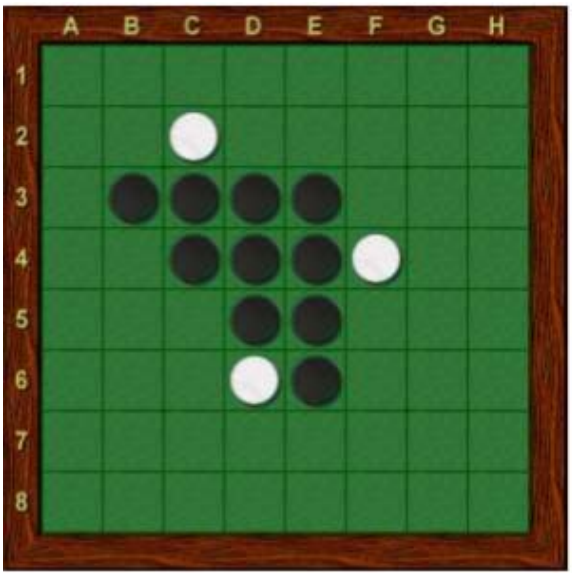
Othello : Console 1

Le jeu



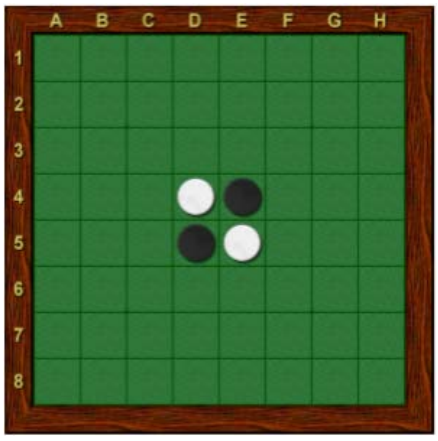
Othello est un jeu de stratégie à deux joueurs : Noir et Blanc. Il se joue sur un plateau unicolore de 64 cases, 8 sur 8, appelé othellier. Ces joueurs disposent de 64 pions bicolores, noirs d'un côté et blancs de l'autre. Un pion est « noir » si sa face noire est visible et « blanc » si sa face blanche est sur le dessus.

But du jeu

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

Position de départ

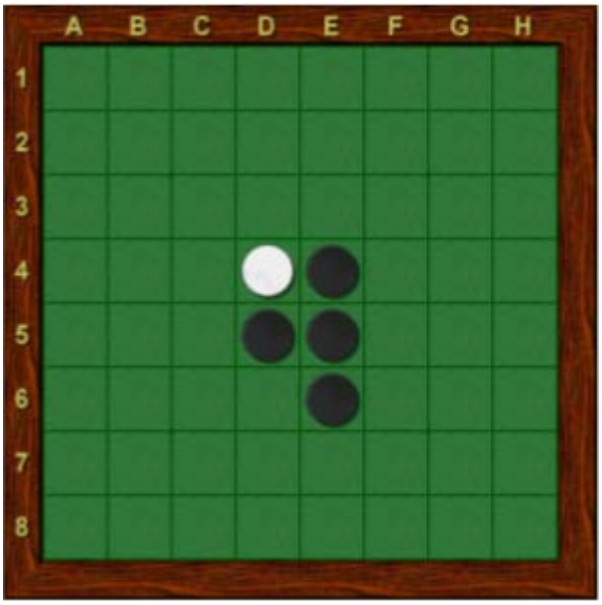
Au début de la partie, deux pions noirs sont placés en e4 et d5 et deux pions blancs en d4 et e5 (voir ci-dessous). Noir commence toujours et les deux adversaires jouent ensuite à tour de rôle.



