Travail pratique # 3
Jeu de dames

Pascal Germain
Walid Boulabiar

Date de remise : le mercredi 20 décembre à minuit
Pondération de la note finale : 16%

1 Objectifs

Ce travail comprend deux volets :

1. En premier lieu, vous devez compléter les fonctions présentées dans les fichiers damier.py, position.py et partie.py, qui se trouvent dans le dossier Partie1, pour pouvoir jouer au jeu de dames avec un affichage en mode console.

2. En second lieu, une fois que nous aurons abordé la matière du module 8 de notre cours (Les interfaces graphiques), vous aurez à compléter une interface graphique en utilisant la librairie tkinter. Pour ce faire, vous devez compléter le fichier interface_dames.py, qui se trouve dans le dossier Partie2.

À la racine du dossier tp3, vous trouverez deux fichiers main_partie1.py et main_partie2.py qui permettent, respectivement, le lancement du jeu en mode console et graphique.

Ce travail vous permettra de vous familiariser davantage avec les structures de données principales de Python comme les listes et les dictionnaires, ainsi que la modélisation de programmes informatiques, via le paradigme orienté objet, et de vous familiariser avec la création d'interfaces graphiques. La majeure partie de la modélisation du problème est déjà faite pour vous. Vous devez prendre le temps de bien lire la documentation fournie dans les modules afin de découvrir les liens entre les diverses classes que vous devez compléter.

Finalement, ce travail vous permettra de valider votre compréhension de la matière des modules 1 à 9, inclusivement. La modélisation que nous vous fournissons tient pour acquis que vous comprenez bien le principe de la décomposition fonctionnelle et la réutilisation de fonction. Souvent une méthode peut être programmée en trois ou quatre lignes lorsqu'on réutilise les méthodes déjà programmées préalablement.

2 Organisation du travail en équipe

Nous vous demandons de travailler en **équipe de deux ou trois**, car c'est un objectif de votre formation universitaire, et vous pourrez ainsi vous partager la charge de travail[1]. **Chaque équipe doit être formée sur le portail des cours, sur la page du TP3.** 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 !

Nous vous fournirons 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 « nuage » (« *cloud* ») et nous pouvons également voir l'historique (qui a fait quoi et quand). De plus, PyCharm intègre ces outils de manière très efficace. Bien que cela soit facultatif, nous vous recommandons vivement d'utiliser un tel service, ici on parle du Git et GitHub. Au besoin, de l'aide peut être donnée durant les séances de TD et tutorat.

3 Comment attaquer le problème

Le problème à résoudre est d'une envergure certaine, **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 ».

Commencez par développer les fonctions nécessaires, dans les différents modules (fichiers), pour permettre le bon fonctionnement du jeu avec un affichage en mode console, puis compléter le module interface_dame.py pour un affichage en mode graphique et en développent les fonctions demandées.

3.1 Partie 1

Nous vous fournissons les fichiers de codes suivants dans le dossier Partie 1 :

position.py, damier.py, partie.py, piece.py, interface_dame.py et canvas_damier.py

Commencez par programmer les méthodes demandées dans position.py, et **testez-les**. Programmez ensuite, dans l'ordre, les méthodes de damier.py, et **testez chacune d'entre elles individuellement**. Complétez avec la programmation de partie.py. Lorsque vous programmez une méthode, assurez-vous de bien regarder quels outils sont à votre disposition.

[1] Pour obtenir la permission d'effectuer le travail individuellement, envoyez une demande bien justifiée à l'enseignant le plus tôt possible.

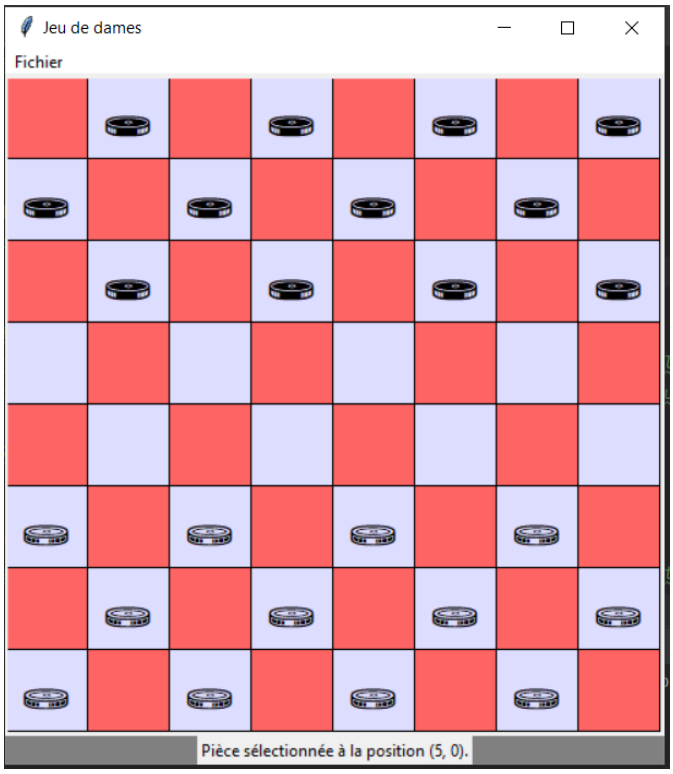
Qu'avez-vous programmé précédemment ? Comment pouvez-vous réutiliser ces méthodes ? Il est très important de programmer et tester au fur et à mesure. **N'essayez surtout pas de tout programmer sans tester**, vous n'y n'arriverez pas. Vous avez réalisé des tests unitaires dans votre TP2, utilisez cette technique pour valider votre programmation.

