T.P. VI - Variables aléatoires

Code Capytale : `d3ff-2870060`

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 $\{1,\dots,25\}$. La probabilité que cette variable aléatoire prenne une valeur entre

* 1 et 12 vaut :

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

* 13 et 15 vaut :

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

* 16 et 25 vaut :

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

```
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 - Fonction de répartition

Exercice 1. Soit X une variable aléatoire telle que

$$\forall\, k \geqslant 3,\, \mathbf{P}\left(X = k\right) = \frac{1}{2}\left[\left(\frac{4}{5}\right)^{k-2} - \left(\frac{2}{3}\right)^{k-2}\right].$$

1. Compléter le script suivant pour qu'il calcule et affiche les probabilités $\mathbf{P}\left(X = k\right)$ pour $k \in [\![3, 22]\!]$.

```
import numpy as np
import matplotlib.pyplot as plt
X = np.arange(0, 23)
U = np.zeros((23, 1))
for k in range(3, 23):
    U[k, 0] = ...

plt.figure()
plt.plot(X, U, '+')
plt.show()
```

2. Compléter la suite d'instructions suivante pour qu'elle affiche et renvoie les valeurs de la fonction de répartition de X aux points d'abscisses entières comprises entre 3 et 22.

```
F = np.cumsum(...)
plt.figure()
plt.plot(X, F)
plt.show()
```

3. Déterminer graphiquement un réel m tel que $\mathbf{P}\left([X \leqslant m]\right) = \mathbf{P}\left([X \geqslant m]\right)$.

Exercice 2. Soit X une variable aléatoire à densité de fonction de répartition

$$F(x) = \begin{cases} 0 & \text{si } x < 1 \\ 1 - \frac{1}{x^2} - \frac{2\ln(x)}{x^2} & \text{si } x \geqslant 1 \end{cases}$$

1. Compléter le programme suivant pour que la fonction `F` prenne en entrée un réel `x` et renvoie la valeur de $F(x)$.

```
import numpy as np
import matplotlib.pyplot as plt

def F(x):
    if x < 1:
        ...
    else:
        ....
```

2. Qu'obtient-on lors de l'exécution du programme suivant ?

```
A = np.zeros((1, 300))
B = np.zeros((1, 300))

for i in range(0, 300):
    A[0,i] = - 2 + 7 * i / 300
    B[0,i] = F(A[0,i])

plt.plot(A, B)
plt.show()
```

III - Loi discrète uniforme

Exercice 3. (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 4. 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)
```

IV - Loi discrète non unifome

Exercice 5. 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 6. 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)
```

Exercice 7. Un appareil est constitué de deux composants électroniques dont les durées de vie sont supposées indépendantes. Cet appareil ne fonctionne que si au moins un des deux composants est en état de marche.
Soient X_1 et X_2 les variables aléatoires égales à la durée de vie de chacun des composants et Z la variable aléatoire égale à la durée de vie des deux composants. On suppose que X_1 et X_2 suivent la même loi géométrique de paramètre $\frac{2}{5}$.

1. À l'aide de la fonction `rd.rand()`, compléter le script suivant pour que la fonction `geom` simule une variable aléatoire de loi géométrique de paramètre $\frac{2}{5}$.

```
def geom():
    X = ...
    while ...
        X = ...
    return X
```

2. Compléter le script suivant pour qu'il crée une fonction `simulZ` qui simule la variable aléatoire Z.

```
def simulZ():
    X1 = geom()
    X2 = geom()
    if X1 > X2:
        Z = ...
    else:
        Z = ...
    return Z
```

V - Variables aléatoires à densité

Exercice 8. Soit $a > 0$. On considère une variable aléatoire Y de loi uniforme sur $[0, 4a^2]$ et on pose $X = \sqrt{Y}$.

1. Expliquer ce que simule la commande `rd.rand() * 4 * a**2`.

2. En déduire une commande qui permette de simuler la variable aléatoire X.

3. On considère $X_1, \ldots, X_n$ des variables aléatoires indépendantes et suivant toutes la même loi que X. On pose $T_n = \frac{3}{4n} \sum_{k=1}^{n} X_k$. Modifier le script ci-dessous pour qu'il permette de simuler T_n pour $n = 100$.

```
X = np.zeros((100, 1))
for i in range(0, 100):
    X[i] = ...
Tn = ...
print(...)
```

Exercice 9. Soit Y une variable aléatoire de loi exponentielle de paramètre 1 et $X = Y + 1$. On note $S_n = \frac{1}{n} \sum_{k=1}^{n} (X_k - 1)$, où X_1, $X_2, \ldots, X_n$ sont des variables aléatoires mutuellement indépendantes, et de même loi que X.
Compléter la suite d'instructions suivante pour qu'elle fournisse une réalisation de la variable aléatoire S_{50}.

```
import numpy.random as rd

a = ...
Y = np.exp(1, (1, 50))
X = Y + np.ones(...)
S = ...
```