Instructions

Retour

Votre mission est de concevoir et **développer un petit jeu permettant de contrôler un robot dans un labyrinthe**. Ce jeu devra être développé en console pour des raisons d'accessibilité. Je l'ai appelé... *Roboc*.

Le jeu est un labyrinthe formé d'**obstacles** : des murs qui sont tout simplement là pour vous ralentir, des portes qui peuvent être traversées et au moins un point par lequel on peut quitter le labyrinthe. Si le robot arrive sur ce point, la partie est considérée comme gagnée.

Contrôle du robot

Le robot est contrôlable grâce à des commandes entrées au clavier. Il doit exister les commandes suivantes :

- *Q* qui doit permettre de **sauvegarder** et **quitter** la partie en cours ;
- *N* qui demande au robot de se **déplacer vers le nord** (c'est-à-dire le haut de votre écran) ;
- *E* qui demande au robot de se **déplacer vers l'est** (c'est-à-dire la droite de votre écran) ;
- *S* qui demande au robot de se **déplacer vers le sud** (c'est-à-dire le bas de votre écran) ;
- *O* qui demande au robot de se **déplacer vers l'ouest** (c'est-à-dire la gauche de votre écran) ;
- Chacune des directions ci-dessus suivies d'un nombre permet d'avancer de **plusieurs cases** (par exemple *E3* pour avancer de trois cases vers l'est).

Affichage du labyrinthe

Le labyrinthe est vu du dessus. Un symbole représente un obstacle ou votre propre robot. Vous pouvez vous référez à l'exemple ci-dessous pour voir quelques exemples de partie.

Pour reconnaître la nature des obstacles, on doit bien évidemment représenter chaque obstacle par un **symbole différent**. Sans quoi, difficile de différencier les murs des portes de sorties.

Fonctionnalités du jeu

Le jeu doit :

- **Enregistrer automatiquement** chaque partie à chaque coup pour permettre de les continuer plus tard ;
- **Proposer plusieurs cartes** faciles à éditer. Chacune des cartes disponibles se trouvera dans un fichier avec l'extension *txt* dans un dossier particulier. Il sera donc facile d'**ajouter, supprimer ou modifier des cartes**. Vous pourrez télécharger les cartes par défaut plus bas.

Au lancement du programme

La première chose à faire est de trouver les cartes existantes, conservées dans nos fichiers *txt*, de les charger et de vérifier qu'une partie n'était pas en cours. Si une partie est en cours, **proposer de rejouer cette partie** (consultez l'exemple de jeu plus bas).

Choisir une carte lance la partie. La même chose se produit si vous demandez à jouer une partie déjà existante, s'il en existe une. **À chaque tour, le labyrinthe est réaffiché** avec la position de chaque obstacle et la position du robot.

Vous serez notés sur :

- Le fait d'arriver à **développer les fonctionnalités de l'exercice** : si l'on peut lancer votre programme et qu'il tourne sans modification, vous aurez la note maximale, peu importe le code source derrière ;
- La **lisibilité du code** : votre code source doit être aussi agréable à lire que possible. Les noms de vos variables, fonctions, classes, modules doivent être cohérents. La présentation de votre code source ne suit pas une règle spécifique, mais elle doit être cohérente (si vous faite un choix de nommage dans un module, faites le même choix dans un autre) ;
- Le **découpage de votre projet** : essayez de bien réfléchir à la façon dont vous découperez votre projet. Les fonctions, classes, modules et éventuellement packages formeront la structure de votre projet ;
- La **documentation de votre code** : indiquez de temps en temps des commentaires et documentations (sous forme de docstring, pour vos classes et fonctions), afin de rendre votre code plus compréhensible pour quelqu'un qui le regarde ;
- L'**ouverture à l'amélioration** : ce dernier point est donné quand votre code est aussi séparé que possible et permettrait sans difficulté des modifications, comme l'ajout d'autres obstacles dans le labyrinthe, l'utilisation de cartes 3D avec des escaliers pour circuler de niveau en niveau, l'utilisation d'un affichage graphique avec l'une des bibliothèques existantes, etc. Si votre code est bien hiérarchisé, l'amélioration est généralement plus simple. **Notez bien que vous n'avez pas à coder ces fonctionnalités, juste à garder en tête que votre programme pourrait évoluer par la suite.**

