T.P. XIII - Sujets concours

I - Fonctions

Exercice 1. (Graphique) ('D'après Ecricome - 2021 - Exercice 2) On considère la fonction g définie sur $\mathbb{R}$ par :

$$\forall\, x \in \mathbb{R},\ g(x) = \begin{cases} 0 & \text{si } x < 1 \\ \frac{\ln(x)}{x^2} & \text{si } x \geqslant 1 \end{cases}$$

1. Compléter le script suivant pour que la fonction `g` prenne en entrée un réel `x` et renvoie $g(\texttt{x})$.

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

def g(x):
    if x >= 1:
        y = ...
    else:
        ...

X = np.linspace(-4, 8, 100)
Y = [g(x) for x in X]
plt.plot(X, Y)
plt.show()
```

2. Qu'obtient-on lors de l'exécution du script précédent ?

Exercice 2. ('D'après Ecricome - 2022 - Exercice 2) Pour tout entier naturel n, on définit f_n sur $]-1,+\infty[$ par :

$$f_n(x) = x^n \ln(1+x).$$

Compléter la suite d'instructions suivante pour qu'elle affiche le graphe des fonctions f_1, f_5, f_{10}, f_{20}, f_{50} sur l'intervalle $[0,1]$.

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

def f(x):
    ...

for n in ...:
    X = np.linspace(0, 1, 100)
    plt.plot(..., ..., label="f" + str(n))

plt.legend()
plt.show()
```

II - Fonction, Série

Exercice 3. ('D'après Ecricome - 2019 - Exercice 2) On définit sur $]0,+\infty[$ la fonction g par :

$$g(x) = 2x - 1 + \ln\left(\frac{x}{x+1}\right).$$

On admet que g admet une unique solution α sur l'intervalle $[0{,}5,1]$.

1. Compléter l'algorithme de dichotomie ci-dessous afin qu'il affiche un encadrement de α à 10^{-2}.

```
import numpy as np

def g(x):
    return ...

a = 0.5
b = 1

while b - a ... :
```

```
    m = ...
    if g(a) * g(m) <= 0:
        b = ...
    else:
        ...

print(...)
```

2. On considère la suite (u_n) définie par $u_0 = 0$ et

$$\forall\, n \geqslant 1,\ u_n = (2n-1) - g(n).$$

La suite d'instructions suivante construit un vecteur `u` contenant les 50 premiers termes de la suite $(u_n)_{n\geqslant 1}$.

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

U = np.zeros((1, 51))

for n in range(1, 51):
    U[n] = (2 * n - 1) - g(n)

X = np.arange(0, 51)
S = np.cumsum(U)

plt.plot(X, S, '+')
plt.show()
```

Interpréter le contenu du vecteur `S`. Que conjecturer à l'aide du graphique précédent ?

III - Suites récurrentes

Exercice 4. ('D'après Ecricome - 2020 - Exercice 1) On considère les suites (u_n) et (v_n) définies par $u_0 = 0$, $v_0 = 1$ et

$$\forall\, n \in \mathbb{N},\ \begin{cases} u_{n+1} &= -2u_n + v_n \\ v_{n+1} &= 3u_n \end{cases}$$

On pose $A = \begin{pmatrix} -2 & 1 \\ 3 & 0 \end{pmatrix}$ et on constate que $\begin{pmatrix} u_{n+1} \\ v_{n+1} \end{pmatrix} = A \begin{pmatrix} u_n \\ v_n \end{pmatrix}$ soit $\begin{pmatrix} u_n \\ v_n \end{pmatrix} = A^n \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
Compléter le script suivant pour qu'il calcule et affiche les termes u_n et v_n pour $n = 20$.

