TD 10 : Prog.

1 Grilles et listes de listes

Soit le tableau ci-dessous :

1	2	3
4	5	6

1. Donner des listes de listes Python qui représentent ce tableau :
   - `g_ligne` : en listant les lignes du tableau
   - `g_col` : en listant les colonnes du tableau
2. Quelle est la longueur de `g_ligne` et `g_col` ? D'un élément de `g_ligne` et `g_col` ?
3. Donner une suite d'instructions qui ajoute une colonne contenant les entiers 7 et 8 à la droite du tableau :
   - dans le cas d'une liste de lignes
   - dans le cas d'une liste de colonnes

1. g_ligne = [[1,2,3],[4,5,6]]

   g_col = [[1,4],[2,5],[3,6]].

2. len(g_ligne) -> 2. len(g_ligne[0]) -> 3
   len(g_col) -> 3. len(g_col[0]) -> 3.

3. g_ligne[0].append(7), g_ligne[1].append(8)

   g_col.append([7,8]).

On considère le code suivant :

```
# crée une grille de 0 avec 1 au centre
ligne = [0, 0, 0]
g = [None] * 3
for i in range(3):
    g[i] = ligne
g[1][1] = 1
```

5. À votre avis, quelle était l'intention de la personne qui a écrit ce code ? Quelle est la valeur référencée par g en fin d'exécution ? Que changer pour revenir à l'intention première ?

g -> [[0,1,0],[0,1,0],[0,1,0]]

il veut juste changer le 2e élément de g.

il faudrait faire une copie de la liste ligne: g[i] = ligne.copy().

On considère la fonction suivante :

```
def positionne(grille:list[list[int]], i:int, j:int, elem:int) -> list[list[int]]:
    g = grille.copy()
    g[i][j] = elem
    return g
```

et l'appel suivant :

```
>>> g1 = [[1, 2], [3, 4]]
>>> g2 = positionne(g1, 1, 1, 5)
```

6. Donner la valeur de g1 et g2 à la fin de l'éxécution.

g1 → [[1,2],[3,4]] ; g2 → [[1,2],[3,5]]

3 Grilles et listes de listes, bis.

On s'intéresse dans un premier temps à des grilles carrées contenant des caractères `'*'` et `'+'` , représentées par la liste de leurs **lignes**.

8. Écrire une fonction `init_grille` qui prend en paramètre un entier positif strictement `n` et qui renvoie une liste de liste de caractères représentant une grille carrée de taille `n` et contenant des `'*'`.

```
def init_grille(n:int) -> list[list[str]]:
    return [['*' for _ in range(n)] for _ in range(n)]
```

9. Écrire une fonction `afficher_grille` qui prend en paramètre une liste de liste de caractères représentant une grille carrée et l'affiche en intercalant un caractère espace entre 2 éléments consécutifs d'une même ligne.

```
def afficher_grille(l:list[list[str]]) -> None:
    for i in range(len(l)):
        for j in range(len(l[0])):
            print(l[i][j], end=" ")
        print()
```

10. Écrire une fonction `grille_triangle_inf` qui prend en paramètre un entier strictement positif `n` et qui renvoie une liste de liste de caractères représentant une grille carrée de taille `n` et contenant des `'*'` sauf sur et sous la première diagonale () qui contient des `'+'`. Voyez-vous comment modifier le code pour générer la liste des paires des indices des cases contenant un `'+'` ?

```
def grille_triangle_inf(n:int)->list[list[str]]:
    g = init_grille(n)
    for i in range(n):
        for j in range(i+1):
            g[i][j] = '+'
    return g
```

11. De même écrire une fonction `grille_triangle_sup` qui prend en paramètre un entier strictement positif `n` et qui renvoie une liste de liste de caractères représentant une grille carrée de taille `n` et contenant des `'*'` sauf sur et au dessus de la première diagonale (sens \) qui contient des `'+'`.

```
def grille_triangle_inf(n:int)->list[list[str]]:
    g=init_grille(n)
    for i in range(n):
        for j in range(i,n)
            g[i][j]='+'
    return g
```

12. Écrire une fonction `complemente_case` qui prend en paramètre une liste de liste de caractères représentant une grille carrée et des indices `i` et `j` entiers dans cette grille, et renvoie une nouvelle grille équivalente au paramètre sauf la case d'indices (i, j) remplacée par un `'+'` si elle contenait un `'*'`, ou par un `'*'` si elle contenait un `'+'`.

```
def complemente_case(l:list[list[str]],i:int,j:int)->list[list[str]]:
    g=deepcopy(l)
    g[i][j]='+' if g[i][j]=='*' else '*'.
    return g.
```

13. Écrire une fonction `grille_croix` qui prend en paramètre un entier `n` impair strictement positif et renvoie une liste de liste de caractères représentant une grille carrée de taille `n` contenant des `'*'` sauf une croix centrale contenant des `'+'`.

```
def grille_croix(n:int)->list[list[str]]:
    g=init_grille()
    for i in range(n):
        g[i][(n//2)+1]='+'
        g[(n//2)+1][i]='+'
    return g.
```

14. Écrire une fonction **colonne** qui prend en paramètre une liste de liste représentant une grille et un indice de colonne **ind** valide dans cette grille et renvoie la colonne d'indice **ind**.

```
def colonne(l:list[list[str]], ind:int)->list[str],
    res=[]
    for i in range(len(l)):
        res.append(l[i][ind])
    Return res.
```

15. Écrire une fonction **transposee** qui prend en paramètre une liste de liste représentant une grille non vide et renvoie une nouvelle grille qui est la transposée du paramètre.

```
def transposee(l:list[list[str]])->list[list[str]]:
    g=[]
    for i in range(len(l)):
        g.appendColonne(l,i)
    Return g.
```

```
def sue_impression(L:list[list[str]], m:list[list[str]])->list[list[str]]:
```