Exercice 1 (S1 2025)**.**

1. On considère des listes non vides ne contenant que 1 ou 2 valeurs différentes.
   Par exemple `['Marwa', 'Ambre', 'Marwa', 'Ambre', 'Ambre']`.
   Écrire, en langage Python, une fonction nommée `election1(L)` qui prend en entrée une liste L de cette forme et qui renvoie l'élément qui est majoritaire.
   La fonction doit renvoyer None en cas d'égalité.
   Sur la liste donnée en exemple, la fonction doit renvoyer 'Ambre'.

2. On considère maintenant des listes non-vides, mais pouvant contenir plus de 2 valeurs différentes. Par exemple `['Olivia', 'Gita', 'Mathys', 'Olivia', 'Mathys']`.
   La liste représente une urne, et on veut savoir qui a obtenu le plus de voix.
   Écrire, en langage Python, une fonction nommée `election2(L)` qui prend une liste ayant cette forme et qui renvoie, la liste des personnes ayant obtenu le maximum de voix (la liste contient plusieurs personnes en cas d'égalité).
   Sur la liste donnée en exemple, la fonction doit renvoyer `['Olivia', 'Mathys']` ou `['Mathys', 'Olivia']` car Olivia et Mathys sont ex aequo avec deux voix chacun.

Exercice 2 (S2 2025)**.**
Soit $n \in \mathbb{N}^*$. On parle de diviseur strict de n pour désigner tout diviseur k de n tel que $1 \leqslant k < n$. On note s_n la somme des diviseurs stricts de n. On dit que l'entier n est parfait si $s_n = n$.
En Python, la commande n%k renvoie le reste de la division de n par k (ce reste est nul si et seulement si k divise n).

1. Écrire une fonction, en langage Python, nommée `s` qui prend en argument un entier n non nul et qui renvoie la somme des diviseurs stricts de n.
2. Écrire une fonction, en langage Python, nommée `prop_parf` qui prend en entrée un entier N et qui renvoie la proportion des nombres n dans l'intervalle $[\![1, N]\!]$ qui sont parfaits.

Exercice 3 (S3 2025)**.**

1. Écrire une fonction, en langage Python, nommée `alea` qui prend en argument un entier positif k et renvoie 1 avec la probabilité $\frac{k+1}{k+2}$ et 0 avec la probabilité $\frac{1}{k+2}$.
2. Un mobile se déplace sur les points à coordonnées entières d'un axe selon les règles suivantes :
   - À l'instant $n = 0$, le mobile est au point d'abscisse 0 .
   - Si à l'instant $n \in \mathbb{N}$, le mobile est au point d'abscisse k, alors à l'instant $n+1$, il est au point d'abscisse $k+1$ avec probabilité $\frac{k+1}{k+2}$ et au point d'abscisse 0 avec probabilité $\frac{1}{k+2}$.

   On note X_n l'abscisse du mobile à l'instant n.

   (a) Écrire une fonction, en langage Python, nommée `simulX` qui prend en entrée un entier naturel n non nul et qui simule n déplacements du mobile et renvoie la valeur de X_n.

   (b) Écrire une fonction, en langage Python, nommée `attend` qui ne prend pas d'argument en entrée et qui simule les déplacements du mobile jusqu'au premier retour au point d'abscisse 0 et renvoie le plus petit n strictement positif pour lequel $X_n = 0$.

Exercice 4 (S4 2025)**.**
On appelle vecteurs creux des vecteurs de très grande dimension, dont la grande majorité des coordonnées sont nulles. Pour coder un vecteur creux, on utilise une liste de deux listes, celle de ses coordonnées non nulles et celle des indices correspondants triée par ordre croissant, suivies de la taille du vecteur.
Ainsi, le vecteur de $\mathbb{R}^9$; $[1,0,3,0,0,0,0,-1,0]$ est représenté par `[[1,3,-1], [0, 2 , 7], 9]`. Les vecteurs usuels sont, eux, codés par des vecteurs numpy.

1. Écrire une fonction `coder` renvoyant la représentation creuse d'un vecteur. Écrire une fonction `decoder` faisant le travail inverse. Par exemple :
   - `coder ([1, 0, 3, 0, 0, 0, 0, -1, 0])` doit renvoyer `[[1, 3, -1], [0, 2, 7], 9]`
   - `decoder ([[1, 3, -1], [0, 2, 7], 9])` doit renvoyer `[1, 0, 3, 0, 0, 0, 0, -1, 0]`

   Pour de tels vecteurs, il serait trop coûteux, tant en temps qu'en mémoire, d'écrire toutes les coordonnées nulles et de calculer comme habituellement dans $\mathbb{R}^n$. C'est pourquoi on utilise le codage de la question précédente.
2. Écrire une fonction `prod_scal(V1, V2)` prenant en argument deux vecteurs creux appartenant au même espace $\mathbb{R}^n$, codés comme dans la première question et qui calcule leur produit scalaire. Il n'est pas autorisé de repasser par la forme usuelle en décodant V1 et V2.

