TP Démineur– Python

**1 Introduction**

Ce TP est à réaliser par groupe de 3 ou 2 en Python sous l’environnement de développement de votre choix. Il sera à rendre le Mercredi 04/01/2023 à 13h37 UTC. Il sera à rendre sur Teams dans les fichiers partagés dans le répertoire « TP démineur » ou par mail à l’adresse contact@sylvaingault.fr. Vous rendrez un rapport en PDF contenant au moins les réponses aux questions (des screenshots de code la plupart du temps) ainsi que toute description nécessaire à la reproduction de vos résultats. Numérotez les questions auxquelles vous répondez dans votre rapport et incluez au moins une phrase pour chaque question. Vous joindrez également votre code, et uniquement votre code (les fichiers que vous avez rédigés). N’oubliez de lister les noms de tous les membres de votre groupe. Le but ici est d’implémenter un jeu de démineur modulaire avec des classes.

**2 Jeu démineur**

Si vous n’êtes pas familier avec ce jeu googlez son nom et faites quelques parties. Une grille 2D est donnée dont les cases sont initialement cachées. Un petit nombre d’entre elles cachent des mines (bombes). Le but du jeu est de découvrir toutes les cases ne cachant pas de mine. Découvrir une mine entraîne la perte immédiate de la partie. Découvrir une case ne cachant pas de mine révèle un nombre entre 0 et 8 indiquant le nombre total de mines dans les 8 cases adjacentes (y compris en diagonale). Il est également possible de positionner un drapeau sur les cases pour les rendre impossible à ouvrir. Celles-ci marquent l’emplacement des mines. L’ouverture d’une case contenant un 0 provoque l’ouverture immédiate des 8 cases adjacentes. Si certaines contiennent des 0, elles sont aussi immédiatement ouvertes.

**Exercice 1 Propreté du code**

Ceci n’est pas un vrai exercice. Il ne sert qu’à vous rappeler que vous serez aussi évalué sur la qualité du code.

1. L’adhésion à la PEP 8 sera évalué. Utilisez un outil tel que pylint régulièrement pour vous aider à détecter les manquements. Votre note dépendra du score fourni par la commande suivante : pylint --enable-all-extensions --disable=R0903,R6103,C0114,C0115,C0116,C0209, C2001,W0149 fichier.py
2. L’organisation globale du code fera aussi parti de l’évaluation.

**Exercice 2 Programme de base**

Le but de cet exercice est d’écrire le programme de base qui demandera les entrées à l’utilisateur et gérera l’évolution globale de la partie.

1. Écrivez un programme qui prend en argument sur la ligne de commande la taille de la grille, hauteur et largeur. Utilisez sys.argv pour accéder aux arguments.

Code :

Une image contenant texte

Description générée automatiquement

2. Faites en sorte que ce programme demande en boucle les coordonnées x et y de la case que l’utilisateur veut ouvrir. Utilisez la fonction input pour lire l’entrée au clavier.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement3. Pour être plus simple à utiliser, on définit un format d’entrées pour que la chaîne "5 2" signifie ouvrir la case à la 5ème ligne, 2ème colonne, et "F 1 3" signifie Mettre un drapeau à la 1ère ligne 3ème colonne. Votre programme pour l’instant devrait simplement parser la ligne entrée et afficher « Ouvrir la case 5, 2 » et « Flagger la case 1, 3 ».

4. Créez une classe MineSweeper qui gérera l’essentiel du déroulement du programme. Ajoutez deux méthodes open et flag. Déplacez les affichages de « Ouvrir la case x, y » et « Flagger la case x, y » dans ces méthodes et appelez-les dans votre programme principal au lieu de faire le print directement.

Une image contenant texte

Description générée automatiquement

5. Pour faire en sorte de pouvoir jouer plusieurs parties, rajoutez une méthode (quasi-vide pour le moment) new\_game pour commencer une nouvelle partie avec une taille de grille donnée, et un attribut is\_playing pour indiquer si une partie est en cours. Il est initialement à False et new\_game doit passer cet attribue à True. Faites en sorte que vos méthodes open et flag lèvent une exception si une partie n’est pas en cours.

