DUMAS Guillaume
GRELAUD Jérémy

Décembre 2019 - L1

Projet N°02 – Python
Black & White

```
A B C D E F
X X X X X X 1
. O O O X X 2
. . O O O X 3
. . O O . . 4
. . O . . . 5
. . . . . . 6

Tour n° 15
 C'est au tour de O
 O  ne peut pas jouer
 Pions :  O  9
          X  9
Commandes :
     PT: passer son tour
     A: abandon
     L: liste des coups valides
Votre choix d'action :
```

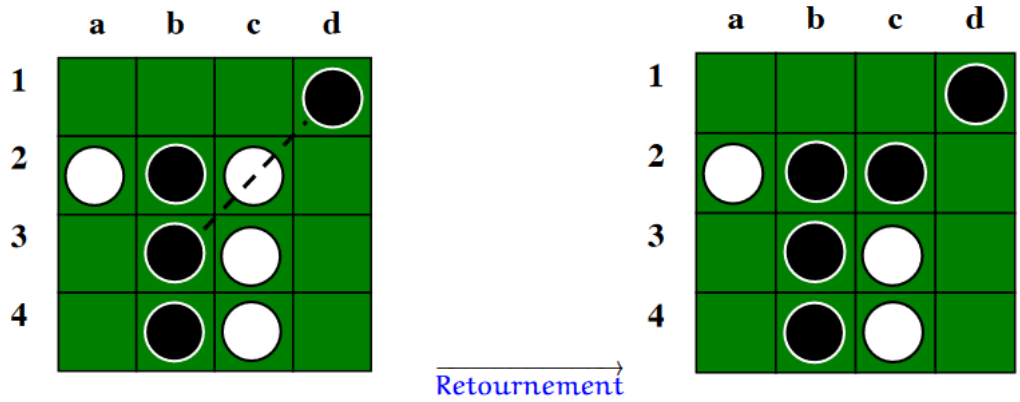
www.apprendre-en-ligne.net

SOMMAIRE

1 INTRODUCTION

Commençons par présenter le jeu du Black & White :

Ce jeu se joue à deux joueurs, l'un joue avec les pions blancs, l'autre avec les pions noirs. Le plateau est subdivisé en plusieurs cases, et est de forme carrée.
Dans un premier temps, un des joueurs va poser un pion de sa couleur. Son objectif est de « capturer » un pion adverse. Pour cela, il doit encadrer le dit-pion.
Par exemple, de cette manière :



C'est au noir de jouer, en plaçant son pion en d1 il encadre le pion blanc en c2.
Une fois encadré, le pion change de couleur. Il devient de la même couleur que les pions qui l'entourent.

Le joueur doit dans tous les cas poser un pion, même si cela n'est pas à son avantage. Cas particulier : si un joueur ne peut pas poser un pion de telle sorte à en entourer au moins un autre adverse, il passe son tour.

La partie s'arrête si un joueur n'a plus de pions de sa couleur sur le plateau, si aucun des deux joueurs ne peut jouer (ainsi, si les deux joueurs passent leur tour, la partie s'arrête par blocage mutuel), ou si le plateau est entièrement rempli.

Le vainqueur est celui qui, à la fin d'une partie, a le plus de pions de sa couleur sur le plateau.

N.B. : Il est impossible de jouer sur une case déjà remplie.

2 Présentation du projet et stratégie de réalisation

L'objectif de notre projet n°2 en Python est de coder le jeu du Black & White, en respectant le cahier des charges donné.

Nous allons détailler la démarche adoptée lors de ce projet en Python.

Dans ce projet, nous allons utiliser la console pour afficher le plateau de jeu, les différentes commandes et réaliser les saisies utilisateur.

Il nous a fallu établir une démarche de telle sorte à trouver les grandes étapes permettant la réalisation de ce jeu.

Lors de la phase de conception générale, on a déterminé les structures de données à utiliser et on a identifié les fonctions principales en les regroupant dans des fichiers. Nous avons défini une fonction pour chaque point du cahier des charges. Dans la phase de conception détaillée, on a mis en œuvre les solutions algorithmiques. De plus, nous avons fait en sorte que toutes les saisies soient sécurisées.
Nous avons développé le jeu en langage Python dans l'environnement Thonny.

Nous détaillerons dans la suite de ce rapport nos différents choix de conception.