Exercice 5 (S5 2025)**.**
Le candidat pourra choisir de représenter les matrices sous la forme d'une liste de listes ou sous la forme d'un array NumPy.

1. Écrire une fonction `somme_ligne(M, i)` qui renvoie la somme des éléments de la ligne i de la matrice carrée M et une fonction `somme_colonne(M, j)` qui renvoie la somme des éléments de la colonne j de la matrice carrée M.

On dit qu'une matrice M est une matrice magique si la somme de chaque ligne et la somme de chaque colonne de M sont toutes égales.

Exemple : $\begin{pmatrix} 11 & 15 & 19 \\ 21 & 7 & 17 \\ 13 & 23 & 9 \end{pmatrix}$ est une matrice magique car la somme de chaque ligne et de chaque colonne fait 45

et $\begin{pmatrix} 7 & 1 & 6 \\ 1 & 15 & 9 \end{pmatrix}$ n'est pas une matrice magique car la somme de la première colonne fait 8 et celle de la deuxième colonne fait 16 .

2. . En déduire une fonction `est_magique(M)` qui renvoie le booléen `True` si la matrice carrée M est magique et `False` sinon.

Exercice 6 (S1 2024)**.**

1. Écrire une fonction d'argument une liste L qui teste si la liste L vérifie l'hypothèse $\mathscr{H}$ suivante :
   $\mathscr{H}$: les éléments de la liste L sont des entiers qui sont compris au sens large entre 0 et $n-1$ avec n la longueur de la liste L.
2. Écrire une fonction d'argument une liste L vérifiant l'hypothèse $\mathscr{H}$ qui teste si la liste L vérifie l'hypothèse $\mathscr{H}'$ suivante :
   $\mathscr{H}'$: la liste L contient exactement une fois chaque valeur entre 0 et $n-1$ où n la longueur de la liste L .

Exercice 7 (S2 2024)**.**
Soit a et b deux entiers naturels avec $a < b$.

1. Écrire une fonction `verif (L, a, b )` qui a pour paramètres une liste L d'entiers naturels et 2 entiers a et b et qui renvoie `True` si tous les éléments de L sont dans l'intervalle $[\![a, b]\!]$ et `False` sinon.

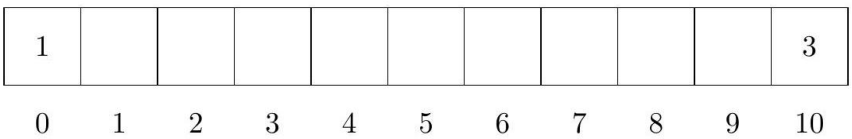
Soit L une liste dont chaque élément est dans l'intervalle d'entiers $[\![a, b]\!]$.

2. Écrire une fonction `denombre (L, a, b )` qui détermine le nombre d'apparitions de chacune des valeurs possibles (entre a et b) de la liste L et qui renvoie le résultat dans une liste dont le premier élément représentera le nombre de a dans L, le deuxième le nombre de $a+1$ dans L etc.
   Exemple : pour $a = 0$ et $b = 4$, `denombre ([1,3,0,4,1,3,1], 0,4)` renverra $[1,3,0,2,1]$.

Exercice 8 (S3 2024)**.**

1. Écrire une fonction python, d'arguments d'entrée i et N, qui simule une marche aléatoire sur $\mathbb{Z}$: le marcheur démarre sur l'entier i et à chaque pas, il avance de 1 ou recule de 1 avec probabilité 1/2. La marche s'arrête après le N-ième pas et la fonction renvoie l'entier sur lequel le marcheur s'est arrêté.

Un plateau de jeu est constitué de $n+1$ cases numérotées de 0 à n avec $n > 1$, les cases 0 et n contenant des valeurs a et b, comme ci-dessous où $n = 10$, $a = 1$ et $b = 3$:



Pour $i \in \{0, 1, \ldots, n\}$, on considère le jeu J_i suivant : on place un pion dans la case de numéro i et tant que l'on n'est pas dans la case 0 ou dans la case n, on lance une pièce équilibrée : si elle tombe sur face, on se déplace vers la gauche ; sinon, on se déplace vers la droite. On admet que la partie se termine presque sûrement : le gain du joueur est la valeur (a ou b) contenue dans la case finale.

2. Écrire une fonction qui prend en paramètres les valeurs n, a, b et i, qui simule ce jeu et qui renvoie le gain.

Exercice 9 (S4 2024)**.**
On souhaite exploiter le suivi d'une randonnée en effectuant des mesures lors de différents points de passage : latitude, longitude, altitude et temps (date).
Ces données sont contenues dans une liste `coords`, où `coords[i]` désigne la liste des 4 mesures effectuées au point de passage numéro i. Ainsi `coords[i][2]` renvoie par exemple l'altitude relevée au point de passage numéro i.

1. Écrire une fonction `temps(coords)` qui renvoie la liste des temps relevés lors de la randonnée `coords`.
2. Sans utiliser la fonction max, écrire une fonction `plus_haut(coords)` qui renvoie la liste `[lat, long]` du point le plus haut lors de la randonnée `coords`.

