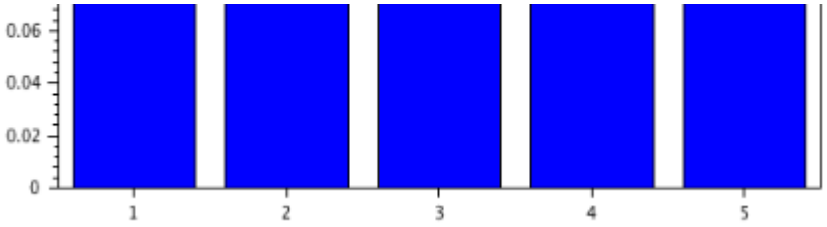
Entrée[ ]:

```
# programme à compléter

import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
N = int(input('Donner un entier naturel non nul'))
S = np.zeros(N)
for k in range(0, 10000) :
    i = 0
    M = N
    while ................. :
        i = i + 1
        M = .......
    S[i] = S[i]+ 1
print(S / 10000)
x=np.arange(1,N+1,1)
plt.bar(x,S/10000)
plt.show()
```

2. On exécute le programme complété ci-dessus. On entre 5 au clavier et on obtient l'histogramme suivant :





Quelle conjecture pouvez-vous émettre sur la loi de la variable aléatoire X ?

3. Retrouver ce résultat en calculant $P(X = 1)$, $P(X = 2)$, $P(X = 3)$, …