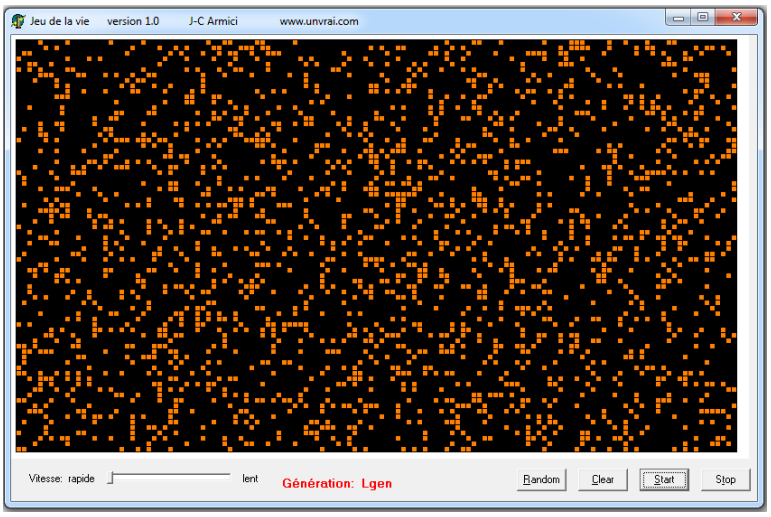
Jeu de la vie

Le jeu de la vie est un automate cellulaire imaginé par John Horton Conway en 1970 et qui est probablement le plus connu de tous les automates cellulaires. Malgré des règles très simples, le jeu de la vie est Turing-complet. Le jeu se déroule sur une grille à deux dimensions dont chacune des cases peut se voir attribuer deux valeurs : “vivante” et “morte”. A chaque tour de jeu, chacune de ces cases change d’état en fonction des huit cases qui l’entourent. Le résultat est impressionnant et rappelle l’évolution des organismes unicellulaires que l’on retrouve en biologie.

La réalisation de ce programme est bon exemple pour réviser des dictionnaires et la POO.

1 Compréhension du jeu

- Lire le lien Wikipédia :
- Exécuter le programme fourni « Vie.exe »



Pour résumer, les règles sont les suivantes :

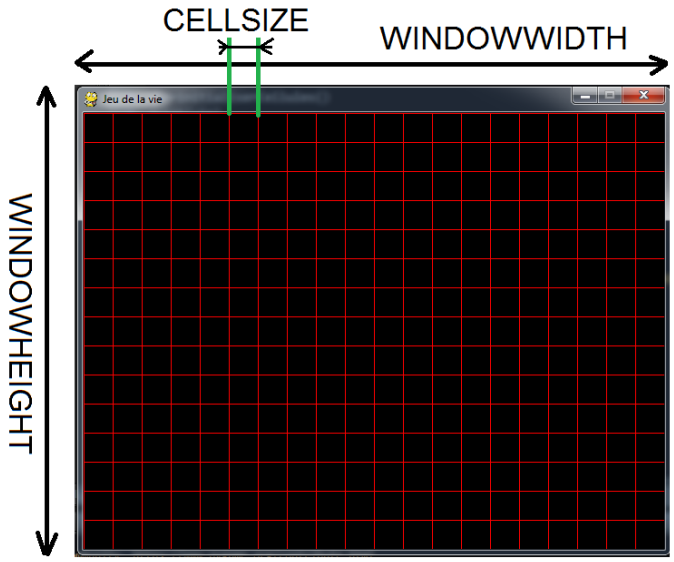
- Si une cellule a exactement trois voisines vivantes, elle est vivante à l’étape suivante.
- Si une cellule a exactement deux voisines vivantes, elle reste dans son état actuel à l’étape suivante.
- Si une cellule a strictement moins de deux ou strictement plus de trois voisines vivantes, elle est morte à l’étape suivante.