La pose d'un pion

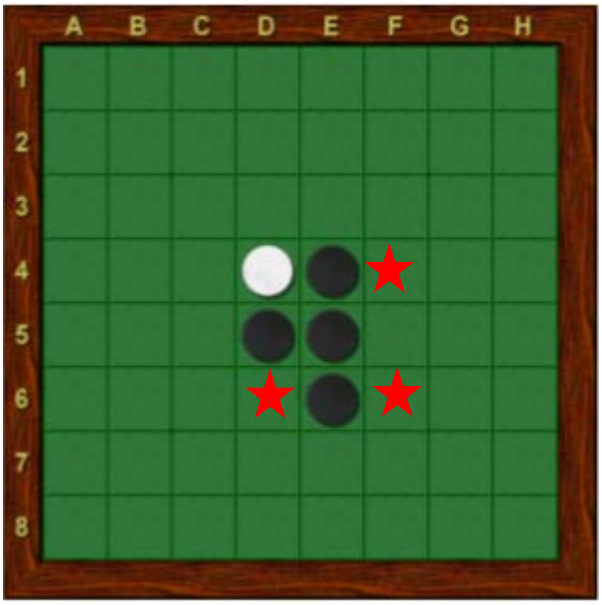
A son tour de jeu, le joueur doit poser un pion de sa couleur sur une case vide de l'othellier, adjacente à un pion adverse. Il doit également, en posant son pion, encadrer un ou plusieurs pions adverses entre le pion qu'il pose et un pion à sa couleur, déjà placé sur l'othellier. Il retourne alors de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

Le premier coup de Noir est, par exemple, en E6.



En jouant E6, il encadre le pion blanc E5 entre le pion qu'il pose et un pion noir déjà présent (ici E4) ; il retourne alors ce pion. Noir aurait aussi pu jouer en F5, C4 ou D3. Ces quatre coups de Noir sont parfaitement symétriques.

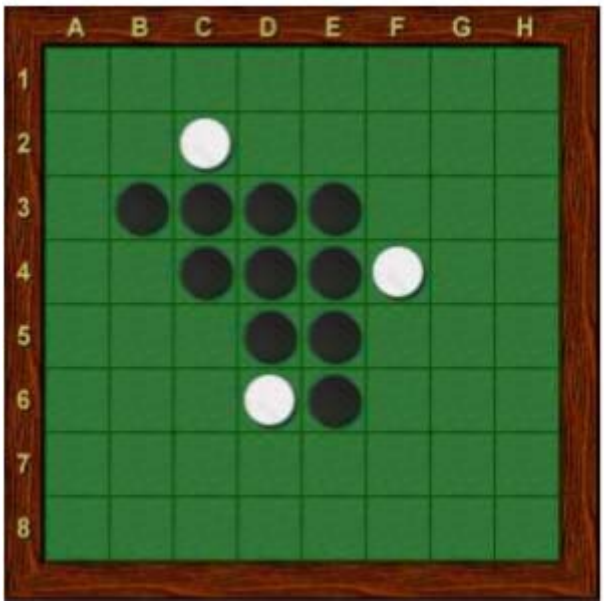
C'est maintenant à Blanc de jouer. Il a trois coups possibles :D6, F4, F6



En effet, il est obligatoire de retourner au moins un pion adverse à chaque coup.

On peut encadrer des pions adverses dans les huit directions. Par ailleurs, dans chaque direction, plusieurs pions peuvent être encadrés. On doit alors tous les retourner.

Par exemple, sur l'othelier ci-dessous, Blanc joue en D2.



Après ce coup, l'othelier devient :

Les pions noirs en D3, D4, D5 sont entourés de deux pions blancs D2, D6 : ils changent de couleurs. De même le pion E3 est entouré de deux pions blancs D2, F4 : il change de couleur.

Analyse du programme à réaliser

L'othelier

Il sera représenté en mémoire par une matrice 9 X 9

0	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	2	1	0	0	0
5	0	0	0	1	2	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Les pions noirs sont représentés par la valeur 1 et les pions blancs par la valeur 2.

Les cases vides sont à 0.

Avec cette disposition, les cases appartenant à l'othelier sont repérées par un numéro de ligne et de colonne allant de 1 à 8.

Les cases de la ligne 0 et de la colonne 0, sont numérotées de 0 à 8 et servent lors de l'affichage de l'othelier à repérer plus facilement les lignes et colonnes.

Pour entrer son choix, le joueur 1 fournira un numéro de ligne et un numéro de colonne, par exemple 65 (ligne 6 colonne 5).

Si le choix est valide, le programme n'aura plus qu'à placer le numéro du joueur dans la case (6,5) du plateau.

Tester si un choix de case est valide

Un choix est valide :

- Si la case jouée est libre : égale à 0
- Si le pion du joueur permet de retourner au moins un pion adverse, ce qui veut dire qu'il est au moins adjacent à un pion adverse.

