Sup de Vinci
Bachelor 1
16 Décembre 2024

Nom:
Prénom:

Initiation à Python

/ 20

Examen machine

Sujet

Implémentation d'un Game of Life simple en Python.
Il s'agit d'une grille sur laquelle chaque cellule peut être *morte* ou *vivante*

À partir de la grille, le jeu définit des règles qui détermineront les cellules qui survivront à la prochaine génération, et celles qui mourront.

Par ce fonctionnement et ces règles simples (décrites plus bas), on peut observer un monde virtuel ayant ses propres règles, constructions communes.

Une version jouable en ligne est disponible ici

Aucune librairie n'est requise, **aucune utilisation de l'IA** ne sera toléré

Question 1

/ 2

Créer une classe `GameOfLife`

Son constructeur prends en argument `width` (la largeur de la grille)
et `height` (la hauteur de la grille)

Stocker la valeur de ces arguments dans des attributs dédiés (`self.width` et `self.height`).

Puis, créer un attribut appelé `self.grid` (nom obligatoire) contenant les données de notre grille, initialisé comme liste vide (remplie à l'étape suivante)

Question 2 / 4

1. À la fin du constructeur, remplir la grille en suivant ce principe:
- Pour chaque ligne de la grille, créer une liste vide
- Pour chaque colonne de cette ligne, ajouter une cellule morte à la liste pour la ligne
- Une fois toutes les colonnes ajoutées à la ligne, ajouter cette ligne à la grille

Note: Une cellule morte sera représentée par une valeur `False`, une cellule vivante par une valeur `True`

2. Tester l'affichage de la grille:
- Créer un nouvel objet de type `GameOfLife`
- Appeler la fonction `play` fournie en passant en argument l'objet créé
- Vérifier que tout fonctionne (utiliser CTRL+C pour quitter la boucle)

Note: Utiliser une fenêtre de terminal agrandie afin de voir l'entièreté de la grille

Question 3 / 2

Dans le constructeur, ajouter un argument `prob_alive`.

Cet argument de type `float` représente la probabilité qu'une cellule soit initialisée comme étant vivante à la création de la grille.

Pour cela modifier le code de création de la grille précédemment écrit et, au lieu de créer chaque cellule comme morte, appeler la fonction `random_boolean` en lui passant la probabilité comme argument.

Question 4 / 1.5

Créer une méthode `is_alive` prenant en argument `x` et `y`, les coordonnées d'une cellule sur la grille

Cette fonction retourne si la cellule visée est vivante ou non.

Pour tout `x` et `y` en dehors de la grille, on considère la cellule morte

Question 5 / 1.5

Créer une méthode `set_cell` prenant en argument `x` et `y`, les coordonnées d'une cellule sur la grille

Uniquement dans le cas où les `x` et `y` passés sont bien dans la grille, change l'état de la cellule en **vivante**

Question 6 / 4

1. Créer une méthode `update` ne prenant aucun argument

Utiliser le même principe que l'étape 2 pour créer une nouvelle grille dans une variable `new_grid`, cependant l'état des cellules créées seront régies par des *règles* (implémentées plus tard)

Les étapes suivantes définissent ce qui doit être fait pour décider du nouvel état de chaque cellule aux coordonnées `x`, `y`.

2. Compter le nombre de cellules *vivantes* parmis les 8 voisins proches

Les voisins proches comprennent les cellules à gauche, droite, haut, bas, et les 4 diagonales

3. Récupérer l'état actuel de la cellule, et décider du nouvel état, en suivant ces règles:

- Si elle est en *vie*, et autour d'elle *moins de 2* cellules sont en vie, elle **meurt**
- Si elle est en *vie*, et autour d'elle *2 ou 3* cellules sont en vie, elle **survit**
- Si elle est en *vie*, et autour d'elle *plus de 3* cellules sont en vie, elle **meurt**
- Si elle est *morte*, et autour d'elle *exactement 3* cellules sont en vie, elle **naît**

4. Tester et valider le fonctionnement de la simulation

Question 7 / 1

Créer une méthode `block` permettant de dessiner un block aux coordonnées `x` et `y` passés en paramètre

Voir ce lien pour savoir comment le construire

Les coordonnées correspondent au coin *haut gauche* du block

Question 8 / 1

Créer une méthode `blinker` permettant de dessiner un blinker aux coordonnées `x` et `y` passés en paramètre

Voir ce lien pour savoir comment le construire

Les coordonnées correspondent au coin *haut gauche* du blinker

Question 9 /1

Créer une méthode `glider` permettant de dessiner un glider aux coordonnées `x` et `y` passés en paramètre

Voir ce lien pour savoir comment le construire

Les coordonnées correspondent au coin *haut gauche* du glider

Question 10 /1

Créer une méthode `beacon` permettant de dessiner un beacon aux coordonnées `x` et `y` passés en paramètre

Voir ce lien pour savoir comment le construire

Les coordonnées correspondent au coin *haut gauche* du beacon

Question 11 /1

Créer une méthode `pulsar` permettant de dessiner un pulsar aux coordonnées `x` et `y` passés en paramètre

Voir ce lien pour savoir comment le construire

Les coordonnées correspondent au coin *haut gauche* du pulsar