# TP3 - Listes

## 1 Listes à une dimension

### 1.1 Premières manipulations

Dans cette partie, les manipulations sont à effectuer dans la console.

**Question 1.** Créer la liste `L = [1,2,3,4,5]` et accéder au terme d'indice 4. Commenter.

**Question 2.** Redéfinir le terme d'indice 4 comme étant la somme des deux précédents.

**Question 3.** Essayer de sommer et de multiplier deux listes entre elles puis de sommer et de multiplier une liste et un entier. Conclure.

**Question 4.** Lire le dernier terme de la liste `L` sans accéder à la longueur de la liste.

**Question 5.** Stocker dans une variable `x` la valeur d'indice 4 de la liste `L` et supprimer cette valeur de la liste en une seule instruction.

**Question 6.** Exécuter le code ci-contre. Qu'observez-vous ?

```
L1 = [0 ,1 ,2 ,3]
L2 = L1
L1[1] = 4
print (L2)
```

Pour générer des listes différentes, on peut utiliser le code suivant :

```
import random as rd  #Importation du module random renommé en rd
L = [rd.randint(1,9) for i in range (10)]
print (L)
```

**Question 7.** Tester le code ci-dessus, puis le modifier afin de créer une liste aléatoire de taille 15 composée d'entiers entre 0 et 7.

### 1.2 Algorithmes fondamentaux

**Question 8.** Ecrire le code de la fonction `index` prenant en paramètre une liste `L` d'éléments d'un certain type et un élément `e` du même type, et renvoyant l'indice de la première occurrence de `e` dans `L` si il est présent et `False` sinon. Ainsi, l'appel `index([7,4,9,-3,5,-3],-3)` renvoie 3.

Evaluer la complexité de cette fonction.

**Question 9.** Ecrire le code de la fonction `somme` prenant en paramètre une liste d'entiers `L` et renvoyant la somme des éléments de `L`.

Evaluer la complexité de cette fonction.

**Question 10.** Ecrire le code de la fonction `present` prenant en paramètre une liste `L` d'éléments d'un certain type et un élément `e` du même type. Elle renvoie un booléen indiquant si `L` contient `e`.

Evaluer la complexité de cette fonction.

**Question 11.** Ecrire le code de la fonction `compter` prenant en paramètre une liste `L` d'éléments d'un certain type et un élément `e` du même type, et qui renvoie le nombre d'occurrence de `e` dans `L`. Par exemple, dans la liste `[4,7,2,7]`, le nombre d'occurrences de 1 est 0, le nombre d'occurrences de 4 est 1 et le nombre d'occurrences de 7 est 2.

Evaluer la complexité de cette fonction.

**Question 12.** Ecrire le code de la fonction `minimum` prenant en paramètre une liste d'entiers (non vide) et renvoyant son minimum. Ainsi, dans la liste `[7,4,9,-3,5,-3]`, le minimum est $-3$.

Evaluer la complexité de cette fonction.

**Question 13.**

1. Ecrire le code de la fonction `indiceMin` prenant en paramètre une liste non vide d'entiers et renvoyant l'indice de la position du minimum. Dans la liste `[7,4,9,-3,5,-3]`, le minimum $-3$ est atteint aux indices 3 et 5. La fonction renverra le dernier rencontré.
2. Ecrire le code de la fonction `listeIndMin` prenant en paramètre une liste non vide d'entiers et renvoyant la liste des indices des minimums. Dans notre exemple, on obtient la liste `[3,5]`.

   Evaluer la complexité de ces fonctions.

## 1.3 Autres exercices

**Question 14.** Ecrire le code de la fonction `supprimer` prenant en paramètre une liste `L` et un élément `e`, et renvoyant la liste `L` de laquelle on a supprimé la première occurrence de `e`.

Evaluer la complexité de cette fonction.

**Question 15.** Ecrire le code de la fonction `miroir` prenant en paramètre une liste `L` et renvoyant une nouvelle liste contenant le miroir de `L`. Ainsi, avec `[7,4,9,-3,5,-3]`, on obtient `[-3,5,-3,9,4,7]`.

