

## TD : Le jeu de la vie

L'objectif de ce sujet est de vous faire découvrir « le jeu de la vie ».

La première partie vous permettra de revoir la notion de tableau à deux dimensions et leur utilisation dans Python. Ces tableaux seront utilisés dans la deuxième partie pour représenter le jeu de la vie.

### I. Tableaux à deux dimensions

#### a. Une grille en Python

En informatique, il est fréquent d'utiliser des tableaux à deux dimensions : des informations sont stockées dans une grille où chaque case est identifiée par un numéro de ligne et un numéro de colonne.

En Python, de tels tableaux sont représentés par des listes de listes.

Pour créer une telle grille, on peut utiliser la fonction suivante

```
def creerGrille(nombreLignes, nombreColonnes):  
    grid = [[]] * nombreLignes  
    for ligne in range(nombreLignes):  
        grid[ligne] = [0] * nombreColonnes  
    return grid
```

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

On peut alors écrire :

- Pour créer une grille avec  $n$  lignes et  $m$  colonnes, contenant la valeur 0 dans chacune de ses cases, on écrit `creerGrille(n,m)`
- Pour stocker une telle grille dans une variable  $g$ , on écrit donc `g = creerGrille(n,m)`
- Pour accéder au contenu de la case située sur la ligne  $i$  et la colonne  $j$  d'une grille  $g$ , on écrit `g[i][j]`
- Pour modifier le contenu de la case située sur la ligne  $i$  et la colonne  $j$  d'une grille  $g$ , en y affectant la valeur  $x$ , on écrit `g[i][j] = x`
- Pour connaître le nombre de lignes que contient une grille  $g$ , on utilise `len(g)`, et pour le nombre de colonnes, on utilise `len(g[0])`

**Question 1.** Créer une grille  $g1$  de 4 lignes et 6 colonnes ne contenant que des 0.

**Question 2.** Créer une grille  $g2$  contenant les caractères ci-dessous :

|   |   |   |   |
|---|---|---|---|
| A | B | C | D |
| E | F | G | H |

## b. Génération aléatoire

**Question 3.** Créer une fonction `creerGrilleNombresAleatoires`, telle que `creerGrilleNombresAleatoires()` renvoie une grille avec 3 lignes et 4 colonnes, et dont chaque case contient un entier tiré aléatoirement entre 0 et 9.

**Question 4.** Créer une fonction `creerGrilleAleatoire`, telle que `creerGrilleAleatoire(n,m,p)` renvoie une grille de booléens avec  $n$  lignes et  $m$  colonnes, dont chaque case a  $p\%$  de chances de contenir la valeur `True` (et `False` sinon).

## c. Parcourir un tableau à deux dimensions

**Question 5.** Imaginer une fonction `afficherGrille`, telle que `afficherGrille(g)` affiche le contenu de la grille  $g$ .

Exemple : `g1 = [ [ 2 , 4 , 8 ] , [ 7 , 5 , 9 ] ]`

`afficherGrille(g1)`

```
2      4      8
7      5      9
```

# II. Le jeu de la vie

## a. Définition

Le « jeu de la vie » a été inventé par le mathématicien britannique John Horton Conway en 1970.

Il s'agit d'un « automate cellulaire », c'est-à-dire d'une machine qui évolue mécaniquement d'un état vers un autre. Cette évolution est déterminée par un ensemble de règles, qui malgré leur simplicité permettent d'obtenir des motifs très complexes.

Le « jeu de la vie » se déroule sur une grille, dont chacune des cases est appelée une « cellule ». Chacune de ces cellules a deux états possibles : « vivante » ou « morte ».