```
import numpy as np

A = np.array(...)
C = np.aray([[0], [1]])
for k in range(1, n+1):
    C = ...
print(...)
```

Exercice 5. ('D'après BCE ESCP - 2021 - Exercice 3) On considère la fonction f définie sur $\mathbb{R}_+$ par

$$f(x) = \begin{cases} x\,\mathrm{e}^{-1/x} & \text{si } x > 0 \\ 0 & \text{si } x = 0 \end{cases}$$

On définit la suite (u_n) par $u_0 = 1$ et par la relation de récurrence

$$\forall\, n \in \mathbb{N},\ u_{n+1} = f(u_n).$$

On admet que la suite (u_n) est décroissante et converge vers 0.
Compléter les commandes suivantes pour qu'elles affichent le rang n à partir duquel $u_n \leqslant 10^{-3}$.

```
n = 0
u = 1
while u > 0.001:
    u = ...
    n = ...
print(n)
```

IV - Suites récurrentes dépendant de n

Exercice 6. ('D'après BCE ESCP - 2018 - Exercice 2) Soit (I_n) la suite définie par $I_0 = \frac{\mathrm{e}^2-1}{2}$ et

$$\forall\, n \geqslant 0,\ 2I_{n+1} + (n+1)I_n = \mathrm{e}^2\,.$$

Compléter la suite d'instructions suivante pour qu'elle calcule et affiche la valeur de I_n pour $n = 20$.

```
n = ...
I = ...
for k in range(1, n+1):
    I = ...
print(I)
```

V - Suites imbriquées

Exercice 7. ('D'après BCE ESCP - 2022 - Exercice 4) On pose $a_0 = 1$, $b_0 = 2$ et pour tout n entier naturel

$$a_{n+1} = \frac{a_n + b_n}{2} \text{ et } b_{n+1} = \sqrt{a_{n+1} b_n}.$$

1. Compléter la suite d'instructions suivante afin qu'elle affiche a_n et b_n pour $n = 100$.

```
n = ...
a = 1
b = 2
for k in range(1, n+1):
    a = ...
    b = ...
print("a = ", a)
print("b = ", b)
```

2. On peut montrer que la suite $(a_n - b_n)$ est décroissante et que (a_n) et (b_n) convergent vers une même limite ℓ. Compléter la suite d'instructions suivante afin qu'elle calcule le plus petit rang n pour lequel $b_n - a_n \leqslant 10^{-3}$.

```
n = 0
a = 1
b = 2
while ...:
    a = ...
    b = ...
    n = ...
print(n)
```

3. L'exécution de la suite d'instructions affiche la valeur 5. Parmi les quatre réels a_5, b_5, $b_5 - a_5$, $a_6 - a_5$, lesquels sont des valeurs approchées de ℓ à moins de 10^{-3} près ?

4. Quelle est la valeur de l'entier affiché après exécution de la suite d'instructions suivante :

```
n = 0
while 1/2**n > 10**(-3):
    n = n + 1
print(n)
```

VI - Suites récurrentes linéaires

Exercice 8. ('D'après BCE BSB - 2017 - Exercice 1) On définit les suites (u_n), (v_n) et (w_n) par $u_0 = 1$, $v_0 = 0$, $w_0 = 2$ et pour tout n entier naturel :

$$\begin{cases} u_{n+1} &= u_n \\ v_{n+1} &= v_n + 2w_n \\ w_{n+1} &= 2u_n + w_n \end{cases}$$

On pose $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 2 & 0 & 1 \end{pmatrix}$ et $X_n = \begin{pmatrix} 1 \\ v_n \\ w_n \end{pmatrix}$. On admet que, pour tout entier naturel n, $X_n = A^n X_0$.
Compléter le programme suivant pour qu'il calcule et mémorise les premiers termes des suites (u_n), (v_n) et (w_n).

