L'intérêt des tableaux est de pouvoir manipuler et/ou analyser un grand nombre de données. Pour les besoins du traitement par informatique, les données sont saisies sous forme de chiffres ou de caractères qui représentent des observations (ex : nombre de chômeurs, quantité de pluie…). Une fois les données saisies, leur ensemble est traité à l'aide d'algorithmes permettant alors de répondre à une question ou sert un autre traitement par informatique.

Partie I : Manipulation de tableaux

Pour chacun des exercices suivants, écrire le code qui réalise la tâche demandée et le tester.

I.1. Écrire une fonction `Moyenne` prenant en argument une liste d'entiers relatifs et renvoyant la moyenne de toutes les valeurs.

I.2. Écrire une fonction `TrouveMax` prenant en argument une d'entiers relatifs et renvoyant l'élément maximal de la liste et son indice.
Note : On peut renvoyer deux valeurs dans le return d'une fonction en les séparant par une virgule.

I.3. Écrire une fonction `ToutPositif` prenant en argument une liste d'entiers relatifs et indiquant en affichant un message si cette liste ne contient que des éléments positifs.

I.4. Le tableau ci-contre résume les notes de DS de 3 élèves au cours de l'année :

Elève	DS 1	DS 2	DS 3	DS 4
A	9,0	10,5	12,0	18,0
B	12,0	15,0	10,5	11,5
C	13,5	7,0	16,0	17,5

Ce tableau peut être représenté par la matrice :

`NotesDS = [[9.0, 10.5, 12.0, 18.0], [12.0, 15.0, 10.5, 11.5], [13.5, 7.0, 16.0, 17.5]]`

a. Ecrire une fonction `TrouveMaxNote` prenant en argument la matrice `NotesDS` et affichant l'élément maximal de la matrice.

b. En utilisant la fonction `Moyenne` (exercice Flash), écrire une fonction `TheBest` prenant en argument la matrice `NotesDS` et qui :

- crée une liste `MoyenneNotes` contenant en éléments la moyenne de chaque élève
- en utilisant la fonction `TrouveMax`, affiche la moyenne des élèves les uns après les autres dans l'ordre de leur moyenne (de la plus grande à la plus faible).
  Aide : Une fois la meilleure moyenne trouvée, l'éliminer de la liste !
  Rappel : La fonction `TrouveMax` renvoie 2 valeurs, on peut alors l'appeler de la manière suivante pour que les 2 valeurs soient enregistrées dans 2 variables : `Max, iMax = TrouveMax(L)`

Pour aller un peu plus loin…
Lorsqu'un élève est absent, le caractère "A" est mis à la place de la note et, dans la moyenne, cette absence n'est pas comptée.
Ajouter les notes de l'élève D ci-contre à la matrice `NoteDS`, puis trouver un moyen efficace (sans trop modifier les fonctions, voir sans les modifier du tout) d'afficher d'abord la meilleure de toutes les notes puis la moyenne des élèves les uns après les autres dans l'ordre de leur moyenne.

Elève	DS 1	DS 2	DS 3	DS 4
D	12,0	16,5	"A"	15,0

Partie II : Importation et manipulation de données

Nous n'avons pour l'instant traité que de très petit nombre de données, et nous pouvions les inclure facilement dans le corps du programme. Mais lorsque l'on veut traiter une quantité d'information plus importante, cette méthode devient inadéquate. On a alors recours à des fichiers de données.

Fichiers de données CSV (=CQFR !)

Il existe plusieurs formats de fichiers permettant de stocker des données, mais nous allons nous concentrer sur deux formats en particuliers :

→ Le format texte .txt (format utilisé par un éditeur de texte simple)
→ Le format Comma Separated Values .csv (format utilisé par un tableur simple)

Dans ces deux types de fichier, les données sont enregistrées **sous forme de chaines de caractères** et séparées par un symbole donné. Dans le cas des fichiers CSV, le séparateur est un point-virgule ";".

Importation de données

L'utilisation d'un fichier ressemble beaucoup à l'utilisation d'un livre : on l'ouvre, on peut le lire ou écrire dedans, on peut le lire une page après l'autre ou aller vers une page quelconque en s'aidant du numéro de page. Une fois terminé, on le ferme.

Lors de l'importation des données, les valeurs lues sont toujours stockées **sous forme de chaines de caractères**. Si on veut des entiers ou des nombres flottant, il faut alors convertir les éléments des listes créées (par compréhension étant la méthode la plus efficace).

Les commandes utiles à l'importation de données à partir d'un fichier sont les suivantes (les noms de variables peuvent être changés) :

• fichier = open("source","r")	Ouvrir le fichier nommé source (en incluant le chemin vers le fichier si nécessaire) en mode lecture ("r" pour *read*) et le stocker dans la variable fichier
• ligne = fichier.readline() ou for ligne in fichier :	Lire une ligne de fichier et la stocker dans la variable ligne. La commande readline (sans s à la fin) lit la ligne courante : la première fois qu'on l'appelle, elle lit la ligne 1 ; la seconde fois, la ligne 2 ; et ainsi de suite. On peut aussi utiliser la lecture par boucle for.
• ligne.rstrip()	Retire le caractère "\n" indiquant un saut de ligne à la fin de ligne
• valeur = ligne.split("symbole")	Place les éléments de ligne, qui sont séparé par le symbole symbole, dans une liste valeur
• fichier.seek(0)	Permet de revenir au début de fichier
• fichier.close()	Ferme le fichier après utilisation