Nous devons donc nous déplacer dans les 8 directions possibles et tester s'il y a des pions adverses à retourner.

Nous partons de la case choisie et nous nous déplaçons dans toutes les directions en appliquant aux coordonnées de la case choisie, les valeurs indiquées dans le tableau ci-dessous.

[-1,-1]	[-1,0]	[-1,1]
[0,-1]	[0,0]	[0,1]
[1,-1]	[1,0]	[1,1]

Dans une direction donnée, nous vérifions à chaque déplacement, que dans la case atteinte nous avons un pion adverse. Si oui, nous continuons le déplacement.

Si nous atteignons une case occupée par un jeton du joueur, alors nous pouvons dire que le choix est valide, puisque ce choix permet de retourner au moins un pion adverse.

Nous pourrions en rester là. Cependant, une fois le choix validé, il faudra passer à la phase de retournement dans toutes les directions des pions de l'adversaire.

Nous allons donc profiter de cette phase de test de validité, pour mémoriser dans une liste, les coordonnées des pions de l'adversaire, qui seront à retourner.

Pour ce faire, une fois que nous avons atteint le pion du joueur, nous repartons en sens opposé, en notant les cases occupées par les pions adverses, jusqu'à ce que nous revenions à la case de départ.

Si après avoir parcouru les huit directions, la liste des cases occupées par l'adversaire est vide, alors il n'y a aucun pion adverse à retourner si l'on joue cette case et le choix n'est pas valide.

Premier algorithme du programme

Nous allons dans un premier temps faire jouer l'ordinateur comme le joueur, c'est-à-dire qu'il faudra rentrer au clavier son choix.

1. Créer la matrice représentant l'othelier et y placer les premiers pions
2. Demander au joueur 1 son choix : vérifier que ce choix est valide
3. Jouer le coup du joueur 1
4. Demander au joueur 2 son choix : vérifier que ce choix est valide
5. Jouer le coup du joueur 2

Nous pouvons boucler 2 ou 3 fois sur les points 2 à 5, de façon à vérifier que les fonctions écrites fonctionnent correctement.

```
import numpy as np
NB_LIGNE = 8
NB_COLONNE=8
JOUEUR=1
ORDINATEUR=2
#======================================================
#initialisation de toutes les données
plateau = creer_plateau()
for tour in range(3):
    #affiche l'état du jeu
    print(plateau)
    #demander au joueur1 son choix de case
    choix = choix_Joueur(plateau, JOUEUR)
    #jouer le coup du joueur1
    jouer_Coup(plateau, JOUEUR, choix[0], choix[1])
    #affiche l'état du jeu
    print(plateau)
    #demander au joueur2 son choix de case
    choix = choix_Joueur(plateau, ORDINATEUR)
    #jouer le coup du joueur2
    jouer_Coup(plateau, ORDINATEUR, choix[0], choix[1])
print(plateau)
```

Les fonctions

Création de l'othelier :

```
#crée une matrice NB_LIGNE, NB_COLONNE remplie de zeros
#place les premiers jetons sur ce plateau
def creer_plateau():
    plateau = np.zeros((NB_LIGNE+1,NB_COLONNE+1),dtype=int)
```

```
    #On numérote les cases de la ligne 0, colonne 0
    for i in range (1,NB_LIGNE+1):
        plateau[i][0] = i
    for i in range (1,NB_COLONNE+1):
        plateau[0][i] = i
    #Placement des premiers pions
    plateau[4][4] = 2
    plateau[4][5] = 1
    plateau[5][4] = 1
    plateau[5][5] = 2
    return plateau
```

Choix du joueur

```
#Permet au joueur d'entrer son choix
#Renvoie une liste contenant les coordonnées de la case choisie [ligne, colonne]
def choix_Joueur(plateau, joueur):
    CHIFFRES = '1 2 3 4 5 6 7 8'.split()
    #On boucle tant que le joueur n'a pas donné un choix valide
    while True:
        print('Joueur :',joueur)
        print('Entrez votre coup (ligne, colonne)')
        choix = input()
        #On vérifie que le joueur a entré juste 2 chiffres allant de 1 à 8
        if len(choix) == 2 and choix[0] in CHIFFRES and choix[1] in CHIFFRES:
            ligne = int(choix[0])
            colonne = int(choix[1])
            #on vérifie que la case choisie est valide
            if coup_valide(plateau, joueur, ligne, colonne) == False:
              print('"Votre choix n'est pas valide."')
```

```
            continue
        else:
            break
    else:
        print('"Votre choix n'est pas valable. Entrez une ligne (1-8) puis une colonne (1-8).'")

    return [ligne, colonne]
```

