Dans les questions faisant intervenir des instructions en langage Python, on prendra soin d'importer les bibliothèques nécessaires lors de leur première utilisation.
Pour traiter les questions d'informatique, les candidats sont invités à ***se référer à l'annexe fournie en fin de sujet.*** *Ils ne sont pas limités à l'utilisation des seules fonctions mentionnées dans cette annexe.*

Exercice 1

Soit n un entier naturel non nul.
Une urne contient n boules indiscernables au toucher et numérotées de 1 à n. On tire une boule au hasard dans l'urne. Si cette boule tirée porte le numéro k, on place alors dans une seconde urne toutes les boules suivantes : une boule numérotée 1, deux boules numérotées 2, et plus généralement pour tout $j \in [\![1, k]\!]$, j boules numérotées j, jusqu'à k boules numérotées k. Les boules de cette deuxième urne sont aussi indiscernables au toucher. On effectue alors un tirage au hasard d'une boule dans cette seconde urne.
Et on note X la variable aléatoire égale au numéro de la première boule tirée et on note Y la variable aléatoire égale au numéro de la deuxième boule tirée.

1. Reconnaître la loi de X et donner son espérance et sa variance.
2. Déterminer $Y(\Omega)$.
3. Soit $k \in [\![1, n]\!]$.
   (a) On suppose que l'événement $[X = k]$ est réalisé.
   Déterminer, en fonction de k, le nombre total de boules présentes dans la seconde urne.
   (b) Pour tout entier j de $[\![1, n]\!]$, exprimer $P_{[X=k]}(Y = j)$ en fonction de k et j.
   On distinguera les cas $j \leqslant k$ ou $j \geqslant k+1$.
4. (a) Déterminer deux réels a et b tels que, pour tout entier naturel k non nul,
$$\frac{1}{k(k+1)} = \frac{a}{k} + \frac{b}{k+1}.$$
   (b) En déduire que, pour tout élément j de $Y(\Omega)$,
$$P(Y = j) = \frac{2(n+1-j)}{n(n+1)}.$$
5. Justifier que Y admet une espérance et montrer que $E(Y) = \dfrac{n+2}{3}$.
6. Les variables aléatoires X et Y sont-elles indépendantes ?
7. (a) Montrer que $E(XY) = \dfrac{(n+1)(4n+5)}{18}$.
   (b) En déduire que $\mathrm{Cov}(X, Y) = \dfrac{n^2-1}{18}$.
8. (a) Écrire une fonction en langage Python, nommée `seconde_urne`, prenant en entrée un entier naturel `k` non nul, et renvoyant une liste contenant 1 élément valant 1, 2 éléments valant 2,..., j éléments valant j, ..., jusqu'à k éléments valant k.
   Par exemple, l'appel de `seconde_urne(4)` renverra `[1, 2, 2, 3, 3, 3, 4, 4, 4, 4]`.
   (b) Recopier et compléter la fonction en langage Python suivante pour qu'elle prenne en entrée un entier naturel `n` non nul, et qu'elle renvoie une réalisation du couple de variables aléatoires (X, Y).

```
import numpy.random as rd

def simul_XY(n):
    X = _____
    urne2 = seconde_urne(_____)
    nb = len(urne2)
    i = rd.randint(0, nb)
    Y = _____
    return X, Y
```

(c) On considère la fonction en langage Python suivante, prenant en entrée un entier naturel n non nul.

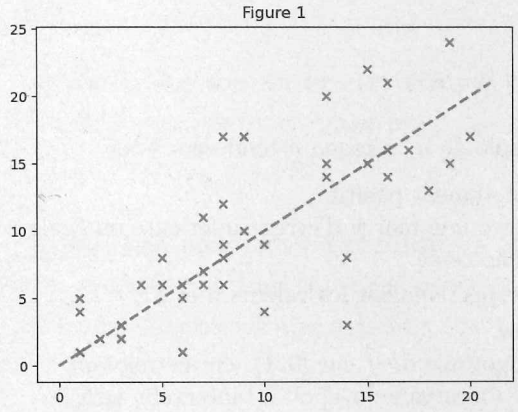
```
def fonction(n):
    liste = [0]*n
    for i in range(10000):
        j = simul_XY(n)[1]
        liste[j-1] = liste[j-1] + 1/10000
    return liste
```

