TP 17 : Bilan : Quelques questions de différents écrits.

1 Ecricome Voie T 2019

1.1 Extrait 1

Une entreprise de restauration collective propose trois formules de repas à n clients, où n est un entier naturel non nul.

Chacun des clients choisit exactement **une** de ces trois formules au hasard et de façon équiprobable puis envoie un bon de commande à l'entreprise. L'entreprise réceptionne alors pour chacun de ses n clients son bon de commande et lui livre la formule choisie.

On suppose que les choix de formule des clients sont mutuellement indépendants.

Pour tout entier naturel k tel que $1 \leqslant k \leqslant n$, on note :
• A_k l'événement : "après réception du $k^{ième}$ bon, une seule formule a été choisie" ;
• B_k l'événement : "après réception du $k^{ième}$ bon, exactement deux formules ont été choisies" ;
• C_k l'événement : "après réception du $k^{ième}$ bon, les trois formules ont été choisies".
Enfin, on pose pour tout entier k tel que $1 \leqslant k \leqslant n$: $a_k = P(A_k), b_k = P(B_k)$ et $c_k = P(C_k)$.
On a montré précédemment dans l'exercice que la suite (c_n) vérifie :

$$\forall n \in \mathbb{N}^\star,\ c_n = 1 - \frac{2^n - 1}{3^{n-1}}.$$

On considère le script `Python` suivant :

```
n = 1
c = 1-(2**n-1)/3**(n-1)
while c < 0.95 :
    n = n + 1
    c = 1-(2**n-1)/3**(n-1)
print(n)
```

Après exécution, on obtient l'affichage suivant :

```
11.
```

Interpréter le résultat dans le contexte de l'énoncé.

1.2 Extrait 2

Soit g la fonction numérique définie sur l'intervalle $]0;+\infty[$ par :

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

1. (a) Calculer la limite de g en 0 et interpréter graphiquement le résultat.

   (b) Montrer que : $\lim\limits_{x\to+\infty} \ln\left(\frac{x}{x+1}\right) = 0$. En déduire la limite de g en $+\infty$.

2. Étudier le sens de variation de g sur $]0;+\infty[$ et dresser son tableau de variation.

3. (a) Démontrer que l'équation $g(x) = 0$ admet une unique solution α sur l'intervalle $]0;+\infty[$.

   (b) Recopier et compléter l'algorithme de dichotomie ci-dessous écrit à l'aide de `Python` afin qu'il affiche un encadrement de α à 10^{-2} près lorsque a et b sont choisis tels que $\alpha \in [a,b]$.

```
1  def g(x):
2     return ..................
3
4  def approx(a,b):
5          while  b - a  ......  :
6                  m  = ..........
7                  if g(...)*g(m) <= 0 :
8                          b = ........
9                  else :
10                         ........
11         return .........
```

2 Edhec Voie E 2019

2.1 Extrait 1

Soit n un entier naturel supérieur ou égal à 3.
Une urne contient une boule noire non numérotée et $n-1$ boules blanches, dont $n-2$ portent le numéro 0 et une porte le numéro 1. On extrait ces boules au hasard, une à une, sans remise, jusqu'à l'apparition de la boule noire.
Pour chaque i de $[\![1, n-1]\!]$, on note B_i l'événement : « le i-ème tirage donne une boule blanche », on pose $\overline{B_i} = N_i$ et on note X la variable aléatoire égale au rang d'apparition de la boule noire. On note Y la variable aléatoire qui vaut 1 si la boule numérotée 1 a été piochée lors de l'expérience précédente, et qui vaut 0 sinon.

On rappelle qu'en `Python`, la commande `randint(a,b)` simule une variable aléatoire suivant la loi uniforme sur $[\![a, b]\!]$.