Une image contenant texte

Description générée automatiquement

6. Modifiez votre programme pour qu’il demande en boucle les actions à l’utilisateur tant que la partie est en cours. Ajoutez une commande newgame et une commande quit que l’utilisateur pourra entrer à tout moment à la place des coordonnées d’une case. Ces commandes indiquent respectivement que l’utilisateur souhaite relancer une partie (même si une partie est déjà en cours) et qu’il souhaite quitter le programme.

Une image contenant texte

Description générée automatiquement

7. Si l’utilisateur tente d’ouvrir ou flagger une case alors qu’une partie n’est pas en cours, gérez l’exception et affichez un message correct à l’utilisateur.

Une image contenant texte

Description générée automatiquement

**Exercice 3 Grilles et cases**

Notre programme n’est pour l’instant pas vraiment un jeu. Dans cet exercice on va implémenter la mécanique du jeu.

1. Créez une classe Grid vide pour le moment. Instanciez-la dans votre méthode new\_game.

Une image contenant texte

Description générée automatiquement

2. Créez une classe abstraite Tile pour représenter une case. Chaque case connaît la grille à laquelle elle appartient, elle connaît ses coordonnées, sait si elle est ouverte ou non et sait si elle a été flaggée ou non. Elle n’a pas encore de méthode autre que \_\_init\_\_. Vous appellerez ses différents attributs \_grid, \_x, \_y, is\_open et is\_flagged. (Remarquez que les deux derniers sont publiques et les autres sont internes.)

Une image contenant texte

Description générée automatiquement

3. Créez une classe TileMine pour représenter une mine et une classe TileHint pour représenter un nombre (éventuellement 0). Toutes les deux dérivent de Tile. Aucune de ces deux classes ne prend d’argument supplémentaire pour le moment. Mais TileHint aura un attribut hint qui vaudra 0 pour le moment.

Une image contenant texte

Description générée automatiquement

4. Rajoutez une méthode \_\_str\_\_ à votre classe Tile. Elle doit retourner la chaîne de caractères "F" si la case est flaggée, "#" si elle est fermée et lever l’exception prédéfinie 2 NotImplementedError si elle est ouverte. Cette méthode étant incomplète, vous la marquerez comme abstraite.

Une image contenant texte

Description générée automatiquement

5. La méthode \_\_str\_\_ des classes TileMine et TileHint doivent appeler la méthode \_\_str\_\_ de leur classe parent si la case est fermée. Si la case est ouverte, TileMine doit retourner la lettre "O". TileHint doit retourner le chiffre sous forme de chaîne de caractère. Si le chiffre est égal à 0 elle renverra une espace.

Une image contenant texte

Description générée automatiquement

6. Dans le constructeur de votre classe Grid instanciez une liste de listes de TileHint de la taille donnée en argument. Vous la stockerez dans un attribut interne appelé \_tiles. Certains de ces objets TileHint vont être remplacé par des objets TileMine dans les 2 questions suivantes.

Une image contenant texte

Description générée automatiquement

7. Vos grilles de démineur contiendront 10% de mines. Pour choisir quelles cases contiendront des mines on créera la liste des coordonnées de toutes les cases sous la forme d’une liste de tuples. Par exemple pour une grille 2 × 3 [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)]. Puis on en prendra 10 pourcents au hasard en utilisant la fonction sample du module random. Implémentez une méthode interne \_mines\_coord dans la classe Grid qui renverra la liste des coordonnées des mines.

Une image contenant texte, moniteur, écran, capture d’écran

Description générée automatiquement

8. Maintenant que vous avez les coordonnées de vos future mines, remplacez les objet TileHint par des instances de TileMine aux coordonnées récupérées.