2 Programmation

Pour les fonctions suivantes, compléter le programme « vie.py »

Compléter la fonction `def tracerGrille():` qui permet de tracer la grille
Attention utiliser les constantes suivantes :

```
WINDOWWIDTH = 640
WINDOWHEIGHT = 480
CELLSIZE = 32
CELLWIDTH = WINDOWWIDTH // CELLSIZE
CELLHEIGHT = WINDOWHEIGHT // CELLSIZE
```



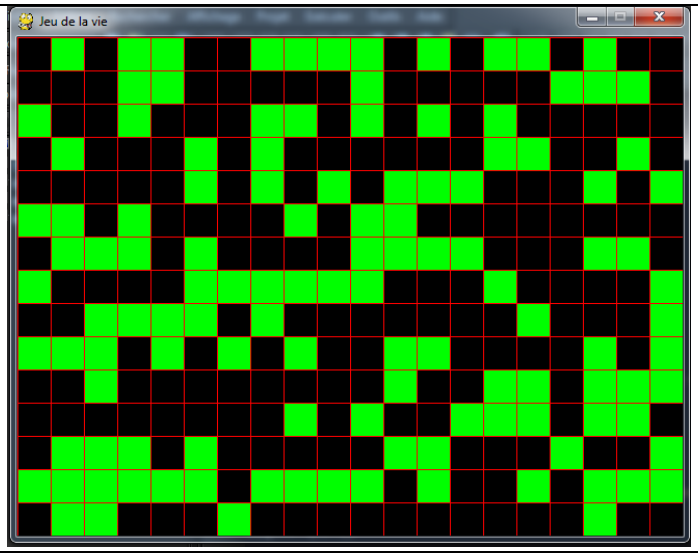
Compléter la fonction
`def initialiserCellules():`
qui permet d'initialiser et retourner le dictionnaire contenant la position x, y de la cellule. A l'initialisation, les cellules seront toutes mortes (0).

Exemple : (6, 2): 0

(6,2) est un tuple qui contient les coordonnées (x,y) de la cellule suivi de l'état de la cellule.
0 signifie cellule morte.

{(0, 0): 0, (1, 0): 0, (2, 0): 0, (3, 0): 0, (4, 0): 0, (5, 0): 0, (6, 0): 0, (7, 0): 0, (8, 0): 0, (9, 0): 0, (10, 0): 0, (11, 0): 0, (12, 0): 0, (13, 0): 0, (14, 0): 0, (15, 0): 0, (16, 0): 0, (17, 0): 0, (18, 0): 0, (19, 0): 0, (0, 1): 0, (1, 1): 0, (2, 1): 0, (3, 1): 0, (4, 1): 0, (5, 1): 0, (6, 1): 0, (7, 1): 0, (8, 1): 0, (9, 1): 0, (10, 1): 0, (11, 1): 0, (12, 1): 0, (13, 1): 0, (14, 1): 0, (15, 1): 0, (16, 1): 0, (17, 1): 0, (18, 1): 0, (19, 1): 0, (0, 2): 0, (1, 2): 0, (2, 2): 0, (3, 2): 0, (4, 2): 0, (5, 2): 0, **(6, 2): 0**, (7, 2): 0, (8, 2): 0, (9, 2): 0, (10, 2): 0, (11, 2): 0, (12, 2): 0, (13, 2): 0, (14, 2): 0, (15, 2): 0, (16, 2): 0, (17, 2): 0, (18, 2): 0, (19, 2): 0, (0, 3): 0, (1, 3): 0, (2, 3): 0, (3, 3): 0, (4, 3): 0, (5, 3): 0, (6, 3): 0, (7, 3): 0, (8, 3): 0, (9, 3): 0, (10, 3): 0, (11, 3): 0, (12, 3): 0, (13, 3): 0, ……**Trop grand pour tout afficher** ……. (19, 11): 0, (0, 12): 0, (1, 12): 0, (2, 12): 0, (3, 12): 0, (4, 12): 0, (5, 12): 0, (6, 12): 0, (7, 12): 0, (8, 12): 0, (9, 12): 0, (10, 12): 0, (11, 12): 0, (12, 12): 0, (13, 12): 0, (14, 12): 0, (15, 12): 0, (16, 12): 0, (17, 12): 0, (18, 12): 0, (19, 12): 0, (0, 13): 0, (1, 13): 0, (2, 13): 0, (3, 13): 0, (4, 13): 0, (5, 13): 0, (6, 13): 0, (7, 13): 0, (8, 13): 0, (9, 13): 0, (10, 13): 0, (11, 13): 0, (12, 13): 0, (13, 13): 0, (14, 13): 0, (15, 13): 0, (16, 13): 0, (17, 13): 0, (18, 13): 0, (19, 13): 0, (0, 14): 0, (1, 14): 0, (2, 14): 0, (3, 14): 0, (4, 14): 0, (5, 14): 0, (6, 14): 0, (7, 14): 0, (8, 14): 0, (9, 14): 0, (10, 14): 0, (11, 14): 0, (12, 14): 0, (13, 14): 0, (14, 14): 0, (15, 14): 0, (16, 14): 0, (17, 14): 0, (18, 14): 0, (19, 14): 0}

