Porjet jeu de plateau
Correction

Christophe Viroulaud

Première - NSI

Eval 05

Sommaire

Erreurs de compréhension

Trois fichiers :

- `jeu.py`
- `fonctions.py`
- `constantes.py`

Remarques

- Possibilité de séparer les fonctions par thème (création, vérification...)
- Un quatrième fichier : `tests.py`

Le fichier **constantes.py** ne contient que des valeurs fixes du jeu :

```
1 TAILLE = 5
2 ON = 1
3 OFF = 0
4 DELTA = ((0, 0), (-1, 0), (0, 1), (1, 0), (0, -1))
```

Remarque

- Ces constantes permettent de modifier rapidement les paramètres du jeu. Par exemple : agrandir la taille du plateau.
- Il faut utiliser ces constantes partout dans le jeu.

À retenir

Il ne faut pas confondre :

- représentation en mémoire (variable),
- affichage dans la console (**`print`**).

```
1 # variable contenue dans la mémoire
2 grille = [1, 1, 1]
3
4 # affichage du tableau dans la console
5 print(grille)
```

Remarque

L'instruction **`print(grille)`** demande à Python d'afficher le contenu du tableau dans la console. Cependant, pour le jeu il faut **mettre en forme** ce contenu pour qu'il soit compréhensible par le joueur.

```
1 print("|" + str(grille[0]) + "|" +
2             str(grille[1]) + "|" +
3             str(grille[2]) + "|")
```

Code 1 – Mise en forme du contenu du tableau

```
1 |1|1|1|
```

Code 2 – Sortie console

Sommaire

Construction d'un jeu - Conception générale

- **Initialiser** la grille
- Tant qu'il n'y a pas de **gagnant**
  - **Afficher** la grille
  - **Demander** les coordonnées au joueur
  - Si les coordonnées sont **valides**, **inverser** les lumières

À retenir

Chaque étape de l'algorithme générale est réalisée par une fonction. La conception détaillée permet de définir la signature des fonctions.

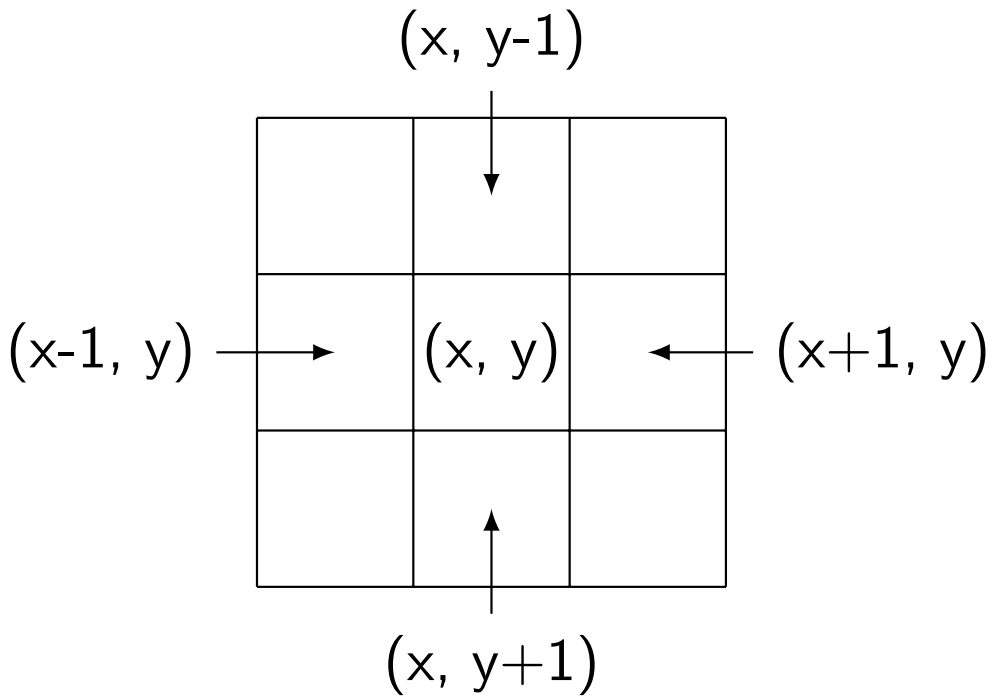
```
1  # initialisation grille
2  grille=initialiser_grille()
3
4  # boucle de jeu
5  while not verifier_gagnant(grille):
6      # afficher la grille dans la console
7      afficher_grille(grille)
8      # demande des coordonnées au joueur
9      ligne, colonne = choisir_case()
10
11     # modifie le plateau si les coordonnées sont
       acceptables
12     if verifier_coordonnees(ligne, colonne):
13         inverser_lumieres(grille, ligne, colonne)
14     else:
15         print("Coordonnées non conformes.")
16
17 # fin du jeu
18 print("gagné!")
```

