Travail pratique # 3
PyDomino

(seul ou en équipe de 2)

Simon Hardy

Date de remise : le lundi 2 avril 2018
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 travail vous permettra de valider votre compréhension de la matière des modules 1 à 6, 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

Que vous choisissiez de faire ce travail seul ou en équipe, vous devez inscrire votre équipe sur le portail des cours, sur la page du TP3. Vous n'aurez pas accès à la boîte de dépôt sinon. Si vous optez pour le travail en équipe, nous vous demandons de traiter votre coéquipier ou coéquipière avec respect, de maintenir une communication ouverte avec lui ou elle.

Nous vous fournissons sur le site Web du cours un tutoriel pour utiliser des technologies permettant de partager votre code avec votre coéquipier. Chaque membre d'une équipe possède sur son ordinateur une copie locale du code, mais le code est synchronisé par un service dans le «cloud» et nous pouvons également voir l'historique (qui a fait quoi et quand). De plus, PyCharm intègre ces outils de manière très conviviale.

3 Le problème à résoudre

Nous vous proposons de programmer le jeu domino en mode console.

Pour bien comprendre le jeu et avant toute programmation, vous devez y jouer ! Vous pouvez jouer en ligne sur ce site : https ://dominoes.playdrift.com/ Il existe un très grand nombre de variantes de ce jeu millénaire. Attention : Nous utiliserons des règles différentes de celles du jeu en ligne.

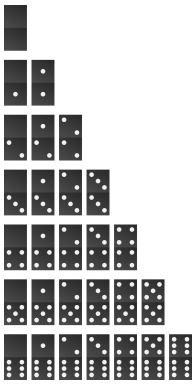
La modélisation orientée objet pour ce TP est fournie, dans un package nommé `pydomino`. Un package est un dossier tout à fait standard, contenant des modules Python, et un module spécial nommé `__init__.py`. Importer des modules qui sont situés dans un package se fait de la manière suivante :

```
from nom_package.nom_module import nom_fonction
from nom_package.nom_module import NomClasse
```

Nous vous rappelons finalement que pour qu'un package soit correctement reconnu par PyCharm, vous devez créer un projet (ou ouvrir le dossier) contenant ce package. Ouvrir les fichiers `.py` individuellement ne fonctionnera pas.

3.1 Les règles du jeu

Le jeu de base que vous devez programmer doit utiliser un ensemble de dominos "double-six".



Les règles du jeu sont les suivantes :

Le jeu peut être joué par 2, 3 ou 4 joueurs. À partir d'un ensemble de 28 dominos "double-six", chaque joueur a une donne de dominos constituée au hasard (son jeu). Pour une partie à 2 joueurs, chaque joueur reçoit 7 dominos. Pour une partie à 3 ou 4 joueurs, chaque joueur reçoit 6 dominos.

Le premier joueur à déposer un domino est celui qui a le domino le plus élevé (les trois dominos les plus élevés sont [6,6], [5,6] et [5,5]). Il doit nécessairement déposer ce domino

en premier. Ensuite, chaque joueur joue à tour de rôle. Pour déposer un domino, un joueur doit choisir un domino de sa donne qui a un numéro identique à une des extrémités de la suite de dominos qui s'assemble sur la table. Par exemple, si la suite sur la table est la suivante : [3,6][6,6][6,5], le joueur dont c'est le tour peut jouer un domino contenant soit le numéro 3 (par exemple, [2,3]), soit le numéro 5 (par exemple [5,5]). Si un joueur ne peut jouer aucun domino, il doit passer son tour.

La partie se termine dans deux conditions :

1. Si un joueur a déposé tous ses dominos et a vidé sa donne. Ce joueur est déclaré gagnant.
2. Si aucun joueur ne peut déposer un domino (ils auront donc tous passé leur tour), la partie est arrêtée. Le joueur à qui il reste le moins de dominos dans sa donne est déclaré gagnant. En comptant le nombre de dominos restants, il se peut qu'il y ait égalité entre 2 joueurs ou plus.

Dans un deuxième temps, nous vous demandons de programmer un deuxième type de parties de dominos : une partie avec une pioche. Dans ce type de partie, un joueur qui n'a pas de dominos à déposer sur le jeu doit piger un domino dans une pioche (les dominos restants qui n'auront pas été distribués au départ) jusqu'à ce qu'il pige un domino qu'il peut jouer. Si la pioche est vide, il passe simplement son tour. Les conditions de fin de partie avec pioche sont identiques aux conditions de fin de partie sans pioche.