Compléter la fonction
`generationAleatoire(vie) :`
qui positionne aléatoirement les cellules vivantes(1) ou mortes(0) du dictionnaire passé en paramètre. La fonction retourne le dictionnaire modifié.
 	
Aide :
 	
Utiliser la fonction suivante :
 	
`randint(0,1)`
	{(0, 0): 1, (1, 0): 0, (2, 0): 1, (3, 0): 0, (4, 0): 1, (5, 0): 0, (6, 0): 0, (7, 0): 0, (8, 0): 0, (9, 0): 1, (10, 0): 0, (11, 0): 1, (12, 0): 1, (13, 0): 1, (14, 0): 1, (15, 0): 0, (16, 0): 0, (17, 0): 1, (18, 0): 1, (19, 0): 0, (0, 1): 1, (1, 1): 1, (2, 1): 1, (3, 1): 0, (4, 1): 0, (5, 1): 0, (6, 1): 0, (7, 1): 0, (8, 1): 1, (9, 1): 1, (10, 1): 1, (11, 1): 1, (12, 1): 0, (13, 1): 1, (14, 1): 0, (15, 1): 1, (16, 1): 0, (17, 1): 0, (18, 1): 0, (19, 1): 0, (0, 2): 1, (1, 2): 1, (2, 2): 1, (3, 2): 0, (4, 2): 1, (5, 2): 1, (6, 2): 1, (7, 2): 0, (8, 2): 1, (9, 2): 0, (10, 2): 0, (11, 2): 0, (12, 2): 0, (13, 2): 1, (14, 2): 1, (15, 2): 0, (16, 2): 1, (17, 2): 0, (18, 2): 1, (19, 2): 1, (0, 3): 0, (1, 3): 0, (2, 3): 1, (3, 3): 1, (4, 3): 1, (5, 3): 0, (6, 3): 1, (7, 3): 0, (8, 3): 0, (9, 3): 0, (10, 3): 0, (11, 3): 0, (12, 3): 1, (13, 3): 0, (14, 3): 0, (15, 3): 0, (16, 3): 0, (17, 3): 0, (18, 3): 0, (19, 3): 0, (0, 4): 0, (1, 4): 1, (2, 4): 0, (3, 4): 1, (4, 4): 1, (5, 4): 0, (6, 4): 0, (7, 4): 0, (8, 4): 1, (9, 4): 1, (10, 4): 0, (11, 4): 0, (12, 4): 0, (13, 4): 1, (14, 4): 0, (15, 4): 0, ……**Trop grand pour tout afficher** ……, (6, 13): 1, (7, 13): 1, (8, 13): 1, (9, 13): 0, (10, 13): 0, (11, 13): 0, (12, 13): 0, (13, 13): 1, (14, 13): 1, (15, 13): 0, (16, 13): 0, (17, 13): 0, (18, 13): 1, (19, 13): 0, (0, 14): 1, (1, 14): 1, (2, 14): 1, (3, 14): 1, (4, 14): 1, (5, 14): 0, (6, 14): 0, (7, 14): 1, (8, 14): 0, (9, 14): 1, (10, 14): 0, (11, 14): 0, (12, 14): 0, (13, 14): 1, (14, 14): 0, (15, 14): 1, (16, 14): 0, (17, 14): 1, (18, 14): 1, (19, 14): 0}