3.2 Partie 2

Une fois votre jeu est fonctionnel avec un affichage en mode console, vous pouvez compléter les méthodes dans interface_dame.py du dossier Partie2, pour permettre un affichage en mode graphique.

4 Le problème à résoudre

Vous devez réaliser un jeu de dames simplifié, où deux joueurs s'affrontent. La modélisation dans le paradigme orienté objet est déjà réalisée pour vous : vous n'avez qu'à programmer le contenu des différentes fonctions, déjà documentées. **Vous devez respecter la modélisation, vous ne pouvez pas changer le comportement des méthodes**. Vous avez en revanche le droit de créer de nouvelles méthodes supplémentaires au besoin (veillez à documenter ces nouvelles méthodes).



4.1 Le jeu de dames simplifié

La version simplifiée du jeu de dames que vous devez réaliser est un mélange du jeu de dames international et de la version anglaise *Checkers*. En voici les règles.

- Un *damier* de 8 cases par 8 contient 24 *pièces* (12 blanches, 12 noires). La position initiale est illustrée dans la figure suivante ;
- Le joueur avec les pièces blanches commence en premier ;
- Une pièce de départ est appelée un *pion*, et peut se déplacer **en diagonale** vers l'avant (vers le haut pour les blancs, vers le bas pour les noirs). Une case doit être libre pour pouvoir s'y déplacer ;
  Lorsqu'un pion atteint le côté opposé du plateau, il devient une *dame*. Cette action se nomme la *promotion*. Une dame a la particularité qu'elle peut aussi se déplacer vers l'arrière (toujours en diagonale) ;
- Une *prise* est l'action de « manger » une pièce adverse. Elle est effectuée en sautant par-dessus la pièce adverse, toujours en diagonale, **vers l'avant ou l'arrière**. On ne peut sauter par-dessus qu'une pièce adverse à la fois : il faut donc que la case d'arrivée soit libre ;
- Après une prise, le joueur courant peut effectuer une (ou plusieurs) prise(s) supplémentaire(s) en utilisant la même pièce ;
- Lors du tour d'un joueur, si celui-ci peut prendre une pièce ennemie, il doit absolument le faire (on ne peut pas déplacer une pièce s'il était possible d'effectuer une prise) ;
- On déduit des deux dernières règles que lorsqu'un joueur commence son tour et prend une pièce adverse, s'il peut continuer son tour en continuant de prendre des pièces adverses avec la même pièce, il **doit** le faire.

4.2 La modélisation de la solution

La solution est modélisée en 6 classes :

- une classe Piece, représentant une pièce du jeu ;
- une classe Position, représentant une position matricielle (ligne, colonne) ;
- une classe Damier, modélisant le damier du jeu ;
- une classe Partie, modélisant une partie de dames entre deux joueurs ;
- une classe FenetrePartie, modélisant une fenêtre graphique contenant un damier ;
- une classe CanvasDamier, modélisant l'affichage à l'écran des composants de la fenêtre graphique du jeu de dames.

Vous trouverez dans les fichiers joints les indications pour chaque méthode à programmer.

Un fichier par classe est fourni, et ceux-ci sont à compléter sauf piece.py et canvas_damier.py auxquels le code est offert en totalité. La modélisation étant déjà faite pour vous, nous avons inclus en commentaires une quantité substantielle d'information expliquant ce qu'il y a à faire pour chaque méthode (fonction). Quelques méthodes sont déjà programmées pour vous, et le reste est à faire.

Notez que le fichier partie.py représente—en plus de la classe une classe Partie—le point d'entrée du programme si vous êtes à l'étape de l'affichage en mode console. Une fois toutes les fonctions complétées, vous pourrez jouer aux dames en exécutant le fichier main_partie1.py. Les fichiers position.py et damier.py—en plus du code des classes respectives—un bloc de tests unitaires. Il vous est demandé de compléter ces tests unitaires, au moins 2 tests par fonction.

