2024

KIT APPRENANT





INTRODUCTION

à l'Object Oriented Programming (OOP) en Python



2024



Partie 2: Othello

Objectif

L'objectif de cette seconde partie est de mettre en pratique les connaissances acquises en OOP au travers d'un cas d'étude : développement d'un jeu d'Othello.

Modalités

- Activité devant être réalisée en groupe (2 à 3 personnes)
- Durée : 2 à 2.5 jours

Démarche pédagogique

- Vous ne coderez pas cette activité sur un Jupyter Notebook mais sur VS Code: (https://code.visualstudio.com/download)
- Vous organiserez votre code en **modules** et **packages** (https://dev.to/codemouse92/dead-simple-python-project-structure-and-imports-38c6)
- Vous créerez un environnement virtuel pour ce projet (https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html)
- Vous générerez un fichier requirements.txt

Compétences associées à l'activité

• Toutes les compétences ont déjà été validées lors de la 1ère partie.

Règles de l'Othello

• Vous pouvez trouver les règles du jeu ici : https://www.ffothello.org/othello/regles-du-jeu/

Déroulé du projet

Ce projet est découpé en deux étapes :

- 1. Développement du jeu d'Othello.
- 2. (Optionnel) Développement d'un adversaire artificiel contre lequel jouer.

KIT APPRENANT



1. <u>Développement du jeu d'Othello.</u>

L'idée n'est pas de créer un très beau jeu mais plutôt un jeu fonctionnel. Vous le développerez en mode **console** uniquement. Ci-après un exemple de rendu possible avec un **O** pour les blancs et un * pour les noirs (vous pouvez changer la notation par un **X** par exemple)

O's move (? for help) g5 Great! You captured 2 pieces! I moved at G3 and captured 2 of your pieces!												
1	+ I		+ 	1		+ 			+- 			
2 1			+ 			+ 						
3			+ *			+ 						
4 1			+ 			+ I 0						
5 1			+ 			+ *						
6 1			i 0	1		+ 						
7 1			+ 			 						
8 1	+		+ 			+ 			†- 			
0's:	nove	(?	+ for	-+- - h	ne l p	+	+		+-		++	

Un exemple de rendu possible

Organisez votre code en classes : par exemple une classe board, pawn, engine,...

Afin de rendre le découpage de votre code le plus clair possible, il vous est recommandé de créer **un fichier .py par classe**.

Pensez à vous répartir le travail en équipe!

Nous vous conseillons très fortement de passer par une étape de **planification papier/ stylo** pour écrire le programme. Quelles classes allez-vous créer ? Quels attributs et quelles méthodes dans quelles classes ? Etc.

Il est possible de vous inspirer du langage de modélisation d'UML (https://www.tutorialspoint.com/uml/uml basic notations.htm) pour schématiser votre planification. Ce type de langage permet de représenter les différents attributs et méthodes utilisés dans une classe ainsi que les relations de cette classe avec les autres.

Il existe de nombreux outils permettant de gérer simplement des diagrammes UML. Par exemple :

OOP en Python



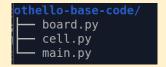
2024 KIT APPRENANT

- DotUML (web)¹: https://dotuml.com/playground.html
- WebSequenceDiagrams (web, diagrammes de séquence uniquement) : https://www.websequencediagrams.com/#
- LucidChart (web, limited free signup): https://www.lucidchart.com/pages/
- StarUML (application native) : https://staruml.io/



En cas de trop grande difficulté sur ce projet très peu guidé jusqu'à présent une base de code est disponible :

• Ce dossier contient 3 fichiers :



- En vous plaçant dans le dossier vous pouvez lancer le jeu en exécutant la commande **python3 main.py**
- Cette base de code permet uniquement d'afficher le jeu.
- Nous pourrons également discuter en groupe de l'organisation de votre code : Quelles classes créer ? Quelles fonctions ?

¹ Extension VS Code: https://marketplace.visualstudio.com/items?itemName=BMLSolutions.dotuml

KIT APPRENANT



2024

2. (Optionnel) Développement d'un adversaire artificiel.

Pour créer un joueur artificiel avec une intelligence basique, des algorithmes basés sur des arbres évaluant les coups possibles sont souvent utilisés.

Dans notre cas l'**algorithme minimax** sera utilisé pour créer un programme capable de jouer à l'Othello (https://fr.wikipedia.org/wiki/Algorithme_minimax).

Vous trouverez ici (https://www.ffothello.org/informatique/algorithmes/) une analyse et des pseudo-codes vous permettant d'implémenter cet algorithme dans le cadre de l'Othello.

Pour les (beaucoup) plus rapides, vous pourrez implémenter l'élagage alpha-bêta pour accélérer les calculs sans avoir besoin de limiter la profondeur de recherche.

Ressource supplémentaire :

• Minimax et élagage alpha-bêta : https://www.youtube.com/watch?v=l-hh51ncqDl&t=29s