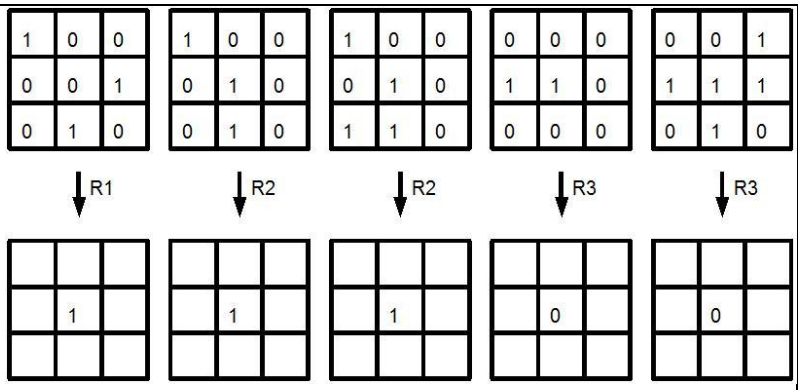
A l'aide de la fonction `def remplirGrille(vie):` Vérifier le bon affichage des carrés dans la fenêtre. Le dictionnaire `vie` est passé en paramètre.	

Compléter la fonction
`def voisins(item,vie):`

qui détermine combien de voisins de la cellule sont en vie.

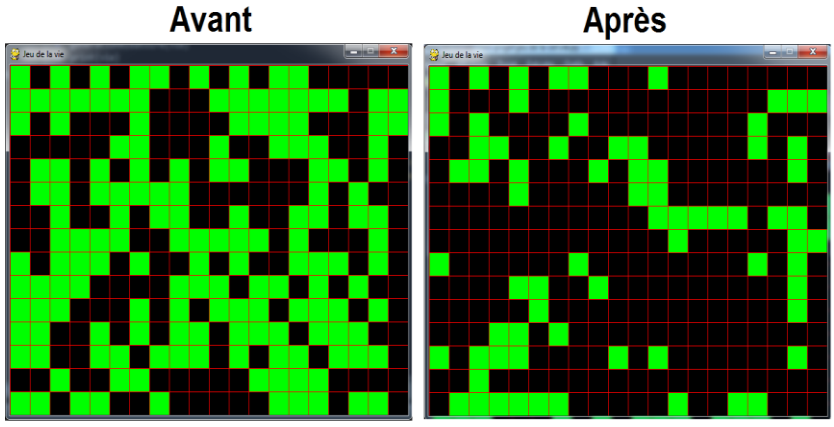
La fonction retourne le nombre de voisin en vie

Rappel item est un tuple (x,y) contenant la position de la cellule.



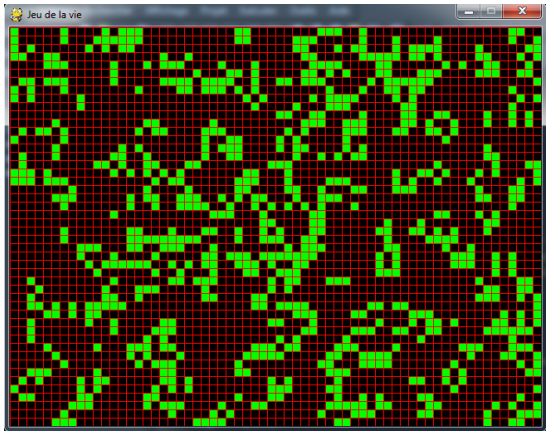
En utilisant la fonction `voisins(item, vie)`, compléter la fonction
`def prochaineEtape(vie):`
qui calcule la prochaine étape et retourne un nouveau dictionnaire

Utiliser la touche « flèche » du haut pour animer les différentes étapes



3 Augmentation du nombre de cellules

Vérifier que le bon fonctionnement du programme lorsque l'on modifie seulement la taille en pixel de la cellule (`CELLSIZE = 5`).



4 Elaborer un programme utilisant la programmation orientée objet au lieu d'utiliser un dictionnaire. (`class Cellule:`)

Aide : Commencer par élaborer un diagramme de classe.