4 Spécifications du programme

1. Le programme doit afficher les instructions selon le type de partie choisi.
2. Toutes les entrées au clavier doivent être validées (type de partie, nombre de joueurs, choix de domino, etc). En cas d'erreur, le programme redemande l'information à l'utilisateur tant qu'une information valide n'a pas été entrée.
3. Si un domino peut être joué à gauche ou à droite du plateau, on demande à l'utilisateur d'indiquer le côté où il veut placer son domino.
4. À chaque tour, l'information pertinente est affichée à l'écran : le numéro du joueur à qui s'est le tour ; l'état du plateau ; la quantité de dominos dans la donne de chaque joueur ; et finalement le contenu de la donne du joueur dont c'est le tour.
5. Le programme doit détecter lorsqu'un joueur doit passer son tour.
6. Le jeu se termine lorsqu'une partie est conclue.

4.1 La classe `Partie`

Le module `partie.py` contient une classe nommée `Partie`, modélisant une partie de domino sans pioche avec ses données et ses traitements. Cette classe manipule un objet de la classe `Plateau` ainsi qu'une liste de donnes qui sont tous les deux des attributs. Pour instancier un objet de la classe `Partie`, on utilise la méthode de classe `nouvelle_partie()` qui initialise le plateau et les donnes selon le nombre de joueurs. La fonction principale de la classe `Partie` est la fonction `jouer()` qui contient la boucle de répétition pour chaque tour de jeu.

Les autres méthodes de cette classe accomplissent différentes tâches nécessaires à une partie de domino comme : afficher les instructions, déterminer le premier joueur (celui qui a le domino le plus élevé), déterminer si un joueur peut jouer un domino à son tour, jouer un domino sur le plateau, et quelques autres...

4.2 La classe `Plateau`

La classe `Tableau` modélise les données et les traitements d'un tableau du jeu de domino, ayant comme principal attribut une liste de dominos qui est vide initialement[1].

Les méthodes de cette classe permettent de connaître les valeurs numériques aux extrémités du plateau. Elles permettent aussi d'ajouter des dominos à gauche ou à droite du plateau.

4.3 La classe `Donne`

La classe `Donne` modélise les données et les traitements de la donne d'un joueur de domino. Cette classe est relativement simple et les méthodes à programmer tiennent toutes en quelques lignes chacune. La classe a un attribut qui est une liste de dominos. Deux méthodes sont à programmer : `jouer()`, qui retire un domino de la donne, et `piger()`, qui ajoute un domino à la donne. Consultez le contenu du module `donne.py` pour y retrouver toutes les informations pertinentes.

4.4 La classe `Domino`

La classe `Domino` modélise les données et les traitements d'une pièce de domino. Cette classe est très simple. La classe a deux attributs : deux nombres entiers pour les deux chiffres sur le domino. Deux méthodes sont à programmer : `inverser()`, qui renvoie un domino où le premier chiffre et le deuxième chiffre sont inversés, et `__str__()`, qui spécifie la représentation d'un objet de cette classe sous forme de chaîne de caractères.

1. Pour ceux qui veulent essayer, le conteneur `deque` pour aussi être utilisé pour contenir les dominos joués. Pour utiliser `deque`, il faut l'importer du module `collections`.

4.5 Les classes `Pioche` et `PartieAvecPioche`

Les classes `Pioche` et `PartieAvecPioche` modélisent les données et les traitements d'une partie de dominos où l'on joue avec une pioche. Ces classes sont des classes dérivées. La classe `Pioche` hérite de la classe `Donne`. Elle définit une méthode spécifique à cette classe : la méthode `prendre_dans_la_pioche()`. La classe `PartieAvecPioche` hérite de la classe `Partie`. Six méthodes sont redéfinies dans cette classe. Consultez le contenu du module `partie_avec_pioche.py` pour y retrouver toutes les informations pertinentes. Ces classes doivent être développées uniquement lorsque le reste du jeu de dominos est fonctionnel.

4.6 Le module `__main__.py`

Lorsque nous travaillons avec un package (composé de modules), il est possible de créer un module nommé `__main__.py`, qui sera le point d'entrée du package, c'est à dire que c'est ce module qui sera exécuté si un utilisateur exécute directement votre package. C'est ce module qui contient le code pour démarrer jouer une partie de domino. Le programme demande quel type de partie on veut jouer, il instancie l'objet de type Partie, puis on appelle la méthode `jouer()`.

4.7 Algorithme de haut-niveau

1. Dans le module `__main__.py` du package, vous créez un objet de la classe Partie à l'aide de la méthode `nouvelle_partie()`. Dans le constructeur de Partie, on crée un objet `plateau` de la classe `Plateau` ainsi qu'une liste de donnes selon le nombre de joueurs choisis.
2. Ensuite, on démarre la partie en appelant la méthode `jouer()`. Ceci lance l'affichage des instructions, on détermine le premier joueur (celui qui a le domino le plus élevé), et on enchaîne avec le tour des joueurs suivants.

5 Comment attaquer le problème

Le problème à résoudre étant plutôt grand, 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 méthode ».

