

# Info224 - TD2

## REMARQUES

Pour ajouter un élément  $e$  dans un tableau  $t$ , on peut :

- utiliser  $t = t + [e]$ ,
- utiliser  $t.append(e)$ .

La seconde méthode est toujours plus efficace...

Dans les exercices suivants, utilisez toujours une boucle `while` lorsque vous voulez stopper une boucle avant la fin d'un tableau : utiliser un `return` au milieu de la boucle n'est pas autorisé.

## TABLEAUX

**Exercice 1.** Écrivez une fonction qui calcule le nombre maximal apparaissant dans une case d'indice *impair* dans un tableau. Par exemple :

```
>>> max_impair([0,1,2,3,4,5,10,21,2,111])
111
```

**Exercice 2.** Écrivez une fonction `max_paquet` qui calcule :

- le maximum des valeurs aux indices  $0, 1, \dots, n-1$ ,
- le maximum des valeurs aux indices  $n, n+1, \dots, 2n-1$ ,
- le maximum des valeurs aux indices  $2n, 2n+1, \dots, 3n-1$ ,
- ...

On aura par exemple :

```
>>> max_paquet([1,2,3,2, 5,4,3,0, 10,11,10,9, 3,13,23], 4)
[3, 5, 11, 23]
```

**Exercice 3.** Écrivez une fonction qui calcule la somme des éléments d'un tableau en utilisant les deux types de boucles `for` :

- `for i in range(0, len(t)):`
- `for e in t: ...`

Faites la même chose pour calculer le maximum d'un tableau, puis pour calculer le numéro de la case contenant le maximum.

## TABLEAUX À DEUX DIMENSIONS

**Exercice 4.** On considère le tableau à 2 dimensions suivant :

```
>>> T = [ [1, 2, ],
           [0, 2, -1, ],
           [1, 0, 3, 2]]
```

Quels sont les résultats des expressions suivantes ?

```
T[1][1]          T[1][2]          T[2]
T[T[2][1]][1]   T[3][3]          T[2][T[0][0]]
```

**Exercice 5.** Une *matrice* est simplement un tableau à deux dimensions, où chaque tableau interne fait la même taille. Par exemple, le tableau T de l'exercice précédent n'est pas une matrice, mais le tableau suivant en est une :

```
>>> M = [ [0, 1, 1, 0],
           [0, 0, 1, 1],
           [1, 2, 3, 4],
           [0, 0, 0, 0],
           [1, 0, 1, 1] ]
```

Écrivez une fonction qui renvoie le nombre de lignes d'une telle matrice.

Écrivez une fonction qui renvoie le nombre de colonnes d'une telle matrice.

**Exercice 6.** Écrivez une fonction ligne qui prend une matrice et un nombre en argument, et qui renvoie :

- la ligne correspondante dans la matrice,
- ou bien le tableau vide si la ligne n'existe pas dans la matrice.

Écrivez une fonction colonne qui prend une matrice et un nombre en argument, et qui renvoie :

- la colonne correspondante dans la matrice,
- ou bien le tableau vide si la ligne n'existe pas dans la matrice.

Écrivez une fonction diagonale qui prend une matrice et qui renvoie la diagonale de la matrice. Par exemple, pour la matrice donnée plus haut, on obtiendra :

```
>>> diagonale(M)
[0, 0, 3, 0]
```

Écrivez une fonction anti\_diagonale qui prend une matrice et qui renvoie l'anti diagonale de la matrice (celle qui part du coin en haut à droite). Par exemple, pour la matrice donnée plus haut, on obtiendra :

```
>>> anti_diagonale(M)
[0, 1, 2, 0]
```

**Exercice 7.** Écrivez une fonction `est_diagonale` qui teste si une matrice est diagonale, c'ad que la matrice est carrée et que toutes ses valeurs sont 0, sauf éventuellement sur la diagonale.

Écrivez une fonction `est_triangulaire` qui teste si une matrice est triangulaire, c'ad que la matrice est carrée et que toutes ses valeurs *sous* la diagonale sont nulles.

**Exercice 8.** Écrivez une fonction `est_triee` qui vérifie si toutes les lignes d'un tableau à 2 dimensions sont triées. Votre fonction devra utiliser des boucles `while` pour s'arrêter le plus tot possible.

**Exercice 9.** Écrivez une fonction `contient_zero` qui vérifie si toutes les lignes d'un tableau à 2 dimensions contiennent au moins une case à 0. Votre fonction devra utiliser des boucles `while` pour s'arrêter le plus tot possible.