1. Compléter le script `Python` suivant afin qu'il simule l'expérience aléatoire décrite dans cet exercice et affiche la valeur prise par la variable aléatoire X.
On admettra que la boule noire est codée tout au long de ce script par le nombre `nB+1`, où `nB` désigne le nombre de boules blanches.

```
1  def Simu(n):
2          nB=n-1
3          X=1
4          u=rd.randint(1,nB+1)
5          while u<nB+1:
6                  nB=------
7                  u=rd.randint(1,------)
8                  X=------
9          print('la boule noire est apparue au tirage numéro',X)
```

2. Compléter les lignes 4 et 9 ajoutées au script précédent afin que le script qui suit renvoie et affiche, en plus de celle prise par X, la valeur prise par Y.

```
1  def SimuBis(n):
2          nB=n-1
3          X=1
4          Y=------
5          u=rd.randint(1,nB+1)
6          while u<nB+1:
7                  nB=------
8                  if u==1 :
9                          Y=------
10                 u=rd.randint(1,------)
11                 X=------
12         print('la boule noire est apparue au tirage numéro',X)
13         print('la valeur de Y est',Y)
```

3 EDHEC voie S 2017

On désigne par n un entier naturel supérieur ou égal à 1 et on considère la fonction f_n définie par :

$$\forall x \in [0,1], \quad f_n(x) = \sum_{k=1}^{n} x^k.$$

1. (a) Compléter la fonction Scilab suivante pour qu'elle renvoie la valeur de $f_n(x)$ à l'appel de $f(x,n)$, où x et n sont donnés par l'utilisateur.

```
1  def f(x,n):
2          S=......
3          for k in range(........):
4                  S=.....
5          return S
```

(b) Transformer pour $x \neq 1$, l'expression de $f_n(x)$ puis en déduire une deuxième façon de déclarer f, en complétant la déclaration suivante où la fonction est toujours nommée f :

```
def f(x,n):
        if x==1 :
                result=.......
         else :
                result=.......
        return result
```

2. On peut montrer que
   - f_n est strictement croissante sur $[0,1]$;
   - $f_n(0) = 0$;
   - $f_n(1) = n$
   - f_n est continue.

   On en déduit que l'équation $f_n(x) = 1$, d'inconnue x élément de $[0,1]$, possède une unique solution α_n dans $[0,1]$.

   On suppose que f_n est déclarée (voir question 1) et on considère les commandes supplémentaires suivantes :

```
n=int(input('entrer la valeur de n : '))
x=0
while f(x,n)<1:
    x=x+0.001
print(x)
```

   Quel est le lien entre le résultat affiché et α_n ?

4 EML 2023 Maths appro

Dans cet exercice, pour $n \in \mathbb{N}^\star$, on pose $S_n = \sum_{k=1}^{n} \frac{1}{k}$. Il a été démontré dans les premières questions que $S_n \sim \ln(n)$. Donc en particulier $\lim\limits_{n\to+\infty} S_n = +\infty$

1. On considère la fonction suivante écrite en langage Python

```
def rang(a):
        k=1
        s=1
        while s<a:
                k=k+1
                s=s+1/k
        return k
```

   Expliquer ce que produit l'appel `rang(50)`

2. Le code suivant

```
from numpy import exp
exp(49)
```

renvoie `1.9073465724950998e+21`

Expliquer rapidement ce que cela laisse penser si l'on fait l'appel `rang(50)`.

5 EDHEC 2023 Maths appro

On effectue des lancers d'une pièce donnant "pile" avec la probabilité $p \in]0,1[$ et "face" avec la probabilité $q = 1 - p$.

On note X la variable aléatoire égale au nombre de "face" obtenus avant le premier "pile".

Selon le principe de la division euclidienne dans $\mathbb{N}$, on admet qu'il existe deux variables aléatoires Q et R définies sur le même espace probabilisé que X et telles que $X = 3Q + R$, avec $X = 3Q + R$, avec $Q(\Omega) = \mathbb{N}$ et $R(\Omega) = \{0, 1, 2\}$.

Par exemple, si X prend la valeur 5, alors Q prend la valeur 1 et R prend la valeur 2.