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. Chacune de ces méthodes possède un commentaire `# TODO`, qui vous indique que vous devez programmer la méthode. Attention de ne pas en oublier !

Si vous comprenez bien l'interaction entre les diverses méthodes, chacune d'entre elles devrait être utilisée. Nous vous suggérons de tester vos méthodes à l'aide de tests unitaires. Nous vous fournissons un répertoire avec des exemples que vous pouvez compléter. Ceci dit, les tests unitaires ne seront pas corrigés et ils ne doivent pas être remis.

6.1 Fonctionnalités facultatives

Pour les équipes qui veulent en faire plus, vous pouvez programmer d'autres variantes du jeu de dominos (voir prochaine section sur le prix Pierre Ardouin).

7 Le prix Pierre Ardouin

Depuis l'automne 2013, le département d'informatique et de génie logiciel a mis en place un concours récompensant l'équipe qui aura produit le meilleur TP/projet dans le cadre d'un cours. Ces travaux de session ont l'envergure d'un mini-projet qui est admissible par rapport aux normes fixées par le département. À la suite des évaluations des travaux, l'enseignant du cours détermine l'équipe gagnante ; chaque membre de l'équipe gagnante reçoit alors une bourse de 50$ ainsi qu'une attestation remise par le département.

De plus, le département d'informatique et de génie logiciel a mis en place une bourse Élite, appelée bourse « Pierre Ardouin », qui vise à récompenser le meilleur projet de session, tous cours confondus. Deux principaux critères guident le choix des évaluateurs dans l'identification du lauréat : l'excellence du travail (par rapport à ce qui est demandé dans l'énoncé) et l'aspect créativité/innovation. Il est actuellement prévu une bourse de 200$ pour récompenser chaque membre de l'équipe « élite » gagnante (pour un maximum de 1000$ pour toute l'équipe). Aussi, le département veille à publier l'information sur un site Web dédié : http ://www.ift.ulaval.ca/vie-etudiante/prix-pierre-ardouin.

À la deuxième moitié du mois de mai de chaque année universitaire, le département organise une cérémonie pour honorer les finalistes et le lauréat du prix « Pierre Ardouin » des sessions d'automne et d'hiver, et leur remettre une attestation.

Le travail pratique 4 sera une continuité du travail pratique 3, et l'ensemble de ces deux travaux sera évalué pour le prix Pierre Ardouin. Notez que l'aspect créativité/innovation sera particulièrement évalué sur les éléments du TP4, qui contiendra une interface graphique.

8 Modalités d'évaluation

Ce travail sera évalué sur 17 points. Voici la grille de correction :

Élément	Pondération
Classe `Domino`	1 point
Classe `Donne`	1 point
Classe `Plateau`	1 point
Fonction `distribuer_dominos`	1 point
Méthode `Partie.afficher_instructions`	0,5 point
Méthode `Partie.trouver_premier_joueur`	0,5 point
Méthode `Partie.tour_du_premier_joueur`	1 point
Méthode `Partie.determiner_si_joueur_joue_ou_passe`	0,5 point
Méthode `Partie.afficher_informations_debut_tour`	0,5 point
Méthode `Partie.demander_numero_domino_a_jouer`	0,5 point
Méthode `Partie.jouer_un_domino`	1 point
Méthode `Partie.tour_du_prochain_joueur`	1 point
Méthode `Partie.trouver_joueur_avec_moins_de_dominos`	0,5 point
Méthode `Partie.afficher_message_egalite`	0,5 point
Méthode `Partie.afficher_message_gagnant`	0,5 point
Méthode `Partie.jouer`	1 point
Module `__main__.py`	1 point
Classe `Pioche`	1 point
Classe `PartieAvecPioche`	1 point
Qualité du code (noms de variables, style (PEP8), commentaires, documentation)	2 points

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

9 Remarques

Plagiat : Tel que 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 : Tout travail remis en retard peut être envoyé par courriel à l'enseignant si le portail des cours n'accepte pas la remise. 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 et fonctions en utilisant la convention suivante : `ma_variable`, `fichier_entree`, etc. Utiliser PyCharm s'avère être une très bonne idée ici, car celui-ci nous donne des indications sur la qualité de notre code (en marge à droite, et souligné).

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`. Ce TP étant développé dans un package, il vous faut zipper le dossier dans lequel se trouve votre package. Vous devez remettre donc une archive `.zip` d'un dossier, contenant :

- `correction.txt`
- dossier `pydomino`
  - `__init__.py`
  - `__main__.py`
  - `partie.py`
  - `plateau.py`
  - `donne.py`
  - `domino.py`
  - `partie_avec_pioche.py`
  - `pioche.py`

Vous n'avez pas à remettre le dossier de tests unitaires. Cette archive doit être remise via le site Web du cours.

Bon travail !