INTERROGATION DE PYTHON N°3

EXERCICE **10 points**

On considère une urne contenant 2 boules noires et une boule blanche. On effectue des tirages successifs avec remise dans cette urne et à chaque tirage on ajoute 2 boules de la même couleur.
On note X_n la variable égale au nombre de boules blanches obtenues au cours des n premiers tirages.

1. Écrire une fonction `tirage(a,b)` qui renvoie 1 si on tire une boule blanche dans une urne composée de a boules noires et b boules blanches et renvoie 0 sinon.

   $\rightsquigarrow$ *On n'oubliera pas d'importer la bibliothèque adaptée en préambule.*

```
1
2  def tirage(a,b):
3
4
5
6
```

2. Compléter la fonction Python permettant de simuler la variable X_n :

```
1  def simule_X(n):
2    a = ...
3    b = ...
4    varX = ...
5    for _ in range(n):
6      if tirage(a,b) == ........ :
7        varX = varX+1
8        b = b+2
9      else:
10       a = ......
11     return ..........
```

3. Compléter la ligne de commande pour obtenir une liste de 100 simulations de la variable X_{10} :

```
1  [.......................... for _ in range( .............. )]
```

4. On rappelle que dans la bibliothèque `numpy` la commande `np.sum(L)` renvoie la somme des éléments de la liste `L`.

```
1  import ........................... as .........
2
3  def fonction(nbsim):
4    L=[simule_X(10) for _ in range( nbsim )]
5    e=np.sum(L)
6    return  e/nbsim
```

Lorsque l'on exécute cette fonction pour `nbsim=1000` on trouve `4,337`.

Interpréter ce résultat en utilisant un langage probabiliste dans le contexte de l'exercice.

INTERROGATION PYTHON N°3 : CORRECTION

EXERCICE **10 points**

On considère une urne contenant 2 boules noires et une boule blanche. On effectue des tirages successifs avec remise dans cette urne et à chaque tirage on ajoute 2 boules de la même couleur.
On note X_n la variable égale au nombre de boules blanches obtenues au cours des n premiers tirages.

```
import random as rd

def tirage(a,b):
  b_blanche = 0
  if r.random() < b/(a+b) :
      b_blanche = 1
    return b_blanche
```

2. Fonction Python permettant de simuler la variable X_n :

```
def simule_X(n):
  a = 2
  b = 1
  varX = 0
  for _ in range(n):
    if tirage(a,b) == 1 :
      varX = varX+1
      b = b+2
    else:
      a = a+2
    return X
```

3. La ligne de commande pour obtenir une liste de 100 simulations de la variable X_{10} :

```
[simule_X(10) for _ in range( 1000 )]
```

4. On considère la fonction suivante :

```
import numpy as np

def fonction(nbsim):
  L=[simule_X(10) for _ in range( nbsim )]
  e=np.sum(L)
  return  e/nbsim
```

La fonction `mystere(nbsim)` simule `nbsim` fois la variable X_{10} et stocke la somme des résultats dans la variable `e`.
Elle affiche en sortie `e/nbsim` correspondant à une moyenne de `nbsim` réalisation de X_{10}.
Cela donne donc une estimation de l'**espérance de la variable** X_{10}**.**
Le résultat `4,337` lorsqu'on l'exécute pour `nbsim=1000` peut s'interpréter de la façon suivante : $\boxed{E(X_{10}) \approx 4,337}$.
Ou bien : lorsqu'on répète un grand nombre de fois (1000 ici) les 10 premier tirages de l'expérience, le nombre moyen de boules blanches obtenues est d'environ 4,337.