Evaluer la complexité de cette fonction.

**Question 16.** Ecrire le code de la fonction `inserer` prenant en paramètre une liste `L` triée dans l'ordre croissant et un élément `e`, et renvoyant une nouvelle liste triée dans l'ordre croissant et contenant les éléments de `L` plus la valeur `e`.

Evaluer la complexité de cette fonction.

**Question 17.** Ecrire le code de la fonction `croissant` prenant en paramètre une liste et renvoyant un booléen indiquant si la liste est rangée dans l'ordre croissant.

Evaluer la complexité de cette fonction.

**Question 18.** Ecrire le code d'une fonction `getListeSuite` qui prend en argument un entier `n` et qui renvoie une liste contenant les $n$ premiers termes de la suite définie par $\begin{cases} u_0 = 3 \\ u_k = 2u_{k-1} + 1 \quad \text{pour } k \geq 1 \end{cases}$

Evaluer la complexité de cette fonction.

**Question 19.** Dans cette question, on considère une liste `L` composée de $-1$ et d'entiers positifs inférieurs à sa taille. Par exemple, une liste de 5 éléments dont les indices vont de 0 à 4 ne contiendra pas d'entier strictement supérieur à 4.

1. Ecrire le code de la fonction `rearrange` qui prend en paramètre une liste `L` comme ci-dessus et qui renvoie une nouvelle liste `newL` contenant les même éléments que `L` telle que `newL[i]=i` si l'entier $i$ est présent dans `L` et `newL[i]=-1` sinon. Par exemple, avec la liste `[-1,-1,6,1,9,3,2,-1,4,-1]`, on doit obtenir la liste `[-1,1,2,3,4,-1,6,-1,-1,9]`.
2. Ecrire une version **en place** de la fonction `rearrange`. Dans cette version, il est interdit de créer une nouvelle liste, il faut modifier la liste `L` donnée en entrée.

Evaluer la complexité de ces fonctions.

**Question 20.** Écrire le code de la fonction `lesDeuxPlusProches` prenant en paramètre une liste d'entiers et retournant une nouvelle liste contenant les deux entiers les plus proches (pas forcément consécutifs). Par exemple, avec la liste `L=[39,28,41,13,23,29,10,14,0,46]`, l'appel `lesDeuxPlusProches(L)` retourne `[28,29]`.

On pourra utiliser la fonction `abs` qui prend en entrée un réel et renvoie sa valeur absolue.

Evaluer la complexité de cette fonction.

**Question 21.** Écrire le code de la fonction `supprimeDoublons` prenant en paramètre une liste d'entiers et retournant une nouvelle liste sans éléments dupliqués. Par exemple, l'appel `supprimeDoublons([6,1,0,6,7,9,1,8,8,5])` retourne `[6,1,0,7,9,8,5]`.

Evaluer la complexité de cette fonction.

**Question 22.** Écrire le code de la fonction `majorityElement` prenant en paramètre une liste d'entiers et retournant l'élément majoritaire. S'il y en a plusieurs, elle retourne l'un d'eux. Par exemple, l'appel `majorityElement([0,3,1,5,3,1,2,1,2,2])` peut retourner 1 car c'est un des éléments le plus présent dans la liste (trois fois, tout comme la valeur 2).

Evaluer la complexité de cette fonction.

**Question 23.** Écrire le code de la fonction `maxSubArray` prenant en paramètre une liste `L` d'entiers relatifs et retournant la sous-liste, composée d'entiers consécutifs de `L`, contenant la plus grande somme. Par exemple, avec la liste `L=[-2,1,-3,4,-1,2, 1,-5,4]`, la fonction doit retourner `[4,-1,2,1]`. La somme des éléments de cette sous-liste vaut 6.

Il est à noter que la sous-liste obtenue a une taille indéterminée. Par exemple, avec la liste `[3,7,1]`, la sous-liste ayant la somme maximale est elle-même.

Evaluer la complexité de cette fonction.

