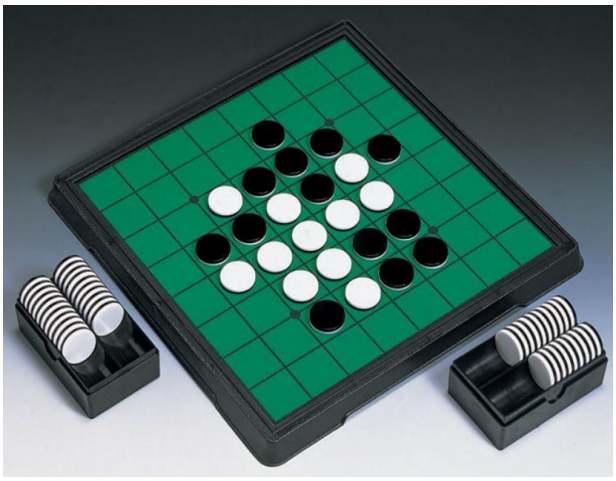
Projet Othello



M1 Mathématiques et Applications
Année universitaire :2017-2018

Rédiger par : Soufiane Fadel

1.Introduction

De quoi s'agit-il ?

Othello est un jeu de société combinatoire abstrait, qui oppose deux joueurs : Noir et Blanc. Il se joue sur un tablier unicolore de 64 cases, 8 sur 8, appelé « othellier ». Les colonnes sont numérotées de gauche à droite par les lettres a à h ; les lignes sont numérotées de haut en bas par les chiffres 1 à 8. Les joueurs disposent de 64 pions bicolores, noirs d'un côté et blancs de l'autre. En début de partie, quatre pions sont déjà placés au centre de l'othellier : deux noirs, en (e,4) et (d,5), et deux blancs, en (d,4) et (e,5).

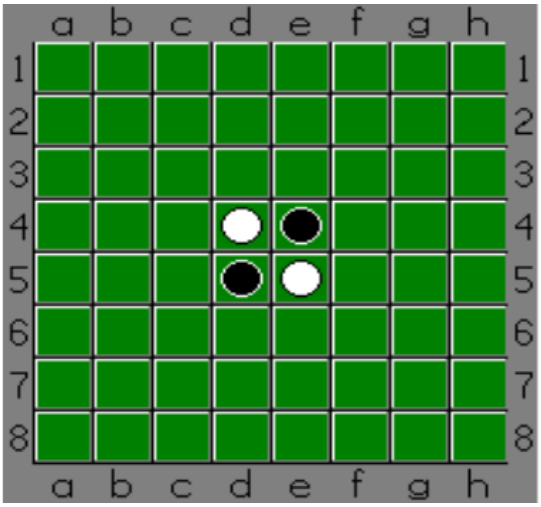
But de jeu :

Avoir plus de pions de sa couleur que l'adversaire à la fin de la partie. Celle-ci s'achève lorsque aucun des deux joueurs ne peut plus jouer de coup légal. Cela intervient généralement lorsque les 64 cases sont occupées.

Règles de jeu :

-Position de départ :

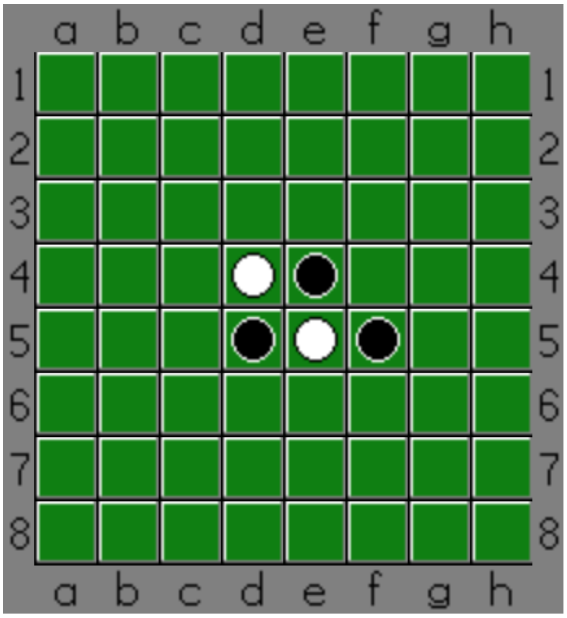
Au début de la partie, deux pions noirs sont placés en (e,4) et (d,5) et deux pions blancs sont placés en (d,4) et (e,5).

