Code de partage avec Capytale : 8309-7242374 **Corrigé**

On utilisera (entre autres) la bibliothèque `numpy.random` sous l'alias `rd`

Exercice 1

On dispose d'une pièce donnant pile avec une probabilité p

1. Par quelle(s) fonction(s) peut-on simuler un tirage de la pièce ?

   La fonction rd.random() répond bien à ce type de question, mais on peut aussi voir le problème comme une loi de Bernoulli et donc utiliser le `rd.binomial(1,p)`, on obtient alors en sortie 1 pour « pile » et 0 pour « face ».

```
def Tirage1(p):
    return rd.binomial(1,p)

def Tirage2(p):
    if rd.random()<p :
        return "pile"
    else:
        return "face"
```

2. Comment simuler une réalisation de la variable X vérifiant $X(\Omega) = \{-1, 1\}$ et $P(X = 1) = p$?

   On peut utiliser des conditions `if ... else ...` ou avoir recours au $(-1)^n$.

```
def X(p):
    return -(-1)**rd.binomial(1,p)
```

3. On considère une suite de tirages de la pièce.
   Compléter la fonction suivante renvoyant la première valeur $n + 1$ pour laquelle on a pile aux rangs n et $n + 1$. On dira qu'on obtient un « double pile » aux rangs n et $n + 1$

   Avec la modélisation choisie, le « double pile » correspond à $u + v = 2$, donc on continue les lancers tant que cette condition n'est pas réalisée en mettant à jour les valeurs de u et v.

```
def DoublePile(p):
    u=X(p); v=X(p)
    # u, v correspondent aux deux premiers tirages
    # n=2 # nombre de tirages effectués.
    while u+v<2 :
        (u, v) = (v, X(p))
        n = n+1
    return n
```

4. Pour obtenir une valeur approchée de l'espérance d'une variable aléatoire Z, on pourra simuler la loi un « grand » nombre de fois (en pratique : 1 000 ou 10 000 fois), et en prendre la moyenne. Si les simulations sont placées dans une liste `z`, on pourra utiliser la commande `np.mean(z)`. Estimer $E(Y)$ où Y donne le rang d'apparition du premier « double pile ». On fera des tests en prenant une valeur pour p ($= 0,5$ par exemple)

   Nota bene : la valeur théorique est $\dfrac{1+p}{p^2}$

   On répète l'expérience 1 000 fois (avec $p = \dfrac{1}{2}$ ci-dessous) et on additionne les résultats obtenus, puis on divise par 1 000 pour obtenir la moyenne.

```
moyenne=0
for i in range(1000):
   moyenne = moyenne + X(1/2)
print(moyenne/1000)
```

Exercice 2

Soit $n \in \mathbb{N}^*$
Une urne U contient $2n$ boules, n rouges et n bleues. On pioche n boules et on note X_n le nombre de boules rouges obtenues.

1. Expliquer ce que renvoie le code suivant (faire plusieurs essais).

```
L=rd.randint(1, 11, 5)
M=[l<=5 for l in L]
print('L=', L)
print('M=', M)
print(np.sum(M))
```

   La liste `L` contient 5 simulations de la loi uniforme sur $[1, 10]$, qui correspondent aux 5 tirages et on considère que les 5 premières sont des boules rouges. La liste `M` contient une liste de « vrais » (si la boule tirée était rouge) ou « faux » (sinon). `sum(M)` donne à la fin le nombre de « vrai », autrement dit le nombre de boules rouges tirées.

2. En déduire une fonction simulant la loi de X_n, et la tester.

   On s'inspire du programme précédent en utilisant n à la place de 10

```
def SimX(n):
  L=rd.randint(1, 2n+1, n)
  M=[l<=n for l in L]
  return(np.sum(M))
```

3. Estimer $\mathbf{E}(X_n)$

   Nota bene : la valeur théorique est $\mathbf{E}(X_n) = \dfrac{n}{2}$ (on ne demande pas de le montrer)

   Comme à l'exercice 1, on répète l'expérience un grand nombre de fois (avec $n = 10$ dans l'exemple) :

```
moyenne=0
for i in range(1000):
   moyenne = moyenne + X(10)
print(moyenne/1000)
```

4. Estimer $\mathbf{P}(X_{10} = 5)$

   De manière analogue, on rejoue l'expérience un grand nombre de fois et on calcule la fréquence d'apparition du 5 (on compte le nombre de fois où le résultat est 5) :

```
p=0
for i in range(1,1001):
    if SimX(10)==5:
        p=p+1
print(p/1000)
```