```
import numpy as np

A = ...
u = np.zeros((11, 1))
v = np.zeros((11, 1))
w = np.zeros((11, 1))
u[0] = 1
v[0] = 0
w[0] = 2
X = np.array([[1], [0], [2]])
for i in range(1, 11):
    X = np.dot(A, X)
```

```
    u[i] = 1
    ...
    ...
```

Exercice 9. ('D'après BCE BSB - 2016 - Exercice 1) On définit les suites (a_n), (b_n) et (c_n) par $a_0 = 0$, $b_0 = 1$, $c_0 = 2$ et pour tout n entier naturel :

$$\begin{cases} a_{n+1} &= 4a_n - 6b_n + 2c_n \\ b_{n+1} &= 2a_n - 4b_n + 2c_n \\ c_{n+1} &= -2a_n + 2b_n \end{cases}$$

On pose $A = \begin{pmatrix} 4 & -6 & 2 \\ 2 & -4 & 2 \\ -2 & 2 & 0 \end{pmatrix}$ et $U_n = \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix}$. On constate alors que $U_n = A^n U_0$.
Compléter la suite d'instructions suivante pour qu'elle calcule et affiche la valeur de a_{10}.

```
import numpy as np

A = ...
U = [[0], [1], [2]]
for i in range(1, 11):
    U = np.dot(A, U)
print(...)
```

Exercice 10. ('D'après BCE ESCP - 2019 - Exercice 2) On définit les suite (a_n), (b_n) et (c_n) par $a_0 = 0$, $b_0 = 0$, $c_0 = 1$ puis

$$\forall\, n \in \mathbb{N},\ \begin{cases} a_{n+1} &= b_n - a_n \\ b_{n+1} &= c_n - a_n \\ c_{n+1} &= -a_n \end{cases}$$

Modifier la suite d'instructions suivante pour qu'elle calcule et affiche la valeur de a_{10}.

```
n = ...
a = 0
b = 0
c = 1
for i in range(1, n+1):
    u = a
    a = ...
    b = ...
    c = ...
print(a)
```

Exercice 11. ('D'après BCE BSB - 2018 - Exercice 1)
on définit la suite $(u_n)_{n\in\mathbb{N}}$ par $u_0 = -1$, $u_1 = 1$ et pour tout n entier naturel,

$$u_{n+2} = 3u_{n+1} + 2u_n - 4.$$

1. Compléter la suite d'instructions suivantes pour qu'elle calcule et affiche la valeur de u_{20}.

```
n = ...
v = -1
u = 1
for i in range(2, n+1):
    a = u
    ...
    v = a
print(u)
```

2. On pose $B = \begin{pmatrix} -4 \\ 0 \end{pmatrix}$, $A = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}$ et $X_n = \begin{pmatrix} u_{n+1} \\ u_n \end{pmatrix}$.

a) Montrer que pour tout n entier naturel, $X_{n+1} = AX_n + B$.

b) Compléter la suite d'instructions suivante pour qu'elle calcule et affiche la valeur de u_{20}.

```
n = ...
X = np.array([1, -1])
A = np.array([[3, 2], [1, 0]])
B = np.array([4, 0])
for i in range(1, n+1):
    X = ...
print(X[1])
```

Exercice 12. ('D'après Ecricome - 2020 - Exercice 1) Soit (u_n) la suite définie par $u_0 = 0$, $u_1 = 1$ et

$$\forall\, n \in \mathbb{N}^*,\ u_{n+1} = 7u_n + 8u_{n-1}.$$

Compléter la suite d'instructions suivante pour qu'elle calcule et affiche la valeur de u_{20}.

```
u = 0
v = 1
for k in ...:
    w = u
    u = ...
    v = ...
print(u)
```

VII - Divers

Exercice 13. ('D'après BCE ESCP - 2020 - Exercice 1) On considère la matrice $A = \begin{pmatrix} 1 & 2 & 1 \\ 1/2 & 1 & 2 \\ 1 & 1/2 & 1 \end{pmatrix}$. On admet que

$$\forall\, n \geqslant 3,\ A^n = 3A^{n-1} + \frac{9}{4}A^{n-3}.$$

Modifier la suite d'instructions suivante pour qu'elle calcule et affiche A^n pour $n = 10$.