3 Conception générale

3.1 Réponse au cahier des charges

Nous avons réalisé une fonction pour répondre à chaque objectif du cahier des charges :

Fonctionnalité du cahier des charges	**Fonction correspondante dans notre projet**
Positionner les pions en début de partie	Placement_premiers_pions
Gérer un plateau de taille variable (de 6x6 minimum)	Creation_plateau Afficher_plateau
Refuser les coups invalides ; Proposer (sur demande) la liste des coups valides	Validite_coup
Repérer le fait qu'un joueur doit passer son tour	Passer_son_tour
Repérer et indiquer tous les cas de fin de partie	Reperer_fin_partie
Proposer d'arrêter la partie à n'importe quel moment	sortie
Déterminer le gagnant	Declarer_vainqueur
Proposer une aide sur les touches à utiliser	menu
Saisir la position d'un pion à poser en indiquant ses coordonnées (vous n'êtes pas obligé d'utiliser un curseur à l'écran)	Placer_un_pion

Nous avons décidé de regrouper les différentes fonctions dans 3 fichiers :

- Le fichier **« affichage.py »** contient toutes les fonctions qui permettent de répondre aux différents problèmes liés à l'affichage, que ce soit du plateau, des pions ou des scores etc…
- Le fichier **« saisies.py »** contient toutes les fonctions qui permettent de gérer et de vérifier la saisie de l'utilisateur, que ce soit pour entrer les coordonnées d'un pion, abandonner ou bien vérifier que la saisie effectuée est bien conforme au modèle imposé, à savoir « lettre chiffre » (ex : d5)
- Le fichier **« validite.py »** contient toutes les fonctions qui permettent de gérer l'aspect de validité de la pose d'un pion : détecter si le coup saisi est valide, déterminer la liste des coups valides ou si un joueur doit passer son tour.

3.2 Choix des principales structures de données

Nous avons choisi pour l'ensemble du projet de normaliser le type de données et le nom des principales variables utilisées.
Pour une même donnée et une même utilisation, on utilise le même type et le même nom de variable.

On a choisi de représenter le plateau de jeu par la variable **plateau,** dont le type est un tableau 2D, c'est-à-dire une liste de listes en Python. Chaque liste correspond à une ligne, la première ligne correspond aux repères des colonnes A,B,C,D,etc…, et la dernière colonne correspond aux repères des lignes 1,2,3,4,etc… La taille de notre plateau est donc (taille_choisie+1 * taille_choisie+1), taille_choisie étant la taille du plateau de jeu effectif choisie par les joueurs.

Chaque case « pion » du plateau effectif contient soit ". " si la case ne contient pas de pion, soit "X " ou "O " en fonction du pion placé.
On a choisi de limiter taille_choisie à un entier entre 6 et 15.

La variable **newpion** est une chaîne de caractères qui contient les coordonnées du pion à poser saisies par l'utilisateur, sous la forme lettre + chiffre (ex : d5). La lettre correspond à la colonne ('a', 'b', 'c',…) et le chiffre à la ligne du plateau de jeu effectif (de 1 à taille_choisie).
La variable **new_pion** est une liste de 2 entiers qui contient l'équivalent des coordonnées de **newpion** en indices (i,j). L'indice i correspond au numéro de ligne et varie de 1 à taille_choisie. L'indice j correspond au numéro de colonne et varie de 1 à taille_choisie.

La variable **liste_coups_valides** est une liste de listes, qui contient la liste des coups valides que peut jouer l'utilisateur. Chaque liste représente les coordonnées d'un coup valide : c'est une liste à 2 éléments entiers (i,j) où i correspond au numéro de ligne dans le tableau 2D **plateau** (il varie donc de 1 à taille_choisie), et j correspond au numéro de colonne dans le tableau 2D **plateau** (il varie donc de 0 à taille_choisie-1) **.** Cette liste permet de vérifier si un joueur peut poser un pion ou doit passer son tour. Elle peut être affichée sur demande de l'utilisateur, à ce moment-là chaque élément sera transformé sous la forme de la variable **newpion** pour une meilleure compréhension de l'utilisateur.

