Travail pratique # 3
PyMineur

(en équipe de 2 ou 3)

Enseignant : Pascal Germain

Date de remise : le lundi 23 novembre 2020
Pondération de la note finale : 12%

1 Objectifs

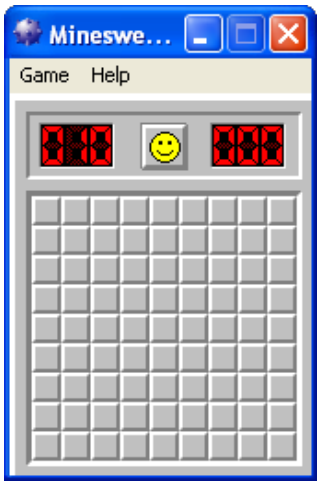
Ce travail a comme principal objectif de vous familiariser davantage avec la programmation orientée objet, via une modélisation déjà faite pour vous, que vous devez d'abord comprendre puis compléter. Ce sera également l'occasion d'utiliser la structure de données « dictionnaire ». Ce travail vous permettra de valider votre compréhension de la matière des modules 1 à 5, inclusivement. La modélisation que nous vous fournissons prend pour acquis que vous comprenez très bien le principe de la décomposition fonctionnelle et la réutilisation de fonctions. Souvent, une méthode pourra être programmée en quelques lignes de code, lorsqu'on réutilise les méthodes programmées préalablement.

2 Organisation du travail en équipe

Nous vous demandons de travailler en équipe car c'est un objectif de votre formation académique, et vous pourrez ainsi vous partager la charge de travail. Pour ceux qui n'ont pas encore trouvé de coéquipier, nous vous invitons à utiliser le forum du cours dans la section prévue à cet effet. Chaque coéquipier doit contribuer à parts égales au développement de ce travail. Laisser son coéquipier faire tout le travail (peu importe les raisons) est inacceptable : vous passerez à côté des objectifs de ce cours. De la même manière, il ne faut pas non plus trop en faire : il faut apprendre à travailler en équipe ! Chaque équipe doit être inscrite sur le portail des cours, sur la page du TP3.

3 Le problème à résoudre

Nous vous proposons de programmer le jeu Démineur en mode console, un jeu très populaire à l'époque qu'il était fourni avec le système d'exploitation Windows[1].



Pour bien comprendre le jeu et avant toute programmation, vous devez y jouer quelques fois. Vous pouvez jouer en ligne sur ce site : http ://minesweeperonline.com/#beginner

La modélisation orientée objet pour ce TP est fournie.

3.1 Les règles du jeu

Le jeu se joue avec un tableau de cases dont le contenu est caché. Ces cases contiennent soit :

- Une mine
- Une case vide
- Un nombre qui indique combien de mines la case a dans son voisinage (les huit cases sur son pourtour). Notez qu'une case vide peut être représentée par le chiffre `0`.

Les mines sont placées aléatoirement dans le tableau.

Les règles du jeu sont les suivantes :

1. Si le joueur choisit une case où une mine est cachée, la mine explose ! La partie est terminée.
2. Si le joueur choisit une case avec un nombre caché, la case est dévoilée et le nombre devient visible.

[1] Microsoft a cessé de fournir le jeu Démineur avec les installations par défaut de leur système d'exploitation à partir de Windows 8. Pour la petite histoire, voir `https://en.wikipedia.org/wiki/Microsoft_Minesweeper` (en anglais)

3. Si le joueur choisit une case vide (donc qui n'a ni mine ni nombre caché), il y a un effet en cascade (voir section plus bas) qui fait le dévoilement de toutes les cases vides dans le voisinage jusqu'à ce que la limite du tableau soit atteinte ou qu'une case avec un numéro caché soit atteinte.

L'objectif du jeu est d'identifier, par la logique, toutes les cases contenant des mines, sans en déclencher aucune.

4 Spécifications du programme

1. La taille du tableau par défaut est de 5 rangées et 5 colonnes, avec 5 mines cachées. Nous vous conseillons de d'abord programmer votre jeu avec ces valeurs par défaut, pour réduire la complexité. Quand tout fonctionnera bien, vous pourrez ensuite tester votre programme avec de plus grandes valeurs.
2. Dans le tableau, les cases vides sont représentées pas le nombre `0` ; ce qui indique que la case n'a aucune mine dans son voisinage. Le voisinage d'une case correspond à toutes les cases immédiatement adjacentes à la case horizontalement, verticalement et dans les diagonales.
3. Le programme devrait vérifier toutes les erreurs d'entrées de l'utilisateur. Deux erreurs de base : l'utilisateur entre des coordonnées qui ne sont pas des entiers pour les numéros de rangée et de colonne ; l'usager choisit des coordonnées qui ne sont pas dans le tableau. En cas d'entrée erronée, le programme demande à utilisateur d'entrer de nouvelles coordonnées. Vous penserez peut-être à d'autres types d'erreurs.
4. Si le joueur entre des coordonnées pour une case déjà dévoilée, informez-le et demandez-lui de nouvelles coordonnées.
5. À la fin de la partie, on informe le joueur s'il a gagné ou perdu et on affiche un tableau solution où toutes les cases sont dévoilées.