1. Expliquer pourquoi la fonction suivante renvoie la valeur prise par X lors de l'expérience décrite en début d'exercice

```
def simulX(p):
        X=rd.geometric(p)-1
        return X
```

2. Compléter la fonction Python suivante afin qu'elle renvoie les valeurs prises respectivement par X, Q et R

```
def div(p):
        X=simulX(p)
        Q=........
        R=........
        return(X,Q,R)
```

6 EDHEC 2025 Maths appro

On considère la suite (B_n) définie par :

$$\forall n \in \mathbb{N}, \quad B_n = \frac{1}{4^n}\binom{2n}{n}$$

On a montré que pour tout entier natuel n, $B_n = \prod_{k=1}^{n} \frac{k+n}{4k}$

Compléter alors le code de la fonction `Python B`, prenant en entrée un entier naturel n, et qui renvoie la valeur de B_n.

```
def B(n):
        P=...
        for k in range(....):
                P=....
        return P
```

7 EML 2025 Maths appro

On suppose, pour toutes les questions en langage `Python`*, les bibliothèques usuelles déjà importées sous leur raccourcis habituels.*

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

On étudie les suites (I_n) et (J_n) définies par

$$\forall n \in \mathbb{N}, \quad I_n = \int_0^1 (1-t^2)dt \text{ et } J_n = \int_{-1}^1 (1-t^2)^n dt$$

On a :

- calculé $I_0 = 1$
- montré que (I_n) est décroissante
- montré que $I_n = \dfrac{2n}{2n+1} I_{n-1}$
- montré que $I_n = \dfrac{(2^n n!)^2}{(2n+1)!}$

Compléter la fonction `Python` ci-dessous pour qu'elle calcule et renvoie la valeur de I_n où n est en argument.

```
def I(n):
        i=1
        for k in .........:
                i=............
        return i
```

8 Ecricome 2025 Maths appro

8.1 Extrait 1

Soit $M = \begin{pmatrix} 0 & 3 & 2 \\ 2 & 2 & 4 \\ 5 & 0 & 2 \end{pmatrix}$ On admet que la fonction φ définie par $\varphi(x) = x^3 - 4x^2 - 12x - 28$ est un polynôme annulateur de M

Ecrire une fonction en langage Python, nommée `PolyAnn` prenant en entrée une matrice M et renvoyant `True` si φ est bien un polynôme annulateur de M et `False` sinon.

8.2 Extrait 2

On suppose dans cette partie et pour les questions d'informatique que les bibliothèques suivantes sont importées ainsi :

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

Soit U une variable aléatoire de loi uniforme sur $]0,1[$.

1. Ecrire une fonction en langage `Python`, nommée `Simu1` qui prend en entrée deux réels x et y strictement positifs et qui renvoie une simulation de $U^{x-1}(1-U)^{y-1}$

2. Soit (U_n) une suite de variables aléatoires indépendantes et de même loi uniforme sur $]0,1[$. On note R_n la variable aléatoire définie par $R_n = \frac{1}{n}\sum_{k=1}^{n} U_k^{x-1}(1-U_k)^{y-1}$

   Ecrire une fonction en langage `Python` nommé `Rn` qui prend en entrée deux réels x et y strictement positifs et un entier n et qui renvoie une simulation de R_n.

9 Maths I 2026 appro

9.1 Formulaire

I) Mathématiques générales

np.linspace(a,b,n)	Crée une matrice ligne de n valeurs uniformément réparties entre a et b (inclus)
np.zeros([n,m])	Crée la matrice nulle de taille n x m
np.zeros(n)	Crée la matrice ligne nulle de taille n
np.arange(a,b,eps)	Renvoie la liste des flottants de a à b (b non compris) de pas constant eps.
np.shapte(M)	Donne la taille de la matrice M sous forme d'un tuple (couple)
np.transpose(M)	Renvoie la transposée de M
np.dot (M,P) ; M.dot(P)	Instructions synonymes, évaluent le produit matriciel MP
np.sum(M)	Renvoie la somme de tous les éléments de M.
np.sum(M,axis=i)	Renvoie un vecteur ligne des sommes de chaque colonne de M si i=0 et des sommes de chaque ligne de M si i=1