```
n = ...
I = np.eye(3)
A = np.array(...)
B = np.dot(A, 1)
for k in range(3, n+1):
    C = 3 * B + 9/4 * I
    I = ...
    A = ...
    B = ...
print(B)
```

VIII - Variables aléatoires discrètes finies

VIII.1 - Lois uniformes

Exercice 14. On effectue une succession de lancers (supposés indépendants) d'une pièce de monnaie équilibrée pour laquelle 0 et 1 sont inscrits sur chacune de ses faces et pour laquelle la probabilité d'obtenir 1 vaut $\frac{1}{2}$ et celle d'obtenir 0 vaur également $\frac{1}{2}$.
On considère la variable aléatoire égale au rang d'apparition du premier 1 et la variable Y égale au rang d'apparition du premier 0.
Compléter la suite d'instructions suivante pour qu'elle permette le calcul et l'affichage des valeurs prises par les variables aléatoires X et Y lors de l'expérience.

```
import numpy.random as rd
x = ...
y = ...
lancer = rd.randint(0, 1)
if lancer == 1:
    while lancer == 1:
        lancer = rd.randint(0, 1)
        y = ...
else:
    while lancer == 0:
        lancer = rd.randint(0, 1)
        x = ...
print(x)
print(y)
\end{exercice}

%----------
\subsection{Lois non uniformes}


%%%%%%%%%%%%%%%%%%%%

\begin{exercice}
\source{D'après BCE BSB - 2018 - Exercice 3}
Une urne contient une boule rouge et deux boules blanches. Un joue
```

```
\‘A partir du deuxième tirage, le joueur re\c{c}oit un point à
chaque fois que la couleur obtenue à
un tirage n’est pas celle qui a été obtenue au tirage précédent. Dans le cas contraire il ne re\c{c}oit aucun point.

\qu Expliquer pourquoi la suite d’instructions suivante permet de
\begin{pcode}
import numpy.random as rd

X = rd.binomial(1, 1/3, size=10000)
```

1. Compléter la suite d'instructions suivante pour que le programme simule $n = 10000$ tirages successifs dans l'urne et qu'il affiche le nombre de points marqués par le joueur.

```
import numpy.random as rd

n = 10000
X = rd.binomial(1, 1/3, size=10000)
G = 0
for i in range(1, 10000):
    if X[i] != X[i-1]:
        G += 1

print(G)
```

VIII.2 - Variables aléatoires discrètes infinies

IX - Variables aléatoires à densité

IX.1 - Simulation à partir d'une loi uniforme

Exercice 15. ('D'après Ecricome - 2021 - Exercice 3) 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
```

Exercice 16. ('D'après Ecricome - 2022 - Exercice 3) 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 $X_1, \dots, X_n$ 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(...)
```

IX.2 -

Exercice 17. ('D'après Ecricome - 2019 - Exercice 3) 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 = ...
```

Exercice 18. ('D'après Ecricome - 2020 - Exercice 3) On bureau de poste dispose de deux guichets. Trois clients, notés A, B et C arrivent en même temps. Les clients A et B se font servir tandisque C attend, puis effectue son opération dès qu'un des deux guichets se libère.
On définit les variables aléatoires X, Y et Z égales à la durée en minutes de l'opération des clients A, B et C respectivement lorsqu'ils sont au guichet. On fixe a et b deux réels strictement positifs et on suppose que $X \sim \mathscr{E}(a)$ et $Y \sim \mathscr{E}(b)$. On suppose que X et Y son indépendantes.
On note T la variable aléatoire égale au temps d'attente en minutes du client C avant de parvenir à l'un des guichets.