Exercice 10 (S5 2024)**.**

1. Écrire une fonction Python qui simule une série de N lancers d'une pièce équilibrée, et qui renvoie la liste des résultats de ces lancers ("Pile" est codé par 1, et "Face" par 0).
2. Écrire une fonction Python qui simule une série de lancers d'une pièce équilibrée jusqu'à l'obtention de la configuration "Pile, Pile, Face", et renvoie le nombre de lancers nécessaires à l'apparition de cette configuration. A l'aide de cette fonction, évaluer le temps moyen d'attente de cette configuration.

Exercice 11 (S1 2023)**.** 1. Écrire une fonction `gene(n)` qui prend en entrée un entier naturel non nul n et retourne une chaîne de n caractères formée aléatoirement, de façon équiprobable, des caractères *' A ', ' C ', ' G ',* 'T'.
2. Écrire une fonction `nbAC(g)` qui prend en entrée une chaîne de caractères formée des caractères *' A ', ' C ', ' G ', ' T '* et qui retourne le nombre de fois où la séquence *' AC '* est présente. Par exemple, `nbAC('GAGCACCCTACTTGGCGCGA')` retournera 2.

Exercice 12 (S2 2023)**.**
On s'intéresse dans cet exercice à des listes d'entiers. L'utilisation de `max` ou de `count` est interdite.

1. Écrire une fonction prenant en argument une liste d'entiers L et un entier $k \in \mathbb{N}$. Cette fonction renvoie True si tous les entiers de cette liste sont compris entre 0 et k et False sinon.
   Par exemple, pour L = [0,2,0], la fonction renvoie True si $k = 2$ ou $k = 3$ et False si $k = 1$.
2. Écrire une fonction de mêmes arguments, où la liste est supposée ne contenir que des entiers compris entre 0 et k. Cette fonction renvoie l'élément le plus fréquent (ou l'un d'entre eux s'il y en a plusieurs).
   Par exemple, pour L = [0,4,0,1,4] et $k = 3$, la fonction renvoie 0 ou 4 .

Exercice 13 (S2 2023)**.** 1. Écrire une fonction `somme(f,a, b,N)` qui retourne $\sum_{k=0}^{N-1} f\left(a + k\frac{b-a}{N}\right)$ où f est une fonction, a et b deux réels ($a < b$), et N un entier supérieur à 1 .
2. Écrire une fonction prenant en entrée une fonction `f`, deux réels `a, b` qui retourne une valeur approchée de $\int_a^b f(t)\,\mathrm{d}t$.
   Utiliser cette fonction pour donner une valeur approchée de $\int_0^{+\infty} \exp\left(-t^2\right)\,\mathrm{d}t$ à l'aide de l'égalité :
$$\int_0^{+\infty} f(t)\,\mathrm{d}t = \int_0^1 \frac{1}{(1-t)^2} f\left(\frac{t}{1-t}\right)\mathrm{d}t$$

Exercice 14 (S4 2023)**.**
Soit $n \in \mathbb{N}$ et $(a_0, \ldots, a_n) \in \mathbb{R}^{n+1}$. On considère le polynôme $P(X) = a_0 + a_1 X + \ldots + a_n X^n$.
Dans Python, on représente ce polynôme P par la liste de ses coefficients $[a_0, \ldots, a_n]$.

1. Écrire une fonction Python `eval` qui prend en argument un polynôme P et un réel a et qui renvoie $P(a)$.
2. Écrire une fonction Python `deriv` qui prend en argument un polynôme P et qui renvoie P'.
3. On considère l'application linéaire f définie sur $\mathbb{R}[X]$ par $f(P)(X) = 2XP(X) - P'(X)$.
   Écrire une fonction Python `f` qui prend en argument un polynôme P et qui renvoie $f(P)$.

Exercice 15 (S5 2023)**.**

1. Écrire une fonction `somme_cumul` qui prend en entrée une liste L d'entiers et qui renvoie une liste M des sommes cumulées, c'est à dire une liste M, de même longueur que L, telle que $M[i] = \sum_{k=0}^{i} L[k]$ pour tout indice i de L.
2. On propose à deux joueurs A et F la résolution d'une suite infinie de problèmes numérotés $0, 1, \ldots, n, \ldots$. Ils débutent la résolution à l'instant 0 du problème numéro 0 . Dès que l'un des joueurs termine la résolution du problème k il passe au problème $k+1$.

   Le numéro d'un problème est attribué au joueur qui le résout en premier. En cas de simultanéité de la résolution d'un problème par les deux joueurs celui-ci n'est pas attribué.

   On souhaite écrire une fonction Python `répartition(duréesA, duréesF)` qui, étant donnés les deux listes des durées de résolution des n premiers problèmes par les deux joueurs, renvoie les listes des numéros des problèmes attribués à A et F pour les n premiers problèmes.

   (a) `répartition([7,1,2,1,2],[4,2,4,3,1])` renvoie `([3,4],[0,1])`.
   Quel couple renvoie l'exécution de `répartition([3,2,5,1,1,1],[4,2,2,4,4,2])` ?

   (b) Écrire la fonction recherchée.