La variable **joueur** est une chaîne de caractères qui permet d'identifier le joueur. Cette chaîne correspond aussi au symbole du pion du joueur : "X " ou "O "

La variable **nbr_pions** est une liste de 2 entiers contenant le nombre de pions de chaque joueur. nbr_pions[0] correspond au nombre de pions du joueur "X " et nbr_pions[1] à celui du joueur "O "

3.3 Fonctions du fichier « affichage.py »

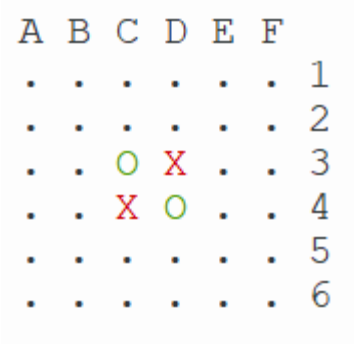
Tous les affichages se font dans la console.

Creation_plateau :

Cette fonction crée le plateau de jeu (variable **plateau**) en fonction de la taille choisie par les joueurs. La création se fait ligne par ligne : toutes les cases « pion » sont initialisées à ". "

Placement_premiers_pions :

Cette fonction détermine les coordonnées des 4 pions de départ qui sont centrés sur le plateau (on ne peut les centrer correctement que lorsque la taille du plateau est paire). Les cases correspondantes sont ensuite modifiées avec "X " ou "O " comme ci-dessous :



Afficher plateau :

Cette fonction affiche le plateau sous forme de chaine pour gérer les 3 couleurs « rouge » ou « vert » en fonction du pion et « noir » pour le reste du plateau.

Regles :

Cette fonction affiche tout simplement les règles du jeu.

Nbr_pions_2_camps :

Cette fonction parcourt le plateau afin de compter le nombre de pions de chaque joueur. Cela permet d'établir et afficher le score, et de déterminer si le plateau est rempli.

Afficher_liste_coups_valides :

A partir de la liste des coups valides obtenue grâce à la fonction liste_coups_valides, cette fonction crée une liste des coups valides à afficher, au format lettre+chiffre (ex : e4), compréhensible par l'utilisateur.

Menu :

Cette fonction affiche une interface pour l'utilisateur qui lui indique :

- Le numéro du tour
- Les scores, nombre de pions de chaque joueur
- Le joueur dont c'est le tour et la possibilité pour ce joueur de poser un pion ou non. Les joueurs sont identifiés par le symbole de leur pion ("X " ou "O ").
- Les commandes à utiliser par le joueur. Cette liste de commandes varie selon que le joueur peut poser un pion ou non

Si le joueur ne peut pas poser de pion, les commandes proposées sont :

```
Commandes :
     P: placer un pion
     A: abandon
     L: liste des coups valides
```

Si le joueur peut poser un pion, les commandes proposées sont :

```
Commandes :
     P: placer un pion
     A: abandon
     L: liste des coups valides
```

N.B. : la partie peut être abandonnée à tout moment par l'un des joueurs.

Reperer_fin_partie :

Cette fonction permet de repérer et afficher les différents cas de fin de partie :

- Plateau plein
- Blocage mutuel (la liste des coups valides est vide pour chacun des 2 joueurs)
- Victoire par élimination (un des 2 joueurs n'a plus aucun pion sur le plateau alors que l'autre en a encore)

Cette fonction est appelée à la fin de chaque tour d'un joueur.

Declarer_vainqueur :

Cette fonction est appelée si une fin de partie a été détectée et elle détermine et affiche le vainqueur en fonction du nombre de pions de chaque joueur.

3.4 Fonctions du fichier « saisies.py »

Verif_saisie :

Cette fonction vérifie si la saisie des coordonnées du pion à poser par l'utilisateur respecte bien le format demandé (lettre+chiffre). La lettre peut être en minuscule ou majuscule.

Conversion_coordonnees :

Cette fonction convertit la saisie par l'utilisateur des coordonnées (lettre+chiffre) stockées dans la variable **newpion,** en une liste à 2 entiers **new_pion**.

Placer_un_pion :

Cette fonction place le pion du joueur dans le plateau en fonction des coordonnées contenues dans la variable **new_pion.** Le contenu de la case correspondante est modifiée en "X " ou "O " en fonction du joueur.

Sortie :

Cette fonction gère l'abandon d'un joueur et demande confirmation avant de mettre fin à la partie.

3.5 Fonctions du fichier « validité.py »

La gestion de la validité des coups est l'une des principales difficultés de ce projet.

Liste_coups_valides :

Cette fonction parcourt le plateau et pour chaque case ne contenant pas de pion (". ") vérifie si le coup correspondant est valide pour le joueur dont c'est le tour. Si c'est le cas la fonction ajoute les coordonnées (i,j) de la case à la liste.

