

EXERCICE 3 (4 points)

Thèmes abordés : structures de données, programmation.

Le « jeu de la vie » se déroule sur une grille à deux dimensions dont les cases, qu'on appelle des « cellules », par analogie avec les cellules vivantes, peuvent prendre deux états distincts : « vivante » (= 1) ou « morte » (= 0).

Une cellule possède au plus huit voisins, qui sont les cellules adjacentes horizontalement, verticalement et diagonalement.

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

- Règle 1 : une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît) ; sinon, elle reste à l'état « morte »
- Règle 2 : une cellule vivante possédant deux ou trois voisines vivantes reste vivante, sinon elle meurt.

Voici un exemple d'évolution du jeu de la vie appliquée à la cellule centrale :

1	1	0
0	0	0
0	0	1

 devient par la règle 1

	1	

1	0	0
0	1	1
1	0	0

 reste par la règle 2

	1	

0	0	0
1	1	0
0	0	0

 devient par la règle 2

	0	

1	1	0
0	1	1
1	1	0

 devient par la règle 2

	0	

Pour initialiser le jeu, on crée en langage Python une grille de dimension 8x8, modélisée par une liste de listes.

1. Initialisation du tableau :

a. Parmi les deux scripts proposés, indiquer celui qui vous semble le plus adapté pour initialiser un tableau de 0. Justifier votre choix

Choix 1	Choix 2
1 ligne = [0,0,0,0,0,0,0,0] 2 jeu = [] 3 for i in range(8) : 4 jeu.append(ligne)	1 jeu = [] 2 for i in range(8) : 3 ligne = [0,0,0,0,0,0,0,0] 4 jeu.append(ligne)

b. Donner l'instruction permettant de modifier la grille jeu afin d'obtenir

```
>>> jeu
[[0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 1, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0]]
```

2.

a. Ecrire en langage Python une fonction remplissage(n, jeu) qui prend en paramètres un entier n et une grille jeu, et qui ajoute aléatoirement exactement n cellules vivantes dans le tableau jeu.

b. Quelles sont les préconditions de cette fonction pour la variable n ?

On propose la fonction en langage Python nombre_de_vivants(i, j, jeu) qui prend en paramètres deux entiers i et j ainsi qu'une grille jeu et qui renvoie le nombre de voisins **vivants** de la cellule tab[i][j] :

```
1 | def nombre_de_vivants(i, j, jeu) :
2 |     nb = 0
3 |     voisins = [(i-1,j-1), (i-1,j), (i-1,j+1), (i,j+1),
4 |                (i+1,j+1), (i+1,j), (i+1,j-1), (i,j-1)]
5 |     for e in voisins :
6 |         if 0 <= ... < 8 and 0 <= ... < 8 :
7 |             nb = nb + jeu[...][...]
8 |     return nb
```

3. Recopier et compléter les pointillés pour que la fonction réponde à la demande.

4. En utilisant la fonction nombre_de_vivants(i, j, jeu) précédente, écrire en langage Python une fonction transfo_cellule(i, j, jeu) qui prend en paramètres deux entiers i et j ainsi qu'une grille jeu et renvoie le nouvel état de la cellule jeu[i][j] (0 ou 1)