II) Algèbre linéaire

al.inv(M)	Renvoie l'inverse de la matrice M
al.matrix_rank(M)	Renvoie le rang de la matrice M
al.matrix_power(M,n)	Renvoie la nième puissance de la matrice M

III) Simulations probabilistes

rd.random([q,r])	Simule une réalisation d'une matrice aléatoire de dimension (q,r) dont les coefficients sont des variables aléatoires indépendantes qui suivent la loi U([0,1])
rd.normal(m,d,[q,r])	Simule une réalisation d'une matrice aléatoire de dimension (q,r) dont les coefficients sont des variables aléatoires indépendantes qui suivent la loi N(m,d²)
rd.gamma(m,a,[q,r])	Simule une réalisation d'une matrice aléatoire de dimension (q,r) dont les coefficients sont des variables aléatoires indépendantes qui suivent la loi Gamma(m,a)

IV) Graphiqueq

plt.plot(X,Y,options)	Génère la courbe des points définies par les listes X et Y suivant les options graphiques définies par la chaîne de caractère options
plt.grid	Affiche le quadrillage
plt.show()	Affiche le graphique

9.2 Extrait 1

Pour les programmes `Python`, on dispose d'un petit formulaire à la fin du sujet. On importe aussi les bibliothèques suivantes :

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

Toute fonction Python écrite en réponse à une question de l'énoncé peut être utilisée dans les programmes ou fonction Python demandés par la suite.

Soit $n \in \mathbb{N}^\star$. Une permutation de $[\![1, n]\!]$ est une bijection de $[\![1, n]\!]$ dans $[\![1, n]\!]$. On note $\mathcal{P}_n$ l'ensemble de toutes les permutations de $[\![1, n]\!]$.

Si σ est une permutation de $[\![1, n]\!]$, on représente σ par le n-uplet $(\sigma(1), ...\sigma(n))$. Par exemple, dans le cas $n = 3$, $(2, 3, 1)$ représente la permutation σ de $\{1, 2, 3\}$ définie par : $\sigma(1) = 2$, $\sigma(2) = 3$ et $\sigma(3) = 1$.

On suppose qu'il a été montré dans une question précédente que pour tout $(x_1, ..., x_n) \in \mathbb{R}^n$, il existe $\alpha \in \mathcal{P}_n$ telle que

$$x_{\sigma(1)} \geq ... \geq x_{\sigma(n)}$$

On écrit une fonction `Python` ayant comme entrée un tableau monodimensionnel de réels X (représentant un vecteur) et qui renvoie un tableau Y contenant les mêmes valeurs que X ordonnées dans l'ordre décroissant et une permutation α correspondant à ce qui est dit juste au dessus.

Compléter la fonction `Python` suivante afin que la fonction `permutevecteur()` ayant comme entrée un tableau de valeurs `X` renvoie le couple (`Y,alpha`) ainsi obtenu.

```
1  def permutevecteur(X):
2          n=len(X)
3          alpha=np.arange(0,n,1)
4          Y=X.copy #Y est un nouveau tableau initialisé avec les valeurs de X
5          for i in range(n):
6                  imax=...
7                  for k in range(i,n):
8                          if Y[k]>...:
9                                  imax=...
10                 if imax>i:
11                         ... , ...=... , ...
12                         alpha[i],alpha[imax]=alpha[imax],alpha[i]
13         return Y,alpha
```

9.3 Extrait 2

Ecrire une fonction Python `bistochastique(n,iter)` ayant deux paramètres d'entrée `n` et `iter`, représentant des entiers naturels non nuls, et qui renvoie une matrice bistochastique (terme défini précédemment dans le sujet) construite de la manière suivante :

— on construit au départ une matrice $A^{(0)}$ de taille $n \times n$ dont chaque coefficient $a_{i,j}^{(0)}$ est obtenu en simulant une réalisation de la loi uniforme sur $]0,1[$,

— pour $0 \leq k < iter$:

- On calcule la matrice $A^{(2k+1)}$ obtenue à partir de la matrice $A^{(2k)}$ en divisant chaque ligne de $A^{(2k)}$ par la somme de ses coefficients :

$$A_{i,j}^{(2k+1)} = \frac{A_{i,j}^{(2k)}}{\sum_{l=1}^{n} A_{i,l}^{(2k)}}, \text{ pour } 1 \leq i,j \leq n$$

- On calcule ensuite la matrice $A^{(2k+2)}$ obtenue à partir de la matrice $A^{(2k+1)}$ en divisant chaque colonne par la somme de ses coefficients :

$$A_{i,j}^{(2k+2)} = \frac{A_{i,j}^{(2k+1)}}{\sum_{l=1}^{n} A_{l,l}^{(2k+2)}}, \text{ pour } 1 \leq i,j \leq n$$

- La fonction revoie la dernière matrice $A^{(2k+2)}$ obtenue quand k=`iter -1`. On admet ici que si `iter` est assez grand, on peut considérer que cette matrice est bistochastique.

10 EML 2026 Appro

On considère une suite $(X_n)_{n\geq 1}$ de variables aléatoires indépendantes de même loi, telles que, pour tout $n \geq 1$, $X_n(\Omega) = \{-1, 1\}$ et

$$P(X_n = 1) = P(X_n = -1) = \frac{1}{2}$$

On introduit alors $S_0 = 0$ et pour tout $n \in \mathbb{N}^\star$, $S_n = \sum_{k=1}^{n} X_k$.

1. Ecrire une fonction d'en-tête `def simul_X() :` qui renvoie une simulation de X_1
2. En déduire l'écriture d'une fonction d'en-tête `def simul_S(n) :` qui prend un argument n entier et renvoie une simulation de S_n.

On note T la variable aléatoire qui vaut -1 si, pour tout $n \in \mathbb{N}^\star$, $S_n \leq 0$ est réalisé, ou sinon, qui prend la valeur du plus petit entier $n \geq 1$ pour lequel $S_n > 0$ est réalisé. On suppose qu'il a été démontré que $P(T = 0) = P(T = -1) = 0$

3. Ecrire une fonction d'en-tête `def simul_T():` qui renvoie une simulation de T.
4. On ajoute les commandes ci-dessous dont l'exécution permet l'affichage ci-contre. Que peut-on conjecturer à propos de T ? (*On commencera par préciser ce que fait la fonction* `mystere`.)

```
def mystere(N):
    ech=np.zeros(N)
    for k in range(N):
        ech[k]=simul_T()
    return np.mean(ech)
for k in range(7):
    print(mystere(1000))
```

Affichage Python

```
>>>
67.88
1246.38
285.62
8181.62
31.36
4394.42
117.58
```

11 Ecricome 2026 Appro

Il a été étudié précédemment dans le sujet une suite de polynôme (G_k) définie par $G_0 = 1$, $G_1 = x$, $G_2 = x^2 - \frac{1}{6}$ et

$$\forall k \in [\![1, n-1]\!], \forall x \in \mathbb{R}, \quad xG_k(x) = G_{k+1}(x) + \frac{k(k+3)}{4(k+2)(k+1)} G_{k-1}(x)$$

En Python, on représente un polynôme de $\mathbb{R}_n[x]$ par le tableau numpy de taille $n+1$ de ses coefficients suivant les puissances croissantes. Ainsi le polynôme $P(x) = x^2 + 2x + 3$ est représenté par le tableau de taille $n+1$ $[3, 2, 1, 0, ..., 0]$. On suppose que la biliothèque `numpy` est importée comme suit :

```
import numpy as np
```

1. Compléter la fonction Python suivante afin qu'elle renvoie le tableau correspondant au polynôme xP.

```
1  def Prod(P,n):
2          Q=np.zeros(n+1)
3          for k in range(...):
4                  ....
5          return Q
```

   b) Ecrire une fonction, en langage Python nommée VP qui prend en argument n et un entier naturel k et qui renvoie le tableau numpy associé à G_k