Important : Nous vous fournissons deux exemples de résultats de l'exécution du programme final. Consultez-les pour bien comprendre ce qui demandé.

4.1 La classe `Partie`

Le module `partie.py` contient une classe nommée `Partie`, modélisant une partie de démineur avec ses données et ses traitements. Cette classe manipule un objet de la classe `Tableau` qui est un attribut. Les méthodes de la classe `Partie` permettent de demander à l'utilisateur les coordonnées de la case à dévoiler, puis appellent les méthodes de la classe `Tableau` pour valider les coordonnées, dévoiler la case et afficher le tableau. À chaque dévoilement, on vérifie si le jeu est terminé ou s'il peut se poursuivre. Consultez le contenu du module `partie.py` pour y retrouver toutes les informations pertinentes.

4.2 La classe `Tableau`

La classe `Tableau` modélise les données et les traitements d'un tableau du jeu démineur, ayant comme principal attribut un dictionnaire de cases.

Les clés de ce dictionnaire sont des paires `(x, y)` représentant les coordonnées d'une case (une paire est en fait un tuple de deux éléments). Les coordonnées d'une case dans le tableau sont composées de deux entiers : le premier nombre correspondant à la rangée, puis le deuxième nombre correspondant à la colonne. Les éléments de ce dictionnaire sont des objets de la classe `Case`.

Les méthodes de cette classe permettent entre autres d'initialiser le tableau en y plaçant les mines, de procéder à diverses validations de coordonnées, d'afficher le tableau et la solution, puis de dévoiler des cases en exécutant l'effet en cascade de dévoilement si nécessaire. Nous vous invitons à consulter le module `tableau.py` pour y retrouver toutes les informations pertinentes.

4.3 La classe `Case`

La classe `Case` modélise les données et les traitements d'une case d'un tableau du jeu démineur. Nous vous fournissons le code de cette classe ; vous n'avez pas à modifier ce fichier. La classe a trois attributs qui spécifient si la case est minée, si elle a été dévoilée et quel est son nombre caché. Consultez le contenu du module `case.py` pour y retrouver toutes les informations pertinentes.

4.4 Algorithme de haut-niveau

1. Dans le fichier `principal.py` (que vous n'avez pas à modifier), on crée un objet de la classe `Partie`, puis on appelle la méthode « jouer ». Celle-ci doit d'abord demander à l'utilisateur les caractéristiques du tableau de jeu : le nombre de lignes, le nombre de colonnes, et le nombre de mines cachées. On initialise ensuite un objet de la classe Tableau, qui est référé par l'attribut `tableau_mines`. Lors de l'instanciation de tableau_mine, un dictionnaire (dont les clés sont les coordonnées des cases et les clés sont les cases elle-même) est initialisé.
2. Lors de l'initialisation du tableau, placez-y les mines aléatoirement. Lorsque vous placez une mine dans une case, incrémentez le nombre caché des cases voisines. Vous obtiendrez quelque chose qui ressemble à ça si vous l'affichez avec la méthode `afficher_solution()` (cette méthode vous est fournis) :

```
  | 1 2 3 4 5
--+-----------
1 | M 1 1 M 1
2 | 1 2 2 2 1
3 | 0 1 M 2 1
4 | 1 2 2 2 M
5 | 1 M 1 1 1
```

3. Affichez le tableau pour le joueur où les cases qui n'ont pas été dévoilées apparaissent comme étant cachées à l'aide de la méthode `afficher_tableau()`. Au début du jeu, toutes les cases devraient être cachées. Au fur et à mesure que le joueur choisit de dévoiler des cases, la vue du joueur doit être mise à jour (ceci est bien illustré par les exemples d'exécutions du programme qui vous sont fournis). Pour programmer la méthode `afficher_tableau`, il est conseillé de s'inspirer fortement de la méthode `afficher_solution` fournie.
4. Demandez au joueur d'entrer les coordonnées d'une case (numéros de rangée et de colonne) du tableau qu'il veut dévoiler. Les rangées et les colonnes sont numérotées à partir de 1.
5. Si le joueur dévoile une mine, montrez la solution et informez-le de sa défaite. La partie se termine.
6. Si le joueur dévoile un numéro, montrez le tableau mis à jour.
7. Si le joueur dévoile une case vide, montrez le tableau après l'effet en cascade.
8. Répétez les étapes 3 à 7 tant que la partie n'est pas finie. La partie se termine si toutes les cases ne contenant pas de mines ont été dévoilées ou si une mine est déclenchée.

4.5 Effet de dévoilement en cascade

Cas 1 - Exigence minimale pour votre TP. Si le joueur choisit une case vide dans le tableau, dévoilez les cases des voisins immédiats.