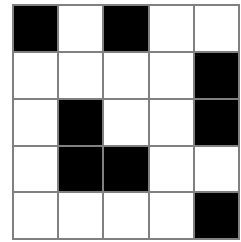
Toutes les cellules de la grille évoluent en même temps, étape par étape, en fonction de l'état des huit cellules voisines (sur les côtés et en diagonale) :

- Si une cellule est vivante,
  - si elle a moins de deux voisines vivantes, elle meurt d'isolement
  - si elle a deux ou trois voisines vivantes, elle reste en vie
  - si elle a plus de trois voisines vivantes, elle meurt de surpopulation
- Si une cellule est morte
  - si elle a exactement trois voisines vivantes, elle naît (elle devient vivante)
  - sinon, elle reste morte

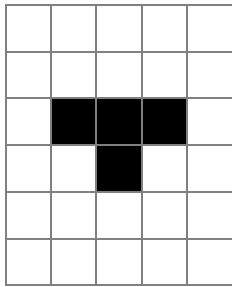
## b. Représentations

Pour représenter le jeu de la vie, on utilise généralement un code couleur :

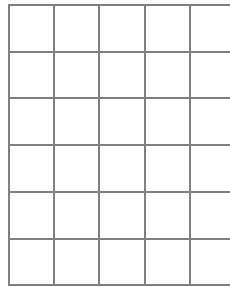
- Les cellules vivantes sont représentées en noir
- Les cellules mortes sont représentées en blanc



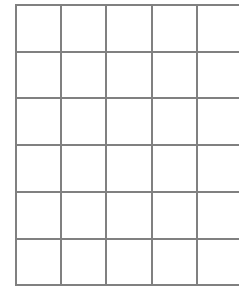
**Question 6.** Tester votre compréhension en reproduisant les grilles ci-dessous sur une feuille et en les remplissant à la main:



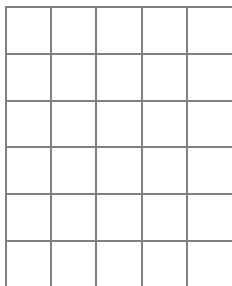
Situation de départ



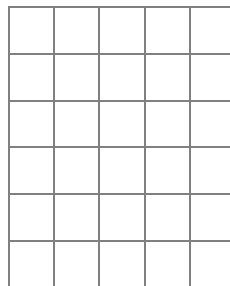
Après une étape



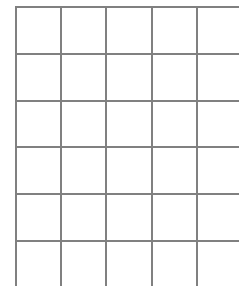
Après deux étapes



Après trois étapes

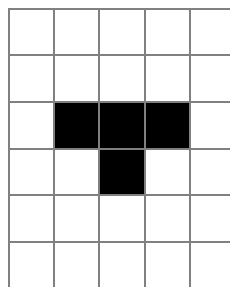


Après quatre étapes



Après cinq étapes

**Question 7.** Créer une grille de booléens correspondant au schéma ci-dessous, et utilisez la fonction `afficherGrille` pour vérifier vos résultats.



### c. Implémentation

**Question 8.** Créer une fonction `positionsCasesVoisines`, telle que `positionsCasesVoisines(g, i, j)` renvoie la liste des positions des cellules voisines de la cellule située sur la ligne `i` et la colonne `j` dans la grille `g`.

**Question 9.** Créer une fonction `nombreVoisinesVivantes`, telle que `nombreVoisinesVivantes(g, i, j)` renvoie le nombre de voisines vivantes de la cellule située sur la ligne `i` et la colonne `j` dans la grille `g`.

**Question 10.** Créer une fonction `etatSuivant`, telle que `etatSuivant(g, i, j)` renvoie l'état de la cellule située sur la ligne `i` et la colonne `j` dans la grille `g` à la prochaine étape du jeu de la vie.

**Question 11.** Créer une fonction `jeuDeLaVie`, telle que `jeuDeLaVie(g)` met à jour la grille `g` selon les règles du jeu de la vie.

**Question 12.** Créer une fonction `jeuDeLaVieAnimation`, telle que `jeuDeLaVieAnimation(g, n)` affiche les `n` (premiers) états successifs de la grille `g` dans le jeu de la vie.