Vérifier si le choix est valide

```
#Renvoie False si le choix n'est pas valide
#Si le choix est valide, renvoie une liste des cases à retourner, si le joueur
#joue cette case
def coup_valide(plateau, joueur, ligne, colonne):
    #Nous vérifions que les coordonnées fournies, correspondent bien à une case
    #du plateau
    if plateau[ligne][colonne] != 0 or not estSurPlateau(ligne, colonne):
        return False
    #Quelle est le chiffre du joueur et celui de son adversaire
    if joueur == 1:
        adversaire = 2
    else:
        adversaire = 1
    #liste des cases à retourner si on joue cette case
    jetonsARetourner = []
    for xdirection, ydirection in [[0, 1], [1, 1], [1, 0], [1, -1], [0, -1], [-1, -1], [-1, 0], [-1, 1]]:
        #on se place sur la case choisie
        x, y = ligne, colonne
        #on passe à la case adjacente dans une direction donnée
        x += xdirection
```

```
        y += ydirection
        #on continue dans cette direction tant que l'on est toujours sur
        #le plateau, et que la case sur laquelle on est, contient un pion de
        #l'adversaire.
        while estSurPlateau(x, y) and plateau[x][y] == adversaire:
            #on passe à la case adjacente toujours dans la même direction
            x += xdirection
            y += ydirection
            #Si la case atteinte est sur le plateau et si elle contient un
            #jeton du joueur:
            if estSurPlateau(x, y) and plateau[x][y] == joueur:
                #on repart en sens inverse en notant dans la liste
                #des jetons à retourner les coordonnées [ligne, colonne] des
                #cases dont il faudra modifier le contenu.
                while True:
                    x -= xdirection
                    y -= ydirection
                    if x == ligne and y == colonne:
                        break
                    jetonsARetourner.append([x, y])
    #S'il n'y a aucun jeton à retourner, alors le choix n'est pas valide,
    #sinon on renvoie la liste
    if len(jetonsARetourner) == 0:
        return False
    return jetonsARetourner

#Renvoie True si les coordonnées fournies correspondent bien à une case
#du plateau.
def estSurPlateau(x, y):
    return x >= 1 and x <= NB_LIGNE and y >= 1 and y <= NB_COLONNE
```

Jouer la case choisie :

```
#Recherche tous les jetons à retourner si l'on joue la case dont les
#coordonnées sont fournies et modifie les valeurs sur le plateau.
def jouer_Coup(plateau, joueur, ligne, colonne):
    jetonsARetourner = coup_valide(plateau, joueur, ligne, colonne)
    #y a t'il des jetons à retourner ?
    if jetonsARetourner == False:
        return False
    #Mettre le chiffre du joueur dans toutes les cases "jeton à retourner"
    plateau[ligne][colonne] = joueur
    for x, y in jetonsARetourner:
        plateau[x][y] = joueur
    return True
```

Test :

Tester ce premier programme en entrant des cases valides, d'autres non valides.

Vérifier que les jetons à retourner sont bien retournés après chaque coup.

Améliorations

1. Permettre au joueur d'arrêter avant la fin du jeu
2. Trouver tous les coups valides pour un joueur donné, à un instant donné du jeu : cette fonction permettra de savoir si le jeu est terminé. En effet si aucun des deux joueurs n'a de coup jouable, alors on a une partie nulle.
3. Calculer le score de chaque joueur pour un plateau donné.
4. Permettre de refaire une partie

Fonctions

Trouver les coups valides

```
#renvoie une liste des coordonnées [ligne, colonne] de toutes les cases
#que le joueur peut jouer.
def trouverCoupsValides(plateau, joueur):
    coupValides = []
```

```
    for x in range(1,NB_LIGNE+1):
        for y in range(1,NB_COLONNE+1):
            if coup_valide(plateau, joueur, x, y) != False:
                coupValides.append([x, y])
    return coupValides
```