Exemples de retour

Vous pourrez trouver ci-dessous ce qu'on pourrait voir en exécutant le programme. Notez que :

- Les symboles utilisés sont O pour un mur, . pour une porte (sur laquelle le robot peut passer), U pour la sortie et X pour le robot lui-même ;
- Quand le robot passe une porte, elle devient invisible et s'affiche de nouveau quand le robot est passé ;
- Le robot ne peut pas passer au travers des murs ;
- L'exemple ci-dessous est un exemple de la carte *facile*.

```
python roboc.py

Labyrinthes existants :
  1 - facile.
  2 - prison.

Entrez un numéro de labyrinthe pour commencer à jouer : 1

OOOOOOOOOO
O O    O O
O . OO   O
O O O   XO
O OOOO O.O
O O O    U
O OOOOOO.O
O O      O
O O OOOOOO
O . O    O
OOOOOOOOOO

> s

OOOOOOOOOO
O O    O O
O . OO   O
O O O    O
O OOOO OXO
O O O    U
O OOOOOO.O
O O      O
O O OOOOOO
O . O    O
OOOOOOOOOO

> n

OOOOOOOOOO
O O    O O
O . OO   O
O O O   XO
O OOOO O.O
O O O    U
O OOOOOO.O
O O      O
O O OOOOOO
O . O    O
OOOOOOOOOO

> s2

OOOOOOOOOO
O O    O O
O . OO   O
```

```
O O O    O
O OOOO OXO
O O O    U
O OOOOOO.O
O O      O
O O OOOOOO
O . O    O
OOOOOOOOOO

OOOOOOOOOO
O O    O O
O . OO   O
O O O    O
O OOOO O.O
O O O   XU
O OOOOOO.O
O O      O
O O OOOOOO
O . O    O
OOOOOOOOOO

> e

OOOOOOOOOO
O O    O O
O . OO   O
O O O    O
O OOOO O.O
O O O    X
O OOOOOO.O
O O      O
O O OOOOOO
O . O    O
OOOOOOOOOO

Félicitations ! Vous avez gagné !
```

Cartes proposées par défaut

Vous pouvez téléchargez un fichier .zip que j'ai préparé, contenant certaines cartes par défaut. Ces fichiers txt doivent être lus par votre programme. Vous êtes libres de les modifier, d'en ajouter ou supprimer, tant que votre programme arrive à lire les fichiers de carte donnés par défaut.

Rendez-vous ici pour télécharger le fichier *.zip* contenant les exemples de carte (https://static.oc-static.com/prod/courses/files/apprenez-a-programmer-en-python/Certification%20Python%20-%20partie%20III%20-%20cartes.zip).

Code de base

L'exercice peut sembler assez compliqué, même s'il s'agit d'un jeu plutôt basique. Je vous propose néanmoins une **base de code** avec certaines fonctionnalités déjà implémentées, comme la lecture de carte. Vous n'êtes pas obligé de l'utiliser, mais cela pourra vous faire gagner du temps ! Rendez-vous par ici pour télécharger le code de base (https://static.oc-static.com/prod/courses/files/apprenez-a-programmer-en-python/Certification%20Python%20-%20partie%20III%20-%20code%20de%20base.zip).

À inclure dans votre correction

Vous devrez proposer en correction un fichier *.zip* qui devra contenir :

- **L'ensemble de votre code source**. Le fichier à exécuter doit s'appeler *roboc.py* et doit se trouver à la racine de votre code source ;
- **Une liste des cartes** proposées par le programme. Le plus simple reste de créer un dossier *cartes* dans lequel se trouve les cartes. Là encore, le programme fourni doit être capable de les trouver sans modification du code.

À vous de jouer !