TP 2

Révisions : listes (tableaux)

10 septembre 2025

Créez un fichier `TP2.py` dans votre dossier d'informatique PSI. Faire **tous les exercices dans ce fichier**.

Lorsque vous testez une fonction, mettre les appels à la fonction directement **dans votre programme** puis mettre les appels en commentaire avec `#` avant de passer à l'exercice suivant (**ne pas les effacer**).

Testez vos fonctions au fur et à mesure pour les vérifier **avec des valeurs refléchies** ! Par un exemple un test sur la liste [1,2,3,...] n'est en général pas un bon test...

A la fin du TP, vous devez déposer votre fichier sur le site `https://cahier-de-prepa.fr/psi-lessouriau`

Exercice 1 : Miroir

Écrivez une fonction `miroir` qui prend en argument une liste ℓ et renvoie une nouvelle liste dont les éléments sont les éléments ℓ mais dans l'ordre inverse de celui dans lequel ils sont dans ℓ. Ne pas utiliser de méthode intrinsèque à Python (par exemple `reverse`), le but est de coder ce type d'opération par vous-mêmes.

Exercice 2 : TestTri

Écrivez une fonction `testTri` qui prend comme argument une liste d'entiers et teste si cette liste est triée en ordre croissant.

Exercice 3 : Incrémente

Écrivez une fonction `incremente` qui prend en argument un tableau bidimensionnel (c'est-à-dire une liste de listes) contenant des entiers, et qui ajoute 1 à chaque entier du tableau puis le renvoie.

Exercice 4 : Table

Écrivez une fonction `table` qui prend en argument deux entiers n et m, et qui renvoie un tableau à deux dimensions (c'est-à-dire une liste de listes) de n lignes et de m colonnes, dont la case d'indice i, j contient le produit $i \times j$ pour tous i, j.

Exercice 5 : Recherche

Écrivez une fonction `recherche` qui prend en argument un tableau bidimensionnel (c'est-à-dire une liste de listes) et une valeur et teste si cette valeur apparaît dans une des cases du tableau.

Exercice 6 : Carré magique

Un carré magique est une matrice carrée telle que la somme de chaque rangée, de chaque colonne et de chaque diagonale a la même valeur. Un carré magique de n lignes est dit normal s'il contient chaque entier compris entre 1 et n^2 exactement une fois. Par exemple, le tableau suivant est un carré magique normal :

6	7	2
1	5	9
8	3	4

Dans la suite on écrit des fonctions pour tester si un tableau bidimensionnel est un carré magique normal.

1. Écrivez une fonction `carre` qui teste si une liste de listes d'entiers donnée en paramètre est une matrice carrée (c'est-à-dire un tableau dont toutes les lignes et les colonnes ont même longueur).
2. Écrivez deux fonctions `sommeLigne` et `sommeColonne` qui prennent en argument un entier i et une matrice (pas forcément carrée) et qui renvoient : pour `sommeLigne`, la somme des éléments de la ie ligne de la matrice, et pour `sommeColonne`, la somme des éléments de la ie colonne de la matrice.

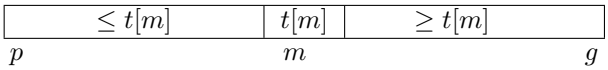
3. Écrivez deux fonctions `sommeDiagonaleMajeure` et `sommeDiagonaleMineure` qui prennent en argument une matrice carrée et renvoient respectivement la somme de la diagonale majeure, et celle de la diagonale mineure du tableau passé en paramètre.
4. Écrivez une fonction `carreMagique` qui prend en argument un tableau bidimensionnel, et teste s'il s'agit d'un carré magique (pas forcément normal).
5. Pour savoir si un carré magique est normal, il faut compter le nombre de fois qu'apparaissent ses éléments. Écrivez une fonction `histogramme` qui prend en argument un tableau bidimensionnel t et qui renvoie l'histogramme du tableau t, c'est-à-dire la liste h de taille n^2 (avec n le nombre de lignes de t) tel que pour tout $i < n^2$, $h[i]$ contient le nombre d'occurrences de la valeur $i+1$ dans le tableau t (on peut aussi choisir de construire h de taille n^2+1 afin que pour $1 \leq i \leq n^2$, $h[i]$ contient le nombre d'occurrences de la valeur i dans le tableau t). Par exemple pour un carré magique 3×3, son histogramme aura 9 cases, pour compter le nombre d'apparition des éléments compris entre 1 et 9.
6. Écrivez une fonction `carreMagiqueNormal` qui prend en argument un tableau bidimensionnel, et teste s'il s'agit d'un carré magique normal.

Exercice 7 : Dichotomie

Lorsque la liste t dans laquelle on recherche une valeur v est triée, il est plus efficace de procéder en utilisant un algorithme appelé recherche par dichotomie plutôt que la recherche linéaire utilisée aux exercices 5 et 6.

L'idée générale de la recherche par dichotomie est de comparer la valeur v recherchée à un élément $t[m]$ qui se situe au milieu de la liste. Ainsi si v est supérieur à $t[m]$, on sait qu'il est inutile de chercher v au début de la liste, et on peut se concentrer sur les indices supérieurs à m pour continuer notre recherche. De même si v est inférieur à $t[m]$, on sait qu'il est inutile de chercher v à la fin de la liste, et on peut se concentrer sur les indices inférieurs à m pour continuer notre recherche. Ainsi chaque étape permet de diviser par deux la taille de la sous-liste dans laquelle on cherche, ce qui est bien plus efficace que de tester tous les éléments un par un.

Plus précisément, la recherche par dichotomie se fait en manipulant trois indices $p \leq m \leq g$ (pour « petit », « moyen » et « grand ») : p et g délimitent la partie de la liste dans laquelle on est en train de chercher, et m désigne l'indice de l'élément du milieu auquel comparer la valeur v recherchée.



Initialement, p vaut 0, et g vaut la taille de la liste moins un. À chaque étape du calcul, m vaut $(p+g)/2$. Si $t[m]$ vaut v (la valeur recherchée), alors l'algorithme termine. Dans le cas contraire, si $t[m] < v$, alors p prend la valeur $m+1$, sinon g prend la valeur $m-1$ (car alors on sait que $t[m] > v$). Si $p > g$, alors l'algorithme termine.

Écrivez une fonction `rechercheDichotomie` qui prend en paramètre une valeur et une liste supposée triée (vous n'avez pas à le vérifier), et qui utilise une recherche par dichotomie pour trouver un indice d'occurrence de la valeur dans la liste (et renvoie `False` si un tel indice n'existe pas).