Coup_valide :

Cette fonction vérifie pour un coup donné (coordonnées du pion à placer) si ce coup permet de capturer un ou plusieurs pions adverses. Pour cela la fonction scrute le plateau à partir de la case donnée horizontalement, verticalement et en diagonale.
S'il est possible de capturer un ou plusieurs pions adverses la fonction retourne alors ces pions : les cases concernées changent de contenu ("X " -> "O " ou "O " -> "X ").

Validite_coup :

Cette fonction vérifie pour un coup donné (coordonnées du pion à placer) si ce coup est valide, c'est-à-dire :

- Qu'il se situe sur le plateau et correspond à une case « pion » (hors première ligne et dernière colonne)
- Que la case ne contient pas déjà de pion (". ")

- Que le coup permet effectivement de capturer un ou plusieurs pions adverses (appel de la fonction coup_valide)

Passer_son_tour :

Cette fonction détermine si un joueur doit passer son tour en fonction de la liste des coups valides. Si cette liste est vide le joueur doit passer son tour.

4 Conception détaillée

4.1 Fonctions du fichier « affichage.py »

Creation_plateau (taille : entier) : tableau 2D

Données copiées : **taille**, entier, la taille du plateau saisie par l'utilisateur

Sortie : **plateau**, tableau 2D, le plateau de jeu initialisé

Description de l'algo :

La fonction crée un tableau 2D d'une taille (taille+1)*(taille+1). Cela permet en effet de créer la première ligne qui correspond aux repères des colonnes A,B,C,D,etc…, et la dernière colonne qui correspond aux repères des lignes 1,2,3,4,etc…

On gère la création du plateau ligne par ligne.

Pour remplir la première ligne :
Pour j de 0 à taille-1

> Remplir chaque case avec les caractères A,B,C,D,etc … Chaque caractère est obtenu à partir de son code ASCII. Le code ASCII correspondant est (j+65) car le code ASCII des lettres majuscules débute à 65.

Fin Pour

Ensuite, pour chaque ligne restante (boucle for sur i de 0 à taille-1) on initialise les cases à ". " pour toutes les colonnes (2ème boucle for sur j de 0 à taille) sauf pour la dernière colonne (j=taille), pour laquelle la case est initialisée avec un chiffre (i+1) correspondant au repère de la ligne sur le plateau effectif.

Placement_premiers_pions (plateau : tableau 2D)

Données modifiées : **plateau**, tableau 2D, le plateau de jeu modifié avec les 4 pions de départ

Variables locales : **taille**, entier, la taille de **plateau**

Description de l'algo :

Calculer les indices dans le tableau 2D **plateau** des 2 pions X de départ : [Ent(taille/2), Ent(taille/2)] et [Ent(taille/2)+1, Ent(taille/2)-1]
Affecter "X " aux cases correspondantes.

Calculer les indices dans le tableau 2D **plateau** des 2 pions O de départ : [Ent(taille/2), Ent(taille/2)-1] et [Ent(taille/2)+1, Ent(taille/2)]
Affecter "O " aux cases correspondantes.

Afficher_plateau (plateau : tableau 2D)

Données copiées : **plateau**, tableau 2D, le plateau de jeu à afficher

Description de l'algo :

Pour chaque ligne du plateau (boucle for sur i) :

- Créer une chaîne de caractères à partir des cases de la ligne (boucle for sur j). Le contenu de chaque case est converti en chaîne de couleur verte pour les "O " rouge pour les "X ", noire sinon.
- Afficher la chaîne.

Regles ()

La fonction ne fait que des print pour afficher les règles du jeu et est appelée sur demande par l'utilisateur.

Nbr_pions_2camps (plateau : tableau 2D) : liste d'entiers

Données copiées : **plateau**, tableau 2D, le plateau de jeu

Variables locales : **compteur_nbX**, entier, le nombre de pions du joueur X
compteur_nbO, entier, le nombre de pions du joueur O

Sortie : liste de 2 entiers [**compteur_nbX, compteur_nbO**]

Description de l'algo :

Initialiser les compteurs compteur_nbX et compteur_nbO à 0.
Parcourir le plateau case par case de chaque ligne (2 boucles for imbriquées) :

- Si la case contient "X ", compteur_nbX <- compteur_nbX+1
- Si la case contient "O ", compteur_nbO <- compteur_nbO+1

