# **Projet : Sous-Marinthe**

Le but est de récupérer toutes les étoiles d’une grille en évitant des barrières que le joueur ne voit pas, avec le moins d’essais possibles.

# **Partie 1 – Génération**

Nous allons d’abord générer la grille du jeu de taille n et choisir aléatoirement en fonction de la taille de la grille un paterne prédéfini de barrières.

Les différents schémas de barrières sont enregistrés dans une liste. Chaque schéma est dans un tuple, qui contient lui-même une liste de listes. Un 0 représente l’absence de barrière, un 1 la présence de barrière. Aussi, on ne peut pas avancer à travers les bords de la grille.

|  | C1 | C2 | C3 |
| --- | --- | --- | --- |
| L1 |  |  |  |
| L2 |  |  |  |
| L3 |  |  |  |

Voici comment est écrit un schéma de barrière :

Lignes Colonnes

L1 L2 L3 C1 C2 C3

([[0,1],[1,0],[0,0]] , [[0,1],[0,0],[0,1]])

Cette liste correspond au schéma à droite

## Entrée

La fonction prend en entrée :

* La taille de la grille
* La position de départ du joueur (paramètre optionnel, aléatoire si rien n’est donné)

## Sortie

La fonction renvoie :

* La grille remplie du symbole \*, avec un O à la position du joueur
* La position du joueur dans un tableau
* La liste des barrières pour les lignes choisie
* La liste des barrières pour les colonnes correspondantes

## Exemple d’entrée/sortie

>>> crea\_grille(3, [2,0])

Plus simplement, la fonction renvoie cette grille :

['\*', '\*', '\*']  
['\*', '\*', '\*']  
['O', '\*', '\*']

# **Partie 2 – Déplacement**

## Nous allons ensuite créer une fonction qui permet de déplacer le personnage, sans prendre en compte les barrières. On peut alors se déplacer en écrivant : « haut », « bas », « gauche », « droite ». Pour plus de simplicité, on peut aussi simplement écrire l’initiale du déplacement souhaité.

NB : L’écriture de la direction avec ou sans majuscule n’a aucune importance.

## Entrée

La fonction prend en entrée :

* La direction souhaitée
* La grille du jeu
* La position du joueur avant le déplacement

## Sortie

La fonction renvoie :

* La grille modifiée
* La nouvelle position du joueur
* Une erreur si on essaye d’avancer contre un bord de la grille

## Exemple d’entrée/sortie

Pour l’exemple, la grille utilisée sera celle de l’exemple du niveau 1. On l’appellera grille.

>>> move(‘haut’, grille, [2,0])

Plus simplement, la fonction renverra cette grille :

['\*', '\*', '\*']  
['O', '\*', '\*']  
[' ', '\*', '\*']

# **Partie 3 - Jouons !**

Nous allons maintenant implémenter la notion de barrière à notre jeu. On pourra alors se déplacer de la même manière qu’au niveau 2 mais cette fois-ci on sera bloqué si notre personnage rencontre une barrière. La console affiche alors un message indiquant l’impossibilité d’avancer.

Nous allons également faire en sorte que le jeu se termine lorsque toutes les \* ont été récupérées.

## Entrée

L’entrée est la même que pour le niveau 2.

## Sortie

La fonction renvoie :

* La grille modifiée
* La nouvelle position du joueur

Cette fois-ci, en cas de déplacement impossible, la fonction affiche simplement dans la console que le déplacement est impossible et la raison (bord de la grille ou barrière). Cela évite de renvoyer une erreur, et donc d’arrêter le programme à chaque déplacement impossible.

## Exemple d’entrée/sortie

Pour l’exemple nous partons de cette grille que l’on appelle grille :

['\*', '\*', '\*']  
['O', '\*', '\*']  
[' ', '\*', '\*']

Nous admettons également qu’une barrière se trouve au-dessus du joueur. Ainsi, on peut regarder ce cas :

>>> move(‘haut’, grille, [1,0])

Il y a une barrière, impossible d'aller vers le haut

['\*', '\*', '\*']  
['O', '\*', '\*']  
[' ', '\*', '\*']

Il faut maintenant créer la boucle de jeu qui permet à l’utilisateur de jouer et indiquer à la fin un score en tenant compte du nombre de barrière touchée par l’utilisateur

**N’oubliez pas de créer un jeu de test pour chacune de ces fonctions.**

**Pensez à vous relire, commenter, indiquer la docstring … avant de rendre le projet au format console.**

# **Partie 4 : Jouez mieux !**

## 

Utiliser Pygame ou tkinter pour rendre votre jeu plus graphique.

**Partie 5 : bonus**

Utiliser un fichier csv pour stocker les scores avec utilisation d’un nom de joueur et possibilité d’afficher les scores.