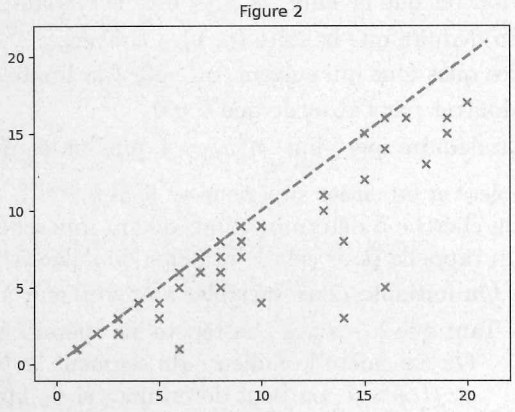
Quelles valeurs les éléments de la liste renvoyée permettent-ils d'estimer ?

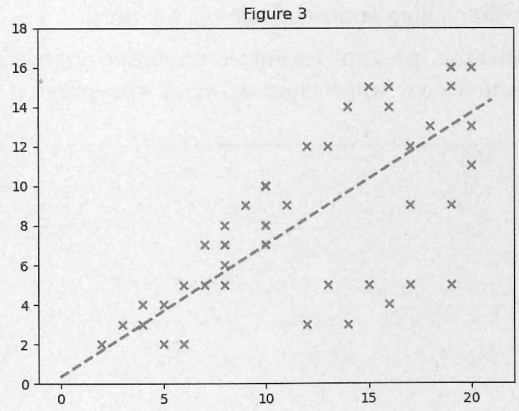
9. Dans toute cette question, on suppose $n = 20$. On simule 50 réalisations du couple de variables aléatoires (X, Y) à l'aide de la fonction `simul_XY` définie à la question 8b). On représente alors les valeurs obtenues sous forme d'un nuage de points, où les valeurs des réalisations de X sont représentées en abscisse et les valeurs des réalisations de Y en ordonnée. On trace également, sur la même figure, la droite de régression linéaire associée à ce nuage de points.

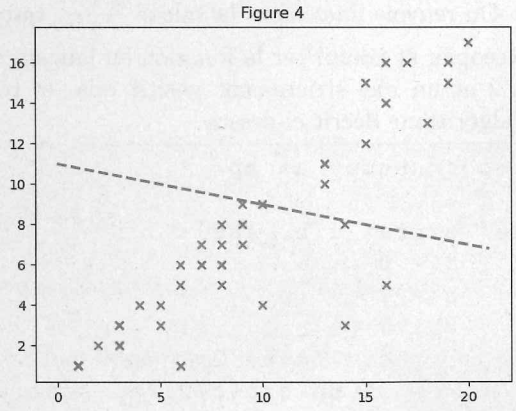
(a) Déterminer par un calcul une valeur approchée des coordonnées du point moyen du nuage de points. Quel théorème de probabilités permet de justifier cette approximation ?

(b) Parmi les figures représentées ci-dessous, en justifiant soigneusement votre réponse, indiquer celle qui correspond au nuage de points et à la droite de régression linéaire étudiés.









Exercice 2

On considère la fonction f définie sur $]0,+\infty[$ par :

$$\forall x \in]0,+\infty[, \qquad f(x) = \frac{e^{\frac{x}{2}}}{\sqrt{x}}.$$

On rappelle que $2 < e < 3$.