Une image contenant texte, capture d’écran, écran, argent

Description générée automatiquement

9. Rajoutez une méthode get\_tile à la classe Grid qui renvoie l’objet Tile à la coordonnée indiquée.

Une image contenant texte

Description générée automatiquement

10. Modifiez votre classe TileHint pour que la valeur de sa case ne soit non pas un attribut, mais une propriété. Elle calculera automatiquement le nombre de mines adjactentes en appelant la méthode get\_tile de la grille à laquelle elle appartient. Elle testera si une case est une mine en utilisant la fonction isinstance(tile, TileHint).

Une image contenant texte

Description générée automatiquement

11. Rajoutez un attribut interne \_hint initialement à None. Faites en sorte qu’il contienne la valeur du hint calculé et faites en sorte que votre propriété hint ne recalcule pas la valeur si elle est déjà présente dans \_hint.

Une image contenant texte

Description générée automatiquement

12. Rajoutez une méthode \_\_str\_\_ à votre classe Grid. Celle-ci doit retourner la chaîne de caractère qui représente la grille. Elle sera appelée automatiquement par la fonction str et par print. Vous utiliserez str(t) pour chaque tuile t pour récupérer la chaîne de caractères qui les représente.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

**Exercice 4 Comportement du jeu**

Dans cet exercice on commence à implémenter la mécanique du jeu.

1. Rajoutez une méthode open à la classe Grid. Celle-ci changera simplement l’attribut is\_open de la case donnée et ouvrira une seule case pour le moment. Elle lèvera une exception si la case est déjà ouverte ou si elle est flaggée. Appelez cette méthode dans votre classe MineSweeper, celle-ci lèvera une exception si la coordonnée est hors de la grille.

Une image contenant texte

Description générée automatiquement

2. Rajoutez une méthode toggle\_flag à la classe Grid. Celle-ci changera simplement l’attribut is\_flag de la case donnée pour le définir ou l’enlever s’il est déjà mis. Elle lèvera une exception si la case est déjà ouverte. Rajoutez une méthode similaire à votre classe MineSweeper, celle-ci lèvera une exception si la coordonnée est hors de la grille.

Une image contenant texte

Description générée automatiquement

3. Rajoutez un compteur remaining de cases non-mine qui ne sont pas encore ouvertes. La 3 méthode open devra le mettre à jour.

Une image contenant texte

Description générée automatiquement

4. Rajoutez une méthode is\_win dans la classe MineSweeper qui indique si le nombre de cases non-mines encore fermées est égal à 0.

Une image contenant texte

Description générée automatiquement

5. Rajoutez méthode is\_lost à MineSweeper qui indique si une mine a été ouverte. Vous devrez modifier votre méthode open pour définir un booléen si une case de mine est ouverte.

6. À la fin de votre méthode open, modifiez votre attribut is\_playing pour indiquer qu’une partie est en cours uniquement si elle n’est ni gagnée ni perdue en utilisant les méthodes is\_win et is\_lost. À ce stade, vous devriez déjà pouvoir jouer.

7. Pour rajouter l’ouverte des zones remplies de 0, on utilisera un algorithme récursif. Cette question ne vous demande que de réorganiser le code. L’algorithme récursif sera réellement implémenté dans la question suivante. Définissez une méthode \_open\_full dans votre classe Grid. Celle-ci appellera une méthode open sur l’objet Tile au lieu de modifier l’attribut is\_open la case donnée en argument. Appelez \_open\_full depuis votre méthode open de la classe Grid. Cette méthode \_open\_full ignorera simplement les cases déjà ouvertes contrairement à open qui lève une exception.

8. Dans votre classe Tile, implémentez une méthode open qui définit simplement is\_open. Dans votre classe TileHint définissez open pour qu’elle appelle la méthode open de la classe parent (Tile). Si le hint de la case vaut 0, appellez également la méthode \_open\_full de la grille à laquelle la case appartient sur les 8 cases adjacentes.