T.P. VI - Variables aléatoires (I)

Code Capytale : `5a40-1016474`

I - Ce qu'il faut savoir

* Le module `numpy.random` permet de générer des nombres pseudo-aléatoires. On importe généralement ce module avec l'instruction `import numpy.random as rd`.
* L'appel `rd.randint(a, b)` simule une variable aléatoire de loi unifome sur l'ensemble des entiers {a, a+1, ..., b-1}.
* Plusieurs variables aléatoires indépendantes peuvent être simulées en utilisant l'option `size`. Ainsi, l'appel `rd.randint(a, b, size=n)` simule `n` variables aléatoires indépendantes suivant toutes une même loi unifome sur l'ensemble des entiers {a, a+1, ..., b-1}.

Vous remarquerez en évaluant plusieurs fois le code suivant que la valeur de la simulation dépend du moment où vous l'effectuez. On modélise bien des phénomènes aléatoires.

```
import numpy.random as rd

print(rd.randint(1, 12))

print(rd.randint(1, 12, size=10))
```

Si la loi n'est pas une loi uniforme. On aura généralement des probabilités qui s'expriment sous forme de fractions. On utilise alors une loi uniforme pour simuler cette probabilité.

Exemple 1

La suite d'instructions suivante permet de simuler une variable aléatoire de loi :

k	4	6	8
$\mathbf{P}([X=k])$	12/25	3/25	10/25

On simule une variable aléatoire U de loi uniforme sur $\{1, \ldots, 25\}$. La probabilité que cette variable aléatoire prenne une valeur entre

* 1 et 12 vaut :

$$\mathbf{P}([1 \leqslant U \leqslant 12]) = \mathbf{P}([U=1]) + \mathbf{P}([U=2]) + \cdots + \mathbf{P}([U=12]) = \frac{12}{25}.$$

* 13 et 15 vaut :

$$\mathbf{P}([13 \leqslant U \leqslant 15]) = \mathbf{P}([U=13]) + \mathbf{P}([U=14]) + \mathbf{P}([U=15]) = \frac{3}{25}.$$

* 16 et 25 vaut :

$$\mathbf{P}([16 \leqslant U \leqslant 25]) = \mathbf{P}([U=16]) + \mathbf{P}([U=17]) + \cdots + \mathbf{P}([U=25]) = \frac{10}{25}.$$

```
import numpy as np
import numpy.random as rd

def simulation():
    u = rd.randint(1, 25)
    if u <= 12:
        return 4
    elif u <= 15:
        return 6
    else:
        return 8
```

```
S = np.zeros((5, 1))
for i in range(0, 5):
    S[i] = simulation()

print(S)
```

Vous n'avez pas à les connaître par cœur, mais d'autres fonctions sont disponibles :

* `rd.binomial(n, p, size=N)` renvoie `N` simulations indépendantes d'une loi binomiale de paramètres `n` et `p`.
* `rd.binomial(1, p, size=N)` renvoie `N` simulations indépendantes d'une loi de Bernouill de paramètre `p`.

II - Loi uniforme : Exercices

Exercice 1. (2 urnes) On considère une urne U contenant deux boules blanches et une boule noire indiscernables au toucher, ainsi qu'une urne V contenant une boule blanche et trois boules noires, elles aussi indiscernables au toucher. On effectue une suite de tirages d'une boule dans ces urnes en procédant comme suit :

* le premier tirage a lieu dans l'urne U ;
* tous les tirages s'effectuent avec remise de la boule piochée dans l'urne dont elle provient ;
* si l'on pioche une boule blanche lors d'un tirage, le tirage suivant a lieu dans l'autre urne ;
* si l'on pioche une boule noire lors d'un tirage, le tirage suivant a lieu dans la même urne.

Pour tout entier naturel n, on note X_n la variable aléatoire égale au nombre de boules blanches piochées au cours des n premiers tirages.
Compléter la suite d'instructions suivante pour qu'elle simule la variable aléatoire X_2 :

```
import numpy.random as rd

def simulation():
    tirage1 = rd.randint(1, 4)
    if tirage1 < 3:
        res1 = 1
        tirage2 = rd.randint(1, 5)
        if tirage2 == 1:
            res2 = 1
        else:
            res2 = 0
    else:
        res1 = 0
        tirage2 = ...
        if tirage2 < 3:
            res2 = ...
        else:
            res2 = ...
    return res1 + res2
```

Exercice 2. Une urne U contient 1 boule noire et 3 boules blanches indiscernables au toucher et une urne V contient 2 boules noires et 2 boules blanches indiscernables au toucher.
On lance une pièce équilibrée. Si elle tombe sur le côté Pile, on tire 2 boules successivement et avec remise dans l'urne U et si elle tombe sur le côté Face, on tire 2 boules successivement et avec remise dans l'urne V. On note T la variable aléatoire égale au nombre de fois où on a tiré une boule noire.
Compléter la suite d'instructions suivante pour qu'elle affiche une simulation de la variable aléatoire T.

```
import numpy.random as rd

T = ...

if rd.randint(1, 3) == 1:
    for k in range(1, 3):
        if rd.randint(1, 5) < 2:
            T = T + 1
else :
    ....
    ....
    ....

print("Une simulation de T : ", T)
```

III - Lois non unifomes : Exercices

Exercice 3. Une puce se déplace sur un axe gradué. À l'instant 0, la puce se trouve sur le point d'abscisse 0. À partir de l'instant 0, la puce effectue à chaque instant, un saut vers la droite selon le protocole suivant :

* elle effectue un saut d'une unité vers la droite avec probabilité $\frac{1}{2}$;
* elle effectue un saut de deux unités vers la droite avec la probabilité $\frac{1}{4}$;
* elle effectue un saut de trois unités vers la droite avec la probabilité $\frac{1}{4}$.

Les différents sauts sont supposés indépendants.

1. Compléter la suite d'instructions suivante pour qu'elle simule les 100 premiers déplacements de la puce.

```
import numpy as np
import numpy.random as rd

A = np.zeros((101, 1))
for k in range(0, 101):
    t = np.randint(1, 5)
    if t <= ...:
        A[k] = 1
    elif t == ...:
        A[k] = 2
    else:
        A[k] = 3

print(A)
```

2. On complète la suite d'instruction en y ajoutant les trois commandes suivantes. Quelle sortie graphique obtient-on ?

```
import matplotlib.pyplot as plt
X = np.arange(0, 101)
Y = np.cumsum(A)
plt.plot(X, Y)
plt.show()
```

Exercice 4. Un mobile se déplace sur les points à coordonnées entières positives d'un axe d'origine O. Au départ, le mobile est à l'origine (point d'abscisse 0). Le mobile se déplace selon la règle suivante : s'il est sur le point d'abscisse $k-1$ ($k \in \mathbb{N}^*$) à l'instant n ($n \in \mathbb{N}$), alors, à l'instant $n+1$, il sera

* sur le point d'abscisse k avec la probabilité $\frac{k}{k+1}$,
* ou sur le point d'abscisse 0 avec la probabilité $\frac{1}{k+1}$.

On note U l'instant auquel le mobile se trouve pour la première fois à l'origine (sans compter son positionnement au départ). On convient que U prend la valeur 0 si le mobile ne revient jamais en O.

Compléter la suite d'instruction suivante afin qu'elle calcule et affiche la valeur prise par U lors de l'expérience aléatoire étudiée.

```
import numpy.random as rd
k = 1
hasard = rd.randint(1, k+1)
while hasard ...:
    k = k + 1
    hasard = ...
print("U a pris la valeur :", k)
```