Afficher_liste_coups_valides (liste_coups_valides : liste de listes) : liste de listes

Données copiées : **liste_coups_valides**, liste de listes à 2 entiers, la liste des coups valides sous forme de coordonnées [i,j] (i et j sont les indices entiers [ligne, colonne] dans le tableau 2D **plateau**)

Sortie : **liste_coups_valides_affichee**, liste de listes à 2 éléments, la liste des coups valides à afficher sous forme de coordonnées au format [lettre, chiffre] compréhensible par l'utilisateur

Description de l'algo :

Pour chaque élément [i, j] de liste_coups_valides :

- Créer une liste **liste** contenant une lettre (en minuscule) correspondant à la colonne, et un entier correspondant à la ligne. La lettre est obtenue à l'aide de son code ASCII (j+97 pour les minuscules)
- Ajouter **liste** à liste_coups_valides_affichee

Menu (passer_tour : booléen , tour : entier , joueur : chaîne de caractères , nbr_pions : liste d'entiers) : chaîne de caractères

Données copiées : **passer_tour**, booléen, True si le joueur doit passer son tour, False sinon
tour, entier, le numéro du tour de jeu (commence à 0)
joueur, chaîne de caractères , "X " ou "O ", le joueur dont c'est le tour
nbr_pions, liste de 2 entiers, la liste du nombre de pions des deux joueurs

Sortie : **choix**, chaîne de caractères, le choix de commande du joueur

Description de l'algo :

Afficher le numéro du tour : **tour** + 1
Afficher le joueur dont c'est le tour et s'il peut jouer ou non (en fonction de **passer_tour**)
Afficher le nombre de pions sur le plateau de chaque joueur.
Afficher la liste des commandes correspondant à **passer_tour**.
Saisir le choix de l'utilisateur.

Reperer_fin_partie (taille : entier, nbr_pions : liste d'entiers, liste_coupsO : liste de listes, liste_coupsX : liste de listes) : booléen

Données copiées : **taille**, entier, la taille du plateau effectif saisie par l'utilisateur
liste_coupsO, liste de listes, la liste des coups possibles du joueur O
liste_coupsX, liste de listes, la liste des coups possibles du joueur X
nbr_pions, liste de 2 entiers, la liste du nombre de pions des deux joueurs

Sortie : booléen, True si la partie est terminée, False sinon

Description de l'algo :

```
Si le nombre de pions total est égal à taille*taille
        Afficher « Partie terminée car plateau plein »
        Retourner True
Sinon si les listes liste_coupsO et liste_coupsX sont vides
        Afficher « Partie terminée par blocage mutuel »
        Retourner True
Sinon si le nombre de pions d'un joueur est nul alors que celui de l'adversaire est non nul
        Afficher « Victoire par éliminiation »
        Retourner True
Sinon
        Retourner False
```

Declarer_vainqueur (nbr_pions : liste d'entiers)

Données copiées : **nbr_pions**, liste de 2 entiers, la liste du nombre de pions des deux joueurs

Description de l'algo :

```
Si nbr_pions [0] = nbr_pions[1]
      Afficher « Il y a égalité »
Fin Si

Si nbr_pions [0] > nbr_pions[1]
      Afficher «  Le joueur X gagne »
Fin Si

Si nbr_pions [0] < nbr_pions[1]
      Afficher «  Le joueur O gagne »
Fin Si
```

Afficher les scores (nombre de pions de chaque joueur).

4.2 Fonctions du fichier « saisies.py »

Verif_saisie (newpion : chaîne de caractères) : booléen

Données copiées : **newpion**, chaîne de caractères, la saisie de l'utilisateur

Sortie : booléen, True si la saisie est correcte False sinon

Description de l'algo :

Si newpion[0] correspond à un caractère (le code ASCII correspondant doit être compris entre 65 et 90 pour une lettre majuscule, ou entre 97 et 122 pour une lettre minuscule)

    Si le reste de newpion correspond à un entier (*)

        Retourner True

    Sinon

        Retourner False

Sinon

    Retourner False

(*) Afin de s'assurer que le programme ne plante pas si l'utilisateur rentre un mot au lieu d'un entier, nous avons utilisé la structure try….except.

Le programme essaie d'abord de convertir la saisie en un entier : pour une variable a, on écrit : try a = int(a). Si a n'est pas un entier, le programme retourne un ValueError que l'on lit dans le except : a n'est pas un entier, on retourne False.