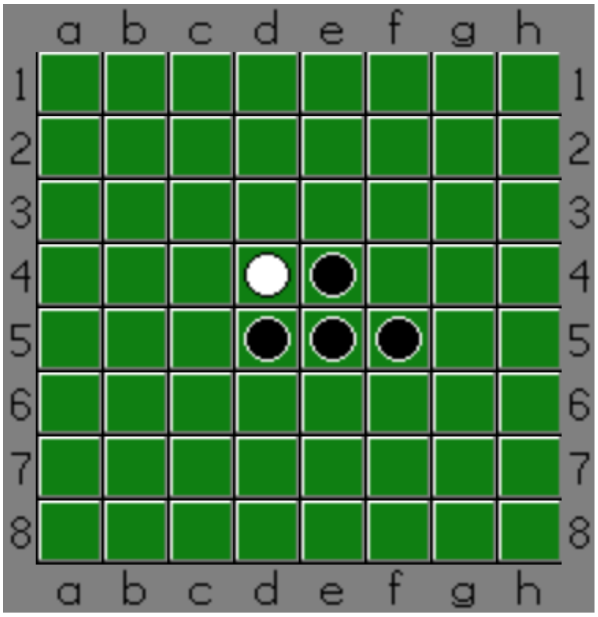


-La pose d’un pion :

A son tour de jeu, le joueur doit poser un pion de sa couleur sur une case vide de l'othellier, adjacente à un pion adverse. Il doit également, en posant son pion, encadrer un ou plusieurs pions adverses entre le pion qu'il pose et un pion à sa couleur, déjà placé sur l'othellier. Il retourne alors de sa couleur le ou les pions qu'il vient d'encadrer. Les pions ne sont ni retirés de l'othellier, ni déplacés d'une case à l'autre.

Exemple : Retournement d’un pion blanc a un pion noir.





Fin de la partie :

La partie est terminée lorsque aucun des deux joueurs ne peut plus jouer. Cela arrive généralement lorsque les 64 cases sont occupées.

2. Coté orienté objet du projet :

Dans ce rapport, nous aborderons en premier les différentes classes du programme, où nous parlerons sur les différentes méthodes de chaque classe. Dans un second temps, nous verrons les phases d'exécutions du jeu. Enfin, nous aborderons la partie intelligence artificielle du code.

2.1 Classe Othello et fonctionnalité de ses Méthodes :

Dans le fichier Othello.py, on peut y trouver la classe Othello ainsi que les fonctions relatives à la classe joueur.

limite(pos):

Elle prend en entrée une position et retourne Vraie c'est la position est dans le plateau sinon retourne Faux. Pour cela, on initialise la position définit par x et y, puis on donne une condition sur x et y afin de tester si une position est dans le plateau ou non.

```
In [42]:    jeu.limite((0,1))
Out[42]: True
```

prochain_coups(jplateau, joueur):

Elle prend en entrées un plateau de jeu, un joueur, , et retourne la liste des coups possibles d'un joueur. Pour cela, on initialise une liste vide (coup) puis en parcourant le plateau, si la case (i,j)0<i,j<8 correspond à un pion du

joueur, on regarde dans les 8 directions si pour la case suivante :

1) On n'est pas en dehors du plateau ;

2) Elle est différente du pion du joueur ;

3) Elle est différente de la case vide ;

Ses conditions sont vérifiées automatiquement par « recherche_coup

Si ces trois conditions sont respectées, alors c'est un pion du joueur adverse. On test alors si pour la case d'après, elle est vide ou non. Si elle l'est, on ajoute ses coordonnées à la liste coup.

```
In [26]: jeu.prochain_coups(jeu.plateau,jnoir)
Out[26]: [(2, 3), (3, 2), (4, 5)]
```

recherche_coup(jplateau, joueur, pos):

Prend en entrée un plateau de jeu, un joueur et la position du coup ; et qui retourne « True » si le coup est possible, « False » sinon. Et cela se fait à l'aide de la fonction auxiliaire recherche_coup_direct décrite ci-dessous.

```
In [24]: jeu.recherche_coup(jeu.plateau, jnoir, (1,0))
Out[24]: False
```

recherche_coup_direct(jplateau, joueur, pos, direc):

Prend en entrée un plateau de jeu, un joueur, une position, et une direction et qui retourne les coups jouables à partir de la position selon les déplacements et les limites fixées.

Cette fonction détermine le coup que peut jouer un joueur, ainsi que la direction à prendre pour retourner les pions adversaires.

```
In [21]: jeu.recherche_coup_direct(jeu.plateau,jnoir, (1,0),(1,0))
Out[21]: False
```

propager_coup(jplateau, joueur, coup):

Prend en entrée un plateau de jeu, un joueur et le coup joué par le joueur,

on regarde dans les 8 directions (i,j) : -1=<i,j<2:

si les condition sont vérifier => je change la position par :

« position = (coup[0] +i, coup[1] +j) »

et tant que je croise pas mon pion j'avance

« position = (position[0] +i, position[1] +j) »