Les fichiers interface_dame.py et canvas_damier.py se trouvent dans le dossier Partie2, permettent la création de l'interface graphique du jeu de dames :

- interface_dame.py est la base du programme que vous devez compléter et modifier pour intégrer les fonctionnalités demandées. Pour jouer au jeu en mode d'affichage graphique, il faut exécuter le fichier main_partie2.py qui appelle interface_dame.py. Vous constaterez que nous avons programmé un évènement qui détecte lorsque l'utilisateur clique sur une pièce, mais que le jeu de dames n'est pas fonctionnel. Commencez par bien lire et comprendre le code fourni. Ensuite, modifiez-le autant que vous le désirez.
- canvas_damier.py contient la définition de la classe « CanvasDamier », qui est un

  « widget » créé pour afficher à l'écran un objet « Damier ».

  Votre interface graphique doit comprendre les fonctions suivantes, donner un nom significatif ainsi qu'une documentation pour chaque fonction créée :

  1. Les pièces peuvent être déplacées sur le damier
  2. Les déplacements invalides ne sont pas acceptés
  3. On peut choisir une nouvelle pièce source après un mouvement invalide
  4. Si une prise est possible, elle est obligatoire
  5. Les prises multiples sont gérées correctement
  6. Le tour des joueurs est alterné correctement
  7. On peut créer une nouvelle partie
  8. On peut quitter le jeu
  9. Une rétroaction est donnée à l'utilisateur (messages)
  10. L'interface doit afficher à l'utilisateur la couleur du joueur qui doit faire un déplacement (tour courant)

5 Modalités d'évaluation

Ce travail sera évalué sur 100 points, selon la grille d'évaluation suivante :

Fonctionnalités		Pondération
Classe Position	positions_diagonales_haut()	2 points
	quatre_positions_diagonales()	3 points
	quatre_positions_sauts()	3 points
	Tests unitaires de la classe Position	3 points
Classe Damier	position_est_dans_damier()	3 points
	piece_peut_se_deplacer_vers()	5 points
	piece_peut_sauter_vers()	5 points
	piece_peut_se_deplacer()	3 points
	piece_peut_faire_une_prise()	3 points
	piece_de_couleur_peut_se_deplacer()	3 points
	piece_de_couleur_peut_faire_une_prise()	3 points
	deplacer()	6 points
	Tests unitaires de la classe Damier	4 points
Classe Partie	position_source_valide()	5 points
	position_cible_valide()	5 points
	demander_positions_deplacement()	6 points
	tour()	6 points
Interface graphique	Les pièces peuvent être déplacées sur le damier	3 points
	Les déplacements invalides ne sont pas acceptés	3 points
	On peut choisir une nouvelle pièce source après un mouvement invalide	3 points
	Si une prise est possible, elle est obligatoire	3 points
	Les prises multiples sont gérées correctement	3 points
	Le tour des joueurs est alterné correctement	3 points
	On peut créer une nouvelle partie	3 points
	On peut quitter le jeu	3 points
	Une rétroaction est donnée à l'utilisateur (messages)	3 points
	Affiche la couleur du joueur qui doit faire un déplacement (tour courant)	3 points
Qualité du code (noms de variables, style, docstrings)		2 points
	Total	100 points

6 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.

Retards : Les travaux pratiques doivent être impérativement envoyés via le portail des cours. Aucune remise par courriel n'est acceptée. Un travail qui ne respecte pas les directives de remise se verra pénalisé de 20%.

Tout travail remis en retard se verra pénalisé de 25% par jour de retard. Chaque journée de retard débute dès la limite de remise dépassée (dès la première minute). Un retard excédant 2 jours provoquera le rejet du travail pour la correction et la note de 0 pour ce travail.

Remises multiples : Il vous est possible de remettre votre TP plusieurs fois sur le portail des cours. Seulement la dernière version sera considérée pour la correction. La correction des TPs ne commence qu'après la date limite du dépôt du TP.

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 indentation, marge à droite, et souligné).

Documentation du programme : Si vous créez de nouvelles fonctions, vous devez les documenter à l'aide de « docstrings ». La seule exception à cette règle est les fonctions de tests unitaires, pour lesquelles nous n'exigeons pas de commentaires.

7 Ce que vous devez rendre

Vous devez rendre une archive .zip contenant les fichiers suivants en gardant la même arborescence des dossiers du TP:

- Dans le dossier Partie1 :
    - Le fichier piece.py sans modification ;
    - Le fichier position.py, complété ;
    - Le fichier damier.py, complété ;
    - Le fichier partie.py, complété ;
- Dans le dossier Partie2 :
    - Le fichier interface_dame.py, complété ;
    - Le fichier canvas_damier.py, sans modification ;
- Le fichier de correction, complété avec vos noms et NI.
- Le fichier main_partie1.py sans modification ;
- Le fichier main_partie2.py sans modification ;

Ne remettez aucun autre fichier ou dossier que les neuf fichiers mentionnés ci-haut. Les travaux comprenant des fichiers inutiles (tel le répertoire de projet .idea de PyCharm ou le répertoire de fichiers précompilés __pycache__ généré par l'interpréteur Python) seront pénalisés.

Cette archive doit être remise via l'utilitaire de remise de travaux, disponible sur le site Web du cours. Ne remettez qu'un travail par équipe.

Bon travail !