Sommaire

Implémentation

```
1 DELTA = ((0, 0), (-1, 0), (0, 1), (1, 0), (0, -1))
```

Code 3 – **constantes.py**



```
1 DELTA = ((0, 0), (-1, 0), (0, 1), (1, 0), (0, -1))
```

Code 4 – **constantes.py**

```
1  def inverser_lumieres(grille: list, x: int, y: int) -> None:
2      # DELTA contient les déplacements autour de la case
       choisie
3      for dx, dy in DELTA:
4          ligne = x + dx
5          colonne = y + dy
6
7          # inversion
8          if grille[ligne][colonne] == ON:
9              grille[ligne][colonne] = OFF
10         else:
11             grille[ligne][colonne] = ON
```

Code 5 – Inversion de toutes les cellules

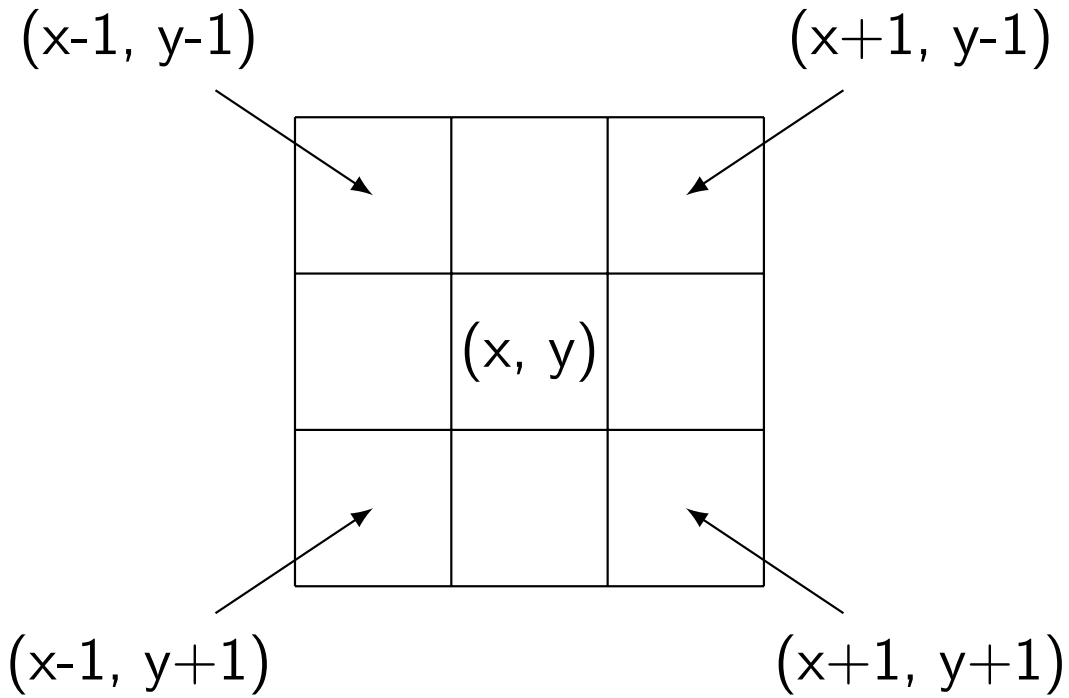
```python
def inverser_lumieres(grille: list, x: int, y: int) -> None:
    # DELTA contient les déplacements autour de la case choisie
    for dx, dy in DELTA:
        ligne = x + dx
        colonne = y + dy

        # vérifie si on est dans la grille
        if verifier_coordonnees(ligne, colonne):
            # inversion
            if grille[ligne][colonne] == ON:
                grille[ligne][colonne] = OFF
            else:
                grille[ligne][colonne] = ON
```

Code 6 – Avec vérification des coordonnées

À retenir

S'il faut vérifier les cases en diagonales, il faut juste modifier `DELTA`. La fonction ne change pas.



Situation

Vous êtes intégré dans l'équipe qui développe le jeu **Tout Éteint**. L'implémentation est bien avancée. Le chef de projet vous demande de coder les fonctions manquantes.

Activité 1 :

1. Télécharger et décompresser l'annexe **`tout-eteint.zip`** sur le site https://cviroulaud.github.io
2. Ouvrir le fichier **`fonctions_verif.py`** et implémenter les fonctions :
   - **`verifier_coordonnees`**
   - **`verifier_gagnant`**
3. Ouvrir et exécuter le fichier **`test.py`**. Un test `OK` assure de la bonne implémentation des deux fonctions.

Correction


```
1 def verifier_coordonnees(ligne: int, colonne: int) -> bool:
2     return ligne >= 0 and ligne < TAILLE and \
3             colonne >= 0 and colonne < TAILLE
```

```python
def verifier_gagnant(grille: list) -> bool:
    # parcourt la grille jusqu'à trouver une case allumée
    for i in range(TAILLE):
        for j in range(TAILLE):
            # une case (au moins) est allumée: perdu
            if grille[i][j] == ON:
                return False

    # toutes les cases sont éteintes
    return True
```