Le fichier ListesPourExos.csv contient les listes test en format CSV pour les exercices (vous pouvez y jeter un coup d'œil en l'ouvrant avec un tableur ou le bloc-note). Le code pour lire la base de données et la séparer en différentes listes que nous allons utiliser dans les exercices est le suivant :

```
1  fichier = open("ListesPourExos.csv","r")
2
3  donnees = []
4  for ligne in fichier:
5      ligne = ligne.rstrip()
6      lecture = ligne.split(";")
7      donnees = donnees+[lecture]
8  fichier.close()
9
10 L1 = donnees[0]
11 ListePairImpair = [int(x) for x in L1[1:]]
12 L2 = donnees[1]
13 ListeDoublons1 = [int(x) for x in L2[1:]]
14 L3 = donnees[2]
15 ListeDoublons2 = [int(x) for x in L3[1:]]
16 L4 = donnees[3]
17 Checkpoint1 = [float(x) for x in L4[1:]]
18 L5 = donnees[4]
19 Checkpoint2 = [float(x) for x in L5[1:]]
20 L6 = donnees[5]
21 Checkpoint3 = [float(x) for x in L6[1:]]
22 L7 = donnees[6]
23 ListeOccurence = [int(x) for x in L7[1:]]
```

Vous pouvez trouver ce code dans le fichier Lecture_Donnees_Exos.py fournit.

II.1. Commenter brièvement le code ci-dessus dans votre fichier de travail pour expliquer ce qu'il fait.

Pour chacun des exercices suivants, écrire le code qui réalise la tâche demandée et le tester. Ne pas hésiter à écrire l'algorithme avant de coder, et penser aux commentaires !!!

II.2. Ecrire une fonction qui prend en argument une liste d'entier et renvoie le premier élément pair.
Tester (avec la méthode assert) qu'avec la liste *ListePairImpair* la fonction renvoie 12.

II.3. Ecrire une fonction qui prend en argument une liste d'entier et renvoie la valeur et l'indice du dernier élément impair.
Tester qu'avec la liste *ListePairImpair* la fonction renvoie (47, 109)

II.4. Ecrire une fonction qui prend en argument une liste d'entier et renvoie False si celle-ci contient des doublons (le même élément présent deux fois ou plus) et True sinon.
Tester qu'avec la liste *ListeDoublons1* la fonction renvoie False et qu'avec la liste *ListeDoublons2* elle renvoie True.

II.5. Lors d'une course à pied, le temps de course est mesuré à différents endroits : à 5km, on enregistre la durée en minutes entre le départ et les 5km ; à 10km, la durée entre le les 5km et les 10km. Les durées sont enregistrées dans les listes *Checkpoint1* et *Checkpoint2*, où l'indice de la liste correspond au numéro de dossard du participant.
Les participants avec les plus grands numéros de dossard couraient en fait un semi-marathon (21km) et non un 10km. Il y avait donc, pour eux, un troisième point de mesure entre les 10km et les 21km. Les durées pour ces coureurs sont enregistrées dans la liste *Checkpoint3*.

On cherche à savoir combien de temps l'évènement a duré (la durée maximale) en heure.

a. Sous quelle condition peut-on additionner les données de deux listes ?

b. Ecrire une fonction qui prend en argument deux listes, vérifie que la condition précédente est validée puis renvoi une nouvelle liste dans laquelle les éléments de chaque liste sont additionnés rang par rang.
Aide : ne pas hésiter à tester les fonctions avec de petites listes de votre création

c. Ecrire une fonction qui prend en arguments deux liste et rallonge la liste la plus courte en ajoutant des zéro en tête de liste pour que les deux listes aient la même longueur.

d. Ecrire une fonction qui prend en argument trois listes de tailles quelconques, additionne leurs éléments rang par rang dans une nouvelle liste puis renvoie la valeur maximale de la nouvelle liste.

Vérification : Avec les listes *Checkpoint1*, *Checkpoint2* et *Checkpoint3*, on voit que l'évènement a duré environ 2,8h.

Pour aller un peu plus loin…

- Ecrire une fonction qui prend en argument une liste d'entier, un élément *n* et une occurrence *p*, et renvoie l'indice de la p-ième occurrence de l'élément dans la liste.

  Exemple : si l'utilisateur choisit p = 2 et l'élément 4 pour la liste L = [2,1,3,5,2,1,0,4,5,6,8,2,1,2,3,4,5], alors on obtient 15, qui est l'indice de la deuxième apparition de 4 dans la liste.

Tester qu'avec la liste *ListeOccurence*, *n*=2 et *p*=6, la fonction renvoie 54.

- Ecrire une fonction permettant de créer une liste de 50 entiers, choisis aléatoirement entre 1 et 100 avec randint, ne contenant aucun doublon.

Savoir faire

- **Calculer la valeur moyenne des éléments d'un tableau**
- **Rechercher un extrémum dans un tableau**
- **Rechercher un élément vérifiant certains critères dans un tableau**
- **Manipuler un fichier CSV**