1. Compléter le code suivant pour qu'il construise une matrice `T` contenant 10 000 réalisations de la variable aléatoire T lorsque $a = 4$ et $b = 5$.

```
def simul(a, b):
    X = rd.exp(scale=a, size=10000)
    Y = rd.exp(scale=b, size=10000)
    T = X
    for k in ...:
        if ....:
            T[k] = ...
    return ...


a = 4
b = 5
T = simul(a, b)
```

2. On considère la fonction suivante :

```
def simul2():
    T = simul(1/2, 1/2)
    Z = rd.exp(1, size=10000)
    n = 0
    for k in range(1, 10001):
        if T[k] + Z[k] > 2:
            n = n + 1
    f = n / 10000
    return f

print(simul2(), simul2(), simul2(), simul2(), simul2())
```

a) Que retourne la fonction `simul2` ?

b) On constate que les résultats renvoyés sont différents mais relativement proches. Sans démonstration, indiquer quel théorème de probabilité assure ce phénomène.

Exercice 19. ('D'après Ecricome - 2017 - Exercice 3) 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 trée 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:
        ....
```

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

for i in range(1:300):
    A[i] = - 2 + 7 * i / 300
    B[i] = F(A[i])

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

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

Exercice 20. ('D'après BCE ESCP - 2018 - Exercice 3) Une urne contient initialement une boule rouge et une boule blanche indiscernables au toucher. On effectue dans cette urne une succession d'expériences aléatoires selon le protocole suivant : on extrait une boule de l'urne et après chaque tirage, la boule tirée est remise dans l'urne et on ajoute dans l'urne une boule de la même couleur que la boule tirée.
On note X_n la variable aléatoire égale au nombre de boules rouges contenues dans l'urne à l'issue de la n^{e} expérience.
Compléter le programme suivant afin qu'il simule une réalisation de la variable aléatorie X_n où $n = 20$.

```
import numpy.random as rd
n = ....
r = 1
b = 1
for k in range(1, n+1):
    if rd.rand() < r / (r + b):
        ...
    else:
        ...
x = ...
print(x)
```

Exercice 21. ('D'après BCE ESCP - 2019 - Exercice 4) Soit Y une variable aléatoire de loi uniforme sur $[0,1[$ et $X = \sqrt{\frac{Y}{1-Y}}$. Compléter la suite d'instructions suivante afin qu'elle simule la variable aléatoire X.

```
Y = ...
X = ...
```

X - Divers

Exercice 22. (???) ('D'après BCE ESCP - 2019 - Exercice 3) Blabla sur K
On suppose que dans une certaine région, pendant une période donnée, seuls deux états météo sont possibles : le beau temps et le mauvais temps. L'étude des bulletins météo du passé laisse penser que le temps qu'il fait un certain jour de cette période dépend du temps qu'il a fait la veille de la façon suivante :

- s'il fait beau un jour donné, la probabilité qu'il fasse beau le lendemain est égal à $\frac{4}{5}$;
- s'il fait mauvais un jour donné, la probabilité qu'il fasse mauvais le lendemain est égal à $\frac{2}{5}$.

On note u_n la probabilité qu'il fasse beau le jour n et $v_n = 1 - u_n$. On peut alors montrer que, en notant $X_n = \begin{pmatrix} u_n & v_n \end{pmatrix}$ et $K = \begin{pmatrix} \frac{4}{5} & \frac{1}{5} \\ \frac{2}{5} & \frac{3}{5} \end{pmatrix}$, alors $X_{n+1} = X_n K$.
On admet que la commande `x = grand(99, 'markov', L, 1)` renvoie un vecteur contenant autant de `1` que de jours de beau temps et autant de `2` que de jours de mauvais temps, et ceci, entre le deuxième et le centième jour.
Compléter la suite d'instructions suivante pour qu'elle renvoie le nombre de jours de beau temps lors des 100 premiers jours de la période considérée, y compris le premier.

```
import numpy as np
K = np.array(...)
x = grand(99, 'markov', K, 1) - 1
n = ...
print("le nombre de jours de beau temps est :", n)
```