# 2 Listes à deux dimensions

## 2.1 Premières manipulations

Dans cette partie, les manipulations sont à effectuer dans la console.

**Question 24.** Créer la liste `L = [[1,0,6], [6,1,1], [0,4,9], [8,7,6]]`.

**Question 25.** Si on la représente par une matrice, combien a-t-elle de lignes et de colonnes ?

**Question 26.** Quelles sont les valeurs des éléments `L[2]`, `L[0][1]`, `L[2][0]`, `L[3][2]` et `L[1][5]`.

**Question 27.** Ajouter la liste `[2,3,7]` à la liste `L`. Décrire la nouvelle matrice obtenue.

**Question 28.** Ecrire le code d'une fonction permettant de rajouter un 1 à la fin de chacune des lignes de la matrice, c'est-à-dire à chaque sous-liste de la liste `L`.

**Question 29.** Évaluer le code ci-contre.

```
L = [[0,1], [2,3]]
print (L)
L[0]
L[1]
L[0][1]
L[1][0]
```

**Question 30.** Trouver la commande pour accéder à la valeur 3 de la liste.

**Question 31.** Modifier les valeurs pour qu'elles soient incrémentées de 1. La liste `L` vaut alors `[[1,2],[3,4]]`.

Pour générer des listes différentes, on peut utiliser le code suivant :

```
import random as rd  #Importation du module random renommé en rd
L = [ [rd.randint(1,9) for j in range (10)] for i in range (15)]
print (L)
```

**Question 32.** Tester le code ci-dessus, puis le modifier afin de créer une liste aléatoire de taille 3 composée de listes de taille 5 contenant des entiers compris entre entre 0 et 5.

## 2.2 Reprise des algorithmes fondamentaux

**Question 33.** Ecrire le code de la fonction `somme2` prenant en paramètre une liste de listes d'entiers `L` et renvoyant la somme des éléments de `L`.

Evaluer la complexité de cette fonction.

**Question 34.** Ecrire le code de la fonction `present2` prenant en paramètre une liste de listes `L` d'éléments d'un certain type et un élément `e` du même type. Elle renvoie un booléen indiquant si `L` contient `e`.

Evaluer la complexité de cette fonction.

**Question 35.** Ecrire le code de la fonction `compter2` prenant en paramètre une liste de listes `L` d'éléments d'un certain type et un élément `e` du même type, et qui renvoie le nombre d'occurrence de `e` dans `L`. Par exemple, dans la liste `[[0,2,2,9] ,[1,4,4,1], [2,4,5,6]]`, le nombre d'occurrences de 2 est 3, le nombre d'occurrences de 9 est 1 et le nombre d'occurrences de 7 est 0.

Evaluer la complexité de cette fonction.

**Question 36.** Ecrire le code de la fonction `minimum2` prenant en paramètre une liste de listes d'entiers (non vide) et renvoyant son minimum. Ainsi, dans la liste `[[-7,-1,0,-9], [4,-3,4,-6], [5,8,-8,-2]]`, le minimum est $-9$.

Evaluer la complexité de cette fonction.

## 2.3 Autres exercices

**Question 37.** Ecrire le code de la fonction `sommeMatrices` prenant en paramètres deux matrices représentées par des listes de listes de même taille, et retournant une nouvelle matrice dont chaque élément d'indice $(i, j)$ est la somme des éléments d'indice $(i, j)$ des matrices données en entrée. Par exemple, avec les matrices `[[8,3, 1], [9, 4, 2]]` et `[[4,5,1], [5,2,4]]`, la fonction retourne `[[12,8,2], [14,6,6]]`.

Evaluer la complexité de cette fonction.

**Question 38.** Écrire le code de la fonction `triangleDePascal` prenant en paramètre un entier naturel `n` et retournant une liste de listes. La $n^{ième}$ sous-liste représente la $n^{ième}$ ligne du triangle de Pascal. Par exemple, l'appel `triangleDePascal(4)` renvoie `[[1],[1,1],[1,2,1],[1,3,3,1]]`.

Evaluer la complexité de cette fonction.