1. (a) Montrer que f est dérivable sur $]0,+\infty[$ et que, pour tout réel x de $]0,+\infty[$:
$$f'(x) = \frac{(x-1)}{2x} f(x).$$
(b) Dresser le tableau de variations de f et déterminer les limites suivantes $\lim\limits_{x\to 0} f(x)$ et $\lim\limits_{x\to+\infty} f(x)$.
(c) Tracer l'allure de la courbe représentative de f.
(d) Montrer que, pour tout entier n supérieur ou égal à 2, l'équation $f(x) = n$, d'inconnue x dans $]0,+\infty[$, possède exactement deux solutions u_n et v_n, avec :
$$0 < u_n < 1 < v_n.$$
2. (a) Montrer que la suite $(v_n)_{n\geqslant 2}$ est croissante.
(b) Montrer par l'absurde que la suite $(v_n)_{n\geqslant 2}$ tend vers $+\infty$ quand n tend vers $+\infty$.
3. (a) Montrer que la suite $(u_n)_{n\geqslant 2}$ est décroissante.
(b) En déduire que la suite $(u_n)_{n\geqslant 2}$ converge.
Dans les questions qui suivent, on note ℓ la limite de la suite $(u_n)_{n\geqslant 2}$.
(c) Montrer par l'absurde que $\ell = 0$.
(d) En déduire que $\lim\limits_{n\to+\infty} n^2 u_n = 1$ puis un équivalent simple de u_n lorsque n tend vers $+\infty$.
4. (a) Soient n un entier supérieur ou égal à 2 et ε un réel strictement positif.
On cherche à déterminer une valeur approchée de u_n avec une marge d'erreur inférieure ou égale à ε.
On rappelle pour cela le principe de l'algorithme de dichotomie.
- On initialise deux variables a et b en leur affectant respectivement les valeurs 0 et 1.
- Tant que $b - a > \varepsilon$, on répète les opérations suivantes.
On considère le milieu c du segment $[a,b]$. Par monotonie de f sur $]0,1]$, en distinguant les cas $f(c) \leqslant n$ et $f(c) > n$, on peut déterminer si u_n appartient à l'intervalle $[a,c]$ ou à l'intervalle $[c,b]$.
Selon le cas, on met alors à jour la valeur de a ou de b pour se restreindre au sous-intervalle approprié.
- On renvoie finalement la valeur $\dfrac{a+b}{2}$, qui constitue une valeur approchée de u_n à ε près.

Recopier et compléter la fonction en langage Python suivante, prenant en entrée un entier `n` supérieur ou égal à 2 et un réel strictement positif `eps`, et renvoyant une valeur approchée de u_n à `eps` près en appliquant l'algorithme décrit ci-dessus.

```
import numpy as np

def approx_u(n, eps):
    a = 0
    b = 1
    while __________:
        c = (a+b)/2
        if np.exp(c/2)/np.sqrt(c) < n:
            __________
        else:
            __________
    return (a+b)/2
```

(b) Écrire une fonction en langage Python, nommée `sp`, prenant en entrée un entier `N` supérieur ou égal à 2 et un réel strictement positif `eps` et renvoyant une valeur approchée de la somme $\sum\limits_{n=2}^{N} u_n$ à `eps` près.

On pourra faire appel à la fonction `approx_u` définie à la question précédente.

Exercice 3

Partie 1

On considère la matrice $A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$. On note f l'endomorphisme de $\mathbb{R}^4$ représenté par la matrice A dans la base canonique $\mathscr{C} = (e_1, e_2, e_3, e_4)$ de $\mathbb{R}^4$.

1. On considère les vecteurs suivants de $\mathbb{R}^4$:

$$u_1 = (-1, 1, 0, 1), \quad u_2 = (0, -1, 1, 0), \quad u_3 = (0, 1, 1, 0), \quad u_4 = (1, 0, 0, 1).$$

On note $\mathscr{B} = (u_1, u_2, u_3, u_4)$.

(a) Montrer que $\mathscr{B}$ est une base de $\mathbb{R}^4$.

(b) Déterminer la matrice représentative de f dans la base $\mathscr{B}$.

(c) En déduire une matrice P de $\mathscr{M}_4(\mathbb{R})$ inversible et une matrice T de $\mathscr{M}_4(\mathbb{R})$ triangulaire telles que $A = PTP^{-1}$.

2. (a) Calculer A^2 et A^3, puis vérifier que $A^3 = 4A^2 - 4A$.

(b) Montrer par récurrence que, pour tout entier naturel n non nul, il existe deux réels a_n et b_n tels que

$$A^n = a_n A^2 + b_n A.$$

vérifiant, pour tout entier naturel n non nul, $a_{n+1} = 4a_n + b_n$ et $b_{n+1} = -4a_n$.