Conversion_coordonnees (newpion : chaîne de caractères) : liste d'entiers

Données copiées : **newpion**, chaîne de caractères, la saisie de l'utilisateur sous la forme lettre+chiffre

Sortie : **new_pion**, liste de 2 entiers, l'équivalent des coordonnées lettre+chiffre en coordonnées (i,j) entières

Description de l'algo :

Passer newpion[0] en majuscule. new_pion[1] (indice colonne j débutant à 1) s'obtient à partir du code ASCII de newpion[0) : code_ascii -64

Convertir le reste de newpion en entier et affecter le résultat à new_pion[0] (indice ligne i)

Placer_un_pion (plateau : tableau 2D , new_pion : liste d'entiers , joueur : chaîne de caractères)

Données copiées : **new_pion**, liste de 2 entiers, les coordonnées (i,j) entières du pion à placer
joueur, chaîne de caractères, le joueur dont c'est le tour

Données modifiées : **plateau**, tableau 2D, le plateau de jeu où il faut placer le pion

Description de l'algo :

Affecter la chaîne **joueur**, qui correspond aussi au symbole du pion du joueur, à la case correspondant à **new_pion**.

Sortie () : booléen

Sortie : booléen True si le joueur confirme son abandon False sinon

Description de l'algo :

Demander confirmation de l'abandon au joueur par la saisie d'un entier 0 (Oui) ou 1 (Non).

La saisie est sécurisée à l'aide de la structure try….except comme dans verif_saisie.
Sur lecture du ValueError dans le except, on met un message d'erreur et on empêche la sortie d'une boucle qui va se répéter tant que l'utilisateur n'a pas correctement rentré ce champ.

4.3 Fonctions du fichier « validité.py »

Coup_valide (plateau : tableau 2D , new_pion : liste d'entiers , joueur : chaîne de caractères) : booléen

Données copiées : **new_pion**, liste de 2 entiers, coordonnées (i,j) entières du coup à valider
joueur, chaîne de caractères, le joueur dont c'est le tour

Données modifiées : **plateau**, tableau 2D, si le coup permet de capturer des pions adverses on les retourne (c'est-à-dire on les change en pions du joueur qui les a capturés)

Sortie : **coup_valide**, booléen, True si le coup permet de capturer un ou plusieurs pions False sinon

Description de l'algo :

coup_valide <- False

A partir de la case correspondant à **new_pion**, dans chacune des 8 directions (horizontale gauche, horizontale droite, verticale haut, verticale bas, diagonale haut-gauche, diagonale haut-droit, diagonale bas-gauche, diagonale bas-droit) :

Si la case adjacente contient un pion du joueur adverse
    Tant que la case contient un pion du joueur adverse
        Passer à la case suivante dans la même direction
    Fin tant que
    Si la case suivante contient un pion du joueur
        Coup_valide <- True
        Affecter le symbole du pion du joueur à toutes les cases adverses parcourues

Retourner coup_valide

Validite_coup (plateau : tableau 2D , new_pion : liste d'entiers , joueur : chaîne de caractères, taille : entier) : booléen

Données copiées : **new_pion**, liste de 2 entiers, coordonnées (i,j) entières du coup à valider
joueur, chaîne de caractères, le joueur dont c'est le tour
taille, entier, la taille du plateau saisie par l'utilisateur

Sortie : booléen True si le coup est valide False sinon

Description de l'algo :

Si la case correspondant à **new_pion** est dans le plateau de jeu effectif
    Si la case contient ". "
        Retourner coup_valide(new_pion,plateau,joueur)
    Sinon
        Retourner False
Sinon
    Retourner False

Liste_coups_valides (plateau : tableau 2D , joueur : chaîne de caractères) : liste de listes

Données copiées : **joueur**, chaîne de caractères, le joueur dont c'est le tour
plateau, tableau 2D, le plateau de jeu à parcourir

Sortie : **liste_coups_valides**, liste de listes, la liste des coups valides détectés lors du parcours du plateau

Description de l'algo :

Pour chaque case du plateau de jeu effectif (2 boucles for imbriquées sur i de 1 à *taille*-1 et sur j de 0 à *taille*-2, avec *taille* correspondant à la taille de **plateau**)
    Si la case contient ". "
        A partir de cette case, dans chacune des 8 directions (horizontale gauche, horizontale droite, verticale haut, verticale bas, diagonale haut-gauche, diagonale haut-droit, diagonale bas-gauche, diagonale bas-droit) :