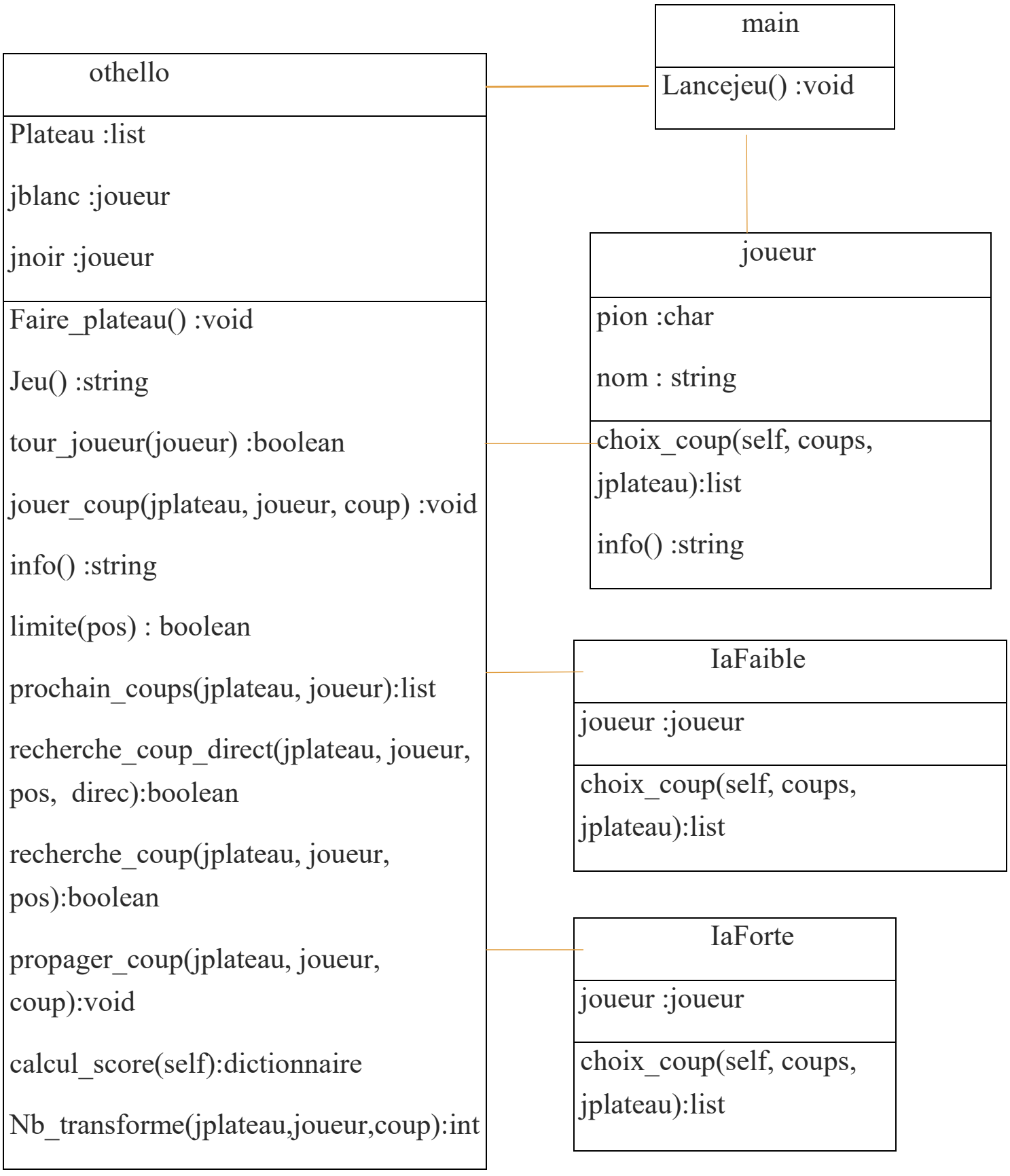
calcul_score(self):

Prend un jeu et affiche le nombre de pions de chaque joueur.

```
In [45]: jeu.calcul_score()
Out[45]: {'o': 2, 'x': 2}
```

2.1 digramme UML sous forme d'un diagramme de classes.

Afin de trouver la relation entre les classes et leurs méthodes, on propose un diagramme dit de classes qui permet une meilleur compréhension du code :


main
Lancejeu() :void
othello
Plateau :list
jblanc :joueur
jnoir :joueur
Faire_plateau() :void
Jeu() :string
tour_joueur(joueur) :boolean
jouer_coup(jplateau, joueur, coup) :void
info() :string
limite(pos) : boolean
prochain_coups(jplateau, joueur):list
recherche_coup_direct(jplateau, joueur, pos, direc):boolean
recherche_coup(jplateau, joueur, pos):boolean
propager_coup(jplateau, joueur, coup):void
calcul_score(self):dictionnaire
Nb_transforme(jplateau,joueur,coup):int
joueur
pion :char
nom : string
choix_coup(self, coups, jplateau):list
info() :string
IaFaible
joueur :joueur
choix_coup(self, coups, jplateau):list
IaForte
joueur :joueur
choix_coup(self, coups, jplateau):list

3. Intelligence Artificielle :

Dans ce projet j'ai essayé de développer une IAForte qui un peu plus intelligente que l'IAFaible qui utilise prend aléatoirement un coup parmi les coups possibles, contrairement à l'IAforte qui prend le coup qui a le plus grand nombre de point a gagné, pour cela on définit la fonction :

Nb_transforme(jplateau,joueur,coup):

Prend en entrée un plateau de jeu, un joueur et le coup joué par le joueur, et retourne le nombre de transformation possible pour un coup donné.

IAFaible VS IAForte ?!

Afin de comparer des deux IA on peut refaire l'expérience plusieurs fois et comparer la moyenne des scores finaux pour chaque IA.