3. (a) Montrer que, pour tout entier naturel n non nul,

$$a_{n+2} = 4a_{n+1} - 4a_n.$$

(b) Déterminer, pour tout entier naturel n non nul, une expression de a_n en fonction de n.

(c) En déduire, pour tout entier naturel n non nul, une expression de b_n en fonction de n.

4. Montrer que, pour tout entier naturel n non nul,

$$A^n = \begin{pmatrix} 2^{n-1} & 0 & 0 & 2^{n-1} \\ (n+1)2^{n-2} & 2^{n-1} & 2^{n-1} & (n-1)2^{n-2} \\ (n+1)2^{n-2} & 2^{n-1} & 2^{n-1} & (n-1)2^{n-2} \\ 2^{n-1} & 0 & 0 & 2^{n-1} \end{pmatrix}.$$

Partie 2

Soient p un entier naturel non nul et G un graphe non pondéré orienté à p sommets. On note $s_0, s_1, \ldots, s_{p-1}$ les sommets de G.

5. (a) Rappeler la définition de la matrice d'adjacence du graphe G.

(b) Soient n un entier naturel non nul, i un entier de $[\![1, p]\!]$ et j un entier de $[\![1, p]\!]$.
Rappeler sans justification l'interprétation du coefficient situé à la ligne i et à la colonne j dans la matrice M^n, où M est la matrice d'adjacence du graphe G.

6. Dans cette question uniquement, on suppose que $p = 4$ et que la matrice d'adjacence du graphe G est la matrice $A = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$ étudiée dans la partie 1.

(a) Représenter les sommets et les arêtes du graphe G sous forme d'un diagramme.

(b) Le graphe G est-il connexe ? Justifier votre réponse.

(c) Soit n un entier naturel non nul.
Déterminer le nombre de chemins de longueur n menant du s_3 au sommet s_0.

Tournez la page s.v.p.

7. Dans cette question et les suivantes, on revient au cas général décrit au début de la partie 2.
Soit s un sommet de G. On dit que le sommet t est un voisin de s quand $s \neq t$ et (s,t) est une arrête du graphe.
Comme le graphe est orienté, si t est un voisin de s, alors s n'est pas forcément un voisin de t.
On appelle liste d'adjacence du graphe G, une liste de p sous-listes telle que, pour tout entier k de $[\![0, p-1]\!]$, la sous-liste située à la position k contient tous les numéros des sommets voisins de s_k.
Par exemple, la liste d'adjacence du graphe étudié à la question 6 est :

```
L = [[0, 3], [0, 1, 2], [0, 1, 2], [0, 3]]
```

Écrire une fonction en langage Python, nommée `matrice_vers_liste`, prenant en entrée la matrice d'adjacence `A` d'un graphe G (définie sous forme de liste de listes) et renvoyant la liste d'adjacence de G.

8. On cherche à écrire une fonction en langage Python permettant d'obtenir la longueur du plus court chemin menant d'un sommet de départ s_i à chaque sommet du graphe G.
On souhaite pour cela appliquer un algorithme faisant intervenir les variables suivantes :
   - Une liste `distances` à p éléments, où l'élément situé à la position k sera égal, à la fin de l'algorithme, à la longueur du plus court chemin menant du sommet de départ s_i au sommet s_k.
   - Une liste `a_explorer` contenant tous les sommets restant à traiter.
   - Une liste `marques` contenant tous les sommets déjà traités.

   Nous donnons ci-dessous la description de l'algorithme :
   - Initialisation des trois listes décrites ci-dessus :
     - ◇ Initialement, chaque élément de la liste `distances` est égal à p, à l'exception du sommet s_i auquel on affecte la distance 0.
     - ◇ La liste `marques` ne contient initialement que le numéro du sommet de départ s_i,
     - ◇ la liste `a_explorer` ne contient initialement que le numéro du sommet de départ s_i.
   - Tant que la liste `a_explorer` n'est pas vide, on répète les opérations suivantes :
     - ◇ Nommer `s` le premier sommet de la liste `a_explorer`, et le retirer de cette liste.
     - ◇ Pour chaque voisin `v` du sommet `s` : si `v` n'est pas dans la liste `marques`, on l'ajoute à la liste `marques`, on l'ajoute à la fin de la liste `a_explorer`, et on lui affecte une distance égale à `distances[s]+1`.