            Si la case adjacente contient un pion du joueur adverse
                Tant que la case contient un pion du joueur adverse
                    Passer à la case suivante dans la même direction
                Fin tant que
                Si la case suivante contient un pion du joueur
                    Ajouter la liste [i,j], correspondant aux coordonnées de la case, à liste_coups_valides

Passer_son_tour (joueur : chaîne de caractères, liste_coups : liste de listes) : booléen

Données copiées : **joueur**, chaîne de caractères, le joueur dont c'est le tour
liste_coups, liste de listes, la liste des coups possibles du joueur dont c'est le tour

Sortie : booléen True si le joueur doit passer son tour False sinon

Description de l'algo :

Si la liste **liste_coups** est vide
    Retourner True
Sinon
    Retourner False

4.4 Programme principal du jeu

Le programme principal appelle l'ensemble des fonctions définies dans les 3 fichiers décrits dans les paragraphes précédents.

Variables locales principales :

taille, entier, la taille du plateau saisie par l'utilisateur

plateau, tableau 2D, le plateau de jeu

tour, entier, le numéro du tour de jeu (commence à 0)

joueur, chaîne de caractères , "X " ou "O ", le joueur dont c'est le tour

nbr_pions, liste de 2 entiers, la liste du nombre de pions des deux joueurs

newpion, chaîne de caractères, les coordonnées du pion saisies par l'utilisateur sous la forme lettre+chiffre

new_pion, liste de 2 entiers, les coordonnées (i,j) entières d'un pion

liste_coups, liste de listes à 2 entiers, la liste des coups valides sous forme de coordonnées [i,j] (i et j sont les indices entiers [ligne, colonne] dans le tableau 2D **plateau**)

passer_tour, booléen, True si le joueur doit passer son tour, False sinon

continuer, booléen, True si la partir peut continuer False sinon

Description de l'algo :

```
taille <- saisir (saisie sécurisée entre 6 et 15)
plateau <- creation_plateau (taille)
placement_premiers_pions (plateau)
afficher_plateau (plateau)
tour <- 0
Tant que continuer = True
        Si tour est pair
                Joueur <- "O "
        Sinon
                Joueur <- "X "

        nbr_pions  <- nbr_pions_2camps (plateau)
        liste_coups <- liste_coups_valides (plateau,joueur)
        passer_tour <-passer_son_tour (joueur,liste_coups)
        choix <- menu(passer_tour,tour,joueur,nbr_pions)
```

5 Test du programme de jeu

```
Tour n° 34
 C'est au tour de X
 X  doit jouer
 Pions :  O  10
          X  25
Commandes :
     P: placer un pion
     A: abandon
     L: liste des coups valides
Votre choix d'action : l
[['f', 6]]
Appuyez sur P pour placer votre pion: p
 C'est au joueur X  de jouer. Où voulez-vous jouer ? (abandonner : A) : f6
A B C D E F
X X X X X X 1
X X O O X X 2
X X X O X X 3
X X X X X X 4
X X O O X X 5
X X X O X X 6

Partie terminée car plateau plein
Le joueur X gagne
Pions :  O  6
         X  30
```

6 Conclusion

Ce second projet fut intéressant à mener pour plusieurs raisons.

Tout d'abord, il nous a permis de confirmer nos compétences acquises lors du premier projet. En effet, nous avons pu affiner nos capacités de travail en équipe et de travail sous la pression d'une date limite proche. En effet, ce projet fut à mener en même temps que le TAI d'algèbre, que le projet en Information numérique, ainsi que le travail d'étudiant.

Nous avons aussi pu perfectionner nos techniques de programmations et apprendre davantage à gérer les fonctions, chose essentielle dans les programmes d'envergure.

Nous avons apprécié travailler sur ce projet, car il fut concret. Il s'agissait de développer un vrai jeu, avec une véritable interaction entre le jeu et l'utilisateur. Il nous a fallu pour cela faire une analyse précise de tout ce que l'utilisateur pouvait effectuer comme action et notamment comme erreur de saisie (comme par exemple rentrer « jambon » au lieu d'une taille entière pour le plateau).

Enfin, ce projet nous a permis de jouer au jeu du Black & White, que nous ne connaissions pas, pour tester nos programmes.