Cas 2 - Exigence facultative pour votre TP. Dans le jeu original de Démineur, si une de ces cases voisines est aussi vide, alors ses voisins à elle devraient alors être considérés pour un dévoilement, et ainsi de suite. Bien sûr, les limites du tableau sont aussi à considérer. L'implémentation de cet effet en cascade se programme naturellement avec l'appel d'une fonction récursive. Comme nous n'avons pas encore vu cette notion, ce n'est pas obligatoire. Mais ceux qui veulent un défi peuvent essayer.

5 Comment attaquer le problème

Le problème à résoudre peut paraître plutôt grand au premier coup d'œil. Commencez par bien comprendre la modélisation fournie. Lisez toute la documentation des diverses classes et méthodes, et réfléchissez à la manière dont vous pourrez résoudre les problèmes. Déjà avec le nom et la documentation des méthodes, vous devriez être en mesure de vous dire « pour programmer cette méthode, je ferai probablement appel à telle et telle autres méthodes ».

Lorsque vous programmez une méthode, assurez-vous de bien regarder quels outils sont à votre disposition. Qu'avez-vous programmé précédemment ? Comment pouvez-vous réutiliser ces méthodes ? Il est très important de programmer et tester au fur et à mesure. N'essayez surtout pas de tout programmer sans tester, vous n'y arriverez pas.

6 Les méthodes à programmer

Vous devez programmer toutes les méthodes qui ne sont pas déjà programmées, ainsi que compléter celles qui sont incomplètes. Chacune de ces méthodes possède un ou plusieurs commentaires `# TODO`, qui vous indiquent que vous devez programmer la méthode, ou la compléter. Attention de ne pas en oublier !

Si vous comprenez bien l'interaction entre les diverses méthodes, chacune d'entre elles devrait être utilisée à au moins un endroit dans votre programme.

7 Tests unitaires

Nous vous demandons de tester vos méthodes à l'aide de tests unitaires. Plus précisément, vous devez compléter les tests unitaires à la fin du fichier `tableau.py`.

8 Modalités d'évaluation

Ce travail sera évalué sur 100 points, et la note sera ramenée sur 12. Voici la grille de correction :

Élément	Pondération
Le programme initialise correctement le tableau de mines	15 points
Le programme affiche le tableau selon l'état de dévoilement des cases	5 point
Le programme valide les coordonnées	5 points
Le programme valide les entrées de l'utilisateur	5 points
Le programme permet de choisir les options de jeu	5 points
Le programme implémente l'effet de dévoilement en cascade (de base)	10 point
Le programme permet un bon déroulement de la partie	20 point
La partie se termine si le joueur déclenche une mine	5 point
La partie se termine si le joueur a dévoilé toutes les cases sans mine	5 point
Tests unitaires de la classe `Tableau`	10 points
Respect de la décomposition fonctionnelle demandée	10 points
Qualité du code (noms de variables, style, commentaires, documentation)	5 points

Notez qu'un programme qui n'est pas fonctionnel (qui ne s'exécute pas ou qui dès plante les premières lignes l'exécution) pourrait recevoir une note de 0.

9 Remarques

Plagiat : Comme décrit dans le plan de cours, le plagiat est strictement interdit. Ne partagez pas votre code source à quiconque. Une politique stricte de tolérance zéro est appliquée en tout temps et sous toute circonstance. Tous les cas détectés seront référés à la direction

de la faculté. Des logiciels comparant chaque paire de TPs pourraient être utilisés pour détecter les cas de plagiat.

Retards : Une pénalité de 25% sera appliquée pour un travail remis le lendemain de la remise. Une note de 0 sera attribuée pour un plus long retard.

Remises multiples : Il vous est possible de remettre votre TP plusieurs fois sur le portail des cours, en conservant le même nom de fichier. La dernière version sera considérée pour la correction. Ne laissez pas plusieurs fichiers avec des noms différents sur le portail.

Respect des normes de programmation : Nous vous demandons de prêter attention au respect des normes de programmation établies pour le langage Python, notamment de nommer vos variables, fonctions et méthodes en utilisant la convention suivante : `ma_variable`.

Mots-clés interdits : Comme pour les TP précédents, nous vous interdisons d'utiliser les instructions `break` et `continue`.

Nouvelles méthodes et nouveaux attributs : Vous pouvez ajouter des méthodes et des attributs aux classes fournies. Le cas échéant, vous devez documenter ces ajouts à l'aide de « docstrings ».

10 Ce que vous devez rendre

Votre programme doit être rédigé à même les fichiers Python fournis, que vous devez compresser dans une archive `.zip`. Vous devez remettre donc une archive `.zip` d'un dossier, contenant uniquement :

- `correction.txt`
- `principal.py`
- `partie.py`
- `tableau.py`
- `case.py`

Cette archive doit être remise via le site Web du cours.

Bon travail !