   (a) On considère le graphe orienté G étudié à la question 6.
   Donner la valeur de la liste `distances` à l'issue de l'exécution de l'algorithme décrit ci-dessus, lorsqu'on l'applique au graphe G en choisissant s_1 comme sommet de départ.

   (b) Recopier et compléter la fonction suivante, prenant en entrée la liste d'adjacence `L` du graphe G et le numéro `i0` du sommet de départ s_i, et renvoyant la liste `distances` après exécution de l'algorithme décrit ci-dessus.

```
def parcours(L, i0):
    p = len(L)
    distances = __________
    distances[i0] = 0
    a_explorer = __________
    marques = __________
    while __________:
        s = _________
        _____________
        for v in __________:
            if v not in marques:
                marques.append(v)
                __________
                __________
    return distances
```

   (c) Modifier la fonction précédente pour qu'elle renvoie la liste de tous les sommets s pour lesquels il existe un chemin menant du sommet de départ s_i au sommet s.

Annexe - Fonctions Python utiles

Manipulation de listes

On suppose que `L` désigne une liste à `n` éléments.

- L'opérateur de concaténation `+`, appliqué entre deux listes, renvoie la liste obtenue en plaçant les éléments de la seconde liste à la suite de ceux de la première liste.
  Par exemple, `[1, 2, 5] + [4, 3]` renvoie la liste `[1, 2, 5, 4, 3]`.
- L'opérateur `*`, appliqué entre une liste `L` et un entier `n`, renvoie la liste obtenue en concaténant `n` fois la liste `L` avec elle-même.
  Par exemple, `[1, 4, 2]*3` renvoie la liste `[1, 4, 2, 1, 4, 2, 1, 4, 2]`.
- La fonction `len` prend en argument d'entrée une liste et renvoie le nombre d'éléments dans cette liste.
- La commande `L.append(x)` permet d'inclure l'élément `x` à la fin de la liste `L`.
- Pour tout entier `i` entre 0 et $n-1$, `L[i]` désigne l'élément d'indice `i` de la liste `L` (les indices commencent à `0`).
- Pour tout entier `i` entre 0 et $n-1$, la commande `del L[i]` retire de la liste `L` l'élément situé à la position `i`.
  Par exemple, à l'issue des instructions

```
L = [5, 4, 8, 1]
del L[1]
```

  la liste `L` vaut `[5, 8, 1]`.

La librairie numpy

- Exemple d'importation : `import numpy as np`.
- Les opérations `+`, `-`, `*`, `/`, `**`, lorsqu'elles sont possibles, peuvent être réalisées entre deux tableaux Numpy de dimensions compatibles et agissent alors **coefficient par coefficient**.
- Les fonctions `np.sqrt` (racine carrée), `np.abs` (valeur absolue), `np.log` (logarithme népérien) et `np.exp` (fonction exponentielle) s'appliquent à une quantité numérique ou à un tableau Numpy de nombres. Dans ce dernier cas, les fonctions sont appliquées à chaque élément du tableau donné en argument d'entrée.

La bibliothèque numpy.random

- Exemple d'importation : `import numpy.random as rd`.
- La fonction `rd.random`, appelée sans argument d'entrée, renvoie une réalisation aléatoire de la loi uniforme sur l'intervalle $[0,1[$. Il est également possible de spécifier les dimensions d'un tableau Numpy en argument d'entrée pour obtenir un tableau dont les coefficients sont des réalisations indépendantes de la loi uniforme sur $[0,1[$.
- La fonction `rd.randint` prend en entrée deux entiers n et p (avec $p > n$) et renvoie une réalisation aléatoire de la loi uniforme discrète sur $[\![n, p-1]\!]$.