Calculer les scores :

```
#comptabilise le nombre de pions de chaque joueur et renvoie un dictionnaire
#dont les clés sont 'joueur' et 'ordinateur'
def calculer_score(plateau):
    joueur_score = 0
    ordinateur_score = 0
    for x in range(1,NB_LIGNE+1):
        for y in range(1,NB_COLONNE+1):
            if plateau[x][y] == JOUEUR:
                joueur_score += 1
            if plateau[x][y] == ORDINATEUR:
                ordinateur_score += 1
    return {'Joueur':joueur_score, 'Ordinateur':ordinateur_score}
```

Modification de la fonction d'entrée du choix du joueur

```
#Permet au joueur d'entrer son choix
#Renvoie une liste contenant les coordonnées de la case choisie [ligne, colonne]
#ou bien 'fin'
def choix_Joueur(plateau, joueur):
    CHIFFRES = '1 2 3 4 5 6 7 8'.split()
    #On boucle tant que le joueur n'a pas donné un choix valide
    while True:
        print('Joueur :',joueur)
```

```
        print('Entrez votre coup (ligne, colonne), "fin" pour quitter')
        choix = input().lower()
        if choix == 'fin':
            return choix
        #On vérifie que le joueur a entré juste 2 chiffres allant de 1 à 8
        if len(choix) == 2 and choix[0] in CHIFFRES and choix[1] in CHIFFRES:
            ligne = int(choix[0])
            colonne = int(choix[1])
            #on vérifie que la case choisie est valide
            if coup_valide(plateau, joueur, ligne, colonne) == False:
                print('"Votre choix n'est pas valide."')
                continue
            else:
                break
        else:
            print('"Votre choix n'est pas valable. Entrez une ligne (1-8) puis une colonne (1-8)."')

    return [ligne, colonne]
```

La boucle de jeu

On rajoute une fonction chargée des initialisations

```
def init() :
    global plateau, fin, tour, scores
    plateau = creer_plateau()
    fin = False
    tour = JOUEUR
    scores= {'Joueur':0, 'Ordinateur':0}
```

```
#============================================================
Init()
while True:
    #initialisation de toutes les données
    plateau = creer_plateau()
    fin = False
    tour = JOUEUR
    while not fin:
        #y a t'il encore des coups possibles
        coupsPossiblesJoueur=trouverCoupsValides(plateau,JOUEUR)
        coupsPossiblesOrdinateur=trouverCoupsValides(plateau,ORDINATEUR)
        if coupsPossiblesJoueur==[] and coupsPossiblesOrdinateur == []:
            fin = True
            break
        if tour == JOUEUR:
            #le joueur a t'il encore des cases à jouer : si oui il joue, sinon il passe son tour
            if coupsPossiblesJoueur!=[]:
                print(plateau)
                #demander au joueur son choix de case
                choix = choix_Joueur(plateau, JOUEUR)
                if choix =='fin':
                    fin = False
                    break
                else:
                    #jouer le coup du joueur
                    jouer_Coup(plateau, JOUEUR, choix[0], choix[1])
            tour = ORDINATEUR
        else:
            #l'ordinateur a t'il encore des cases à jouer
```

```
            if coupsPossiblesOrdinateur!=[]:
                print(plateau)
                #demander au joueur son choix de case
                choix = choix_Joueur(plateau, ORDINATEUR)
                if choix =='fin':
                    fin = False
                    break
                else:
                    #jouer le coup de l'ordinateur
                    jouer_Coup(plateau, ORDINATEUR, choix[0], choix[1])
            tour = JOUEUR
    #fin de partie : affichage des scores
    print(plateau)
    scores = calculer_score(plateau)
    print('Joueur %s points. Ordinateur %s points.' % (scores['Joueur'],
scores['Ordinateur']))
    if scores['Joueur'] > scores['Ordinateur']:
        print('Vous gagnez par %s points! ' % (scores['Joueur'] - scores['Ordinateur']))
    elif scores['Joueur'] < scores['Ordinateur']:
        print('Vous perdez par %s points! ' % (scores['Ordinateur'] - scores['Joueur']))
    else:
        print('Partie nulle!')
    print('Une autre partie? (oui / non)')
    if input().lower().startswith('o'):
        init()
    else:
        break
```

Test :

Tester à fond tous ces ajouts et modifications pour pouvoir réutiliser cette version sans problème.