DM d'informatique : pour la rentrée

Jeu de la vie

Le **jeu de la vie** n'est pas un jeu, puisqu'il ne nécessite aucun joueur. Il s'agit d'un **automate** : un modèle où chaque état conduit mécaniquement à l'état suivant à partir de règles pré-établies.

Le « jeu » se déroule sur une grille à deux dimensions, dont les cases - qu'on appelle des « cellules » - peuvent prendre deux états distincts : « vivantes » ou « mortes ».

Sauf celles des bords de la grille, chaque cellule a huit voisins (on peut faire un dessin pour s'en convaincre) :

	$j-1$	j	$j+1$
$i-1$			
i		x	
$i+1$			

À chaque étape, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines à l'étape précédente de la façon suivante :

- Une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît).
- Une cellule vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt.

Une case contenant une cellule "morte" sera représentée par la valeur 0, alors qu'une cellule "vivante" sera représentée par la valeur 1. Par exemple les configurations suivantes donnent à l'étape d'après une grille vide (mort de toutes les cellules) :

0	0	0	0	0
0	1	0	1	0
0	0	0	0	0

ou

0	0	0	0	0
0	1	1	0	0
0	0	0	0	0

ou

0	0	0	0	0
0	1	1	0	1
0	0	0	0	0

$\rightarrow$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

En revanche cette configuration est "cyclique" :

0	0	0	0	0
0	1	1	1	0
0	0	0	0	0

$\rightarrow$

0	0	1	0	0
0	0	1	0	0
0	0	1	0	0

$\rightarrow$

0	0	0	0	0
0	1	1	1	0
0	0	0	0	0

$\rightarrow \ldots$

Une **grille de taille** $n \times m$ sera représentée en Python par une liste de listes. Par exemple une grille 3×5 :

```
G = [ [0,0,0,0,0], [0,1,1,1,0], [0,0,0,0,0] ]
```
représente

0	0	0	0	0
0	1	1	1	0
0	0	0	0	0

❒ Q1. Que donne la configuration suivante après une étape ? Et après deux étapes ?

0	0	0	0	0
0	1	1	1	0
0	0	1	0	0
0	0	0	0	0

Dans toute la suite, les grilles seront de taille carré $n \times n$.

❒ Q2. Ecrire une fonction `liste_mortes(n)` qui prend en argument un entier naturel n et qui renvoie une liste de taille n ne contenant que des 0.

```
>>> liste_mortes(3)
[0, 0, 0]
```

❒ Q3. Ecrire une fonction `grille_mortes(n)` qui prend en argument un entier naturel n et qui renvoie une grille vide de taille $n \times n$ (ne contenant que des 0). On pourra utiliser `liste_mortes(n)`.

```
>>> grille_mortes(3)
[ [0, 0, 0], [0, 0, 0], [0, 0, 0] ]
```

❒ Q4. Ecrire une fonction `grille_ex(n)` qui renvoie une grille de taille n contenant les cellules vivantes placées de la façon suivante (pour $n = 5$) :

1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1

❒ Q5. Ecrire une fonction `nb_vivantes(G)` qui prend en argument une grille `G` qui renvoie le nombre de cellules vivantes de la grille `G`.

❒ Q6. Ecrire une fonction `vois_vivantes(G, i, j)` qui prend en argument une grille `G`, les coordonnées d'une case (i, j), et qui renvoie le nombre de cellules vivantes voisines de la case (i, j) dans la grille `G`. Il n'est demandé de traité que le cas des cellules qui ne sont pas sur le bord de la grille : $0 < i < len(G) - 1$ et $0 < j < len(G) - 1$.

BONUS : (à faire à la fin si le temps le permet) compléter la fonction `vois_vivantes` pour que l'algorithme fonctionne aussi pour les cellules du bord.

On suppose dans la suite que la fonction `vois_vivantes` est bien définie pour toutes les cellules (y compris celles du bord).

❒ Q7. Compléter la fonction `copie(G)` qui prend en argument une grille et qui renvoie une copie indépendante de la grille (il y a 3 lignes à compléter).

```
def copie(G):
    L = []
    n = len(G)
    for i in range(n):
        sL =
        for j in range(n):
            sL.append(          )
        L.append(          )
    return L
```

❒ Q8. Ecrire une fonction `etape(G)` qui renvoie la grille `G` après une étape du jeu de la vie.

```
>>> G = [ [0, 0, 0, 0], [0, 1, 1, 0], [0, 0, 1, 0], [0, 0, 0, 0] ]
>>> etape(G)
[[0, 0, 0, 0], [0, 1, 1, 0], [0, 1, 1, 0], [0, 0, 0, 0]]
```

❒ Q9. Ecrire une fonction `jeu_vie(G, n)` qui renvoie la grille après n étapes du jeu de la vie à partir de la grille `G`.

❒ Q10. Ecrire une fonction `jeu_vie_stop(G, n)` pour qu'il s'arrête soit au bout de n étapes soit avant si la grille devient morte (sans cellule vivante) avant les n étapes. Dans ce dernier cas, on renverra la phrase "les cellules sont toutes mortes".

❒ Q11. Déterminer un motif (à 4, puis 6 cellules vivantes) qui est stable par le jeu de la vie : à chaque étape, les cellules vivantes restent précisément au même endroit, aucune ne meurt et aucune ne naît... Celles qui sont présentes sont donc immortelles.

Quelques rappels de fonctions autorisées sur les listes.

`len(L)`	renvoie la taille (le nombre d'éléments) de la liste `L`
`L.append(b)`	ajoute l'élément `b` enfin de liste `L`