Activité 10 - Jeux de Plateau

Equipe Pédagogique LU1IN0*1

Consignes : Cette activité se compose d'une première partie guidée, suivie de suggestions. Il est conseillé (mais pas obligatoire) de traiter en entier la partie guidée avant de choisir une ou plusieurs suggestions à explorer.

L'objectif de cette activité est l'étude des procédures sans valeur de retour qui modifient des tableaux (ici des matrices) à travers la réalisation de différents jeux de plateau.

1 Partie Guidée : Le Tic-Tac-Toe

Le *tic-tac-toe* (ou *jeu du morpion*), est un jeu à joueur se jouant sur un plateau de taille 3×3 (une "grille"). A chaque joueur est associé un symbole (traditionnellement, une crois ou un cercle) et à tour de rôle les joueurs inscrivent leur symbole dans une case encore vide.

Le jeu s'arrête quand le même symbole apparait dans les trois cases d'une même ligne, d'une même colonne, ou d'une même diagonale. Le joueur associé au symbole gagne la partie.

Plus de renseignements peuvent être trouvés sur *Wikipedia* :
`https://fr.wikipedia.org/wiki/Tic-tac-toe`

On utilisera les alias de type suivants :

```
CaseT = str
# les elements de CaseT sont soit " " soit "O" soit "X"

PlateauT = List[List[CaseT]]
# les elements de PlateauT sont des matrices 3x3
```

ainsi `CaseT` est le type d'une case du plateau (soit `" "`, la case vide, soit `"X"`, soit `"O"`) de *tic-tac-toe* et `PlateauT` le type du plateau de jeu dans son ensemble, correspondant à une matrice où la cellule de coordonnées (i, j) du plateau `pla` est accessible par l'expression `pla[i][j]`.

Question 1. Ecrire une fonction `plateau_vide` qui ne prend rien en entrée et renvoie un plateau de jeu vide pour le *tic-tac-toe* (de taille 3×3, donc).

```
pla1 : PlateauT = plateaut_vide()
assert pla1[1][1] == " "
assert pla1[0][2] == " "
```

Question 2. Ecrire une fonction `videt` qui prend en entrée un plateau de *tic-tac-toe* et deux entier `i` et `j` entre `0` et `2` et décide si la case de coordonnées (i, j) du plateau est vide.

```
assert videt(pla1, 1, 1) == True
assert videt(pla1, 0, 2) == True
```

Question 3. Ecrire deux **procédures** `jouex` et `joueo` qui prennent en entrée un plateau `pla` et deux entiers `i` et `j` et qui inscrivent le symbole X et O (respectivement) dans la case de coordonnées (i, j).

```
assert videt(pla1, 0, 2) == True
assert jouex(pla1, 1, 1) == None
assert joueo(pla1, 0, 2) == None
assert videt(pla1, 0, 2) == False
```

Question 4. Ecrire une fonction `dessine_plateaut` qui prend en entrée un plateau de *tic-tac-toe* et qui renvoie une chaine de caractère correspondant à un affichage du plateau : la case $(0, 0)$ étant en bas à gauche.

On décrira le plateau ligne par ligne dans la chaîne résultat, en ajoutant le symbole `"\n"` apres chaque ligne.

```
assert dessine_plateaut(pla1) == 'O  \n X \n   \n'
```

Ainsi `print(dessine_plateaut(pla1))` affiche :

```
=== Evaluation de : 'print(dessine_plateaut(pla1))' ===
O
 X

============================================================
```

Pour plus de visibilité on pourra dessiner "les bords" :

```
assert dessine_plateaut(pla1) == '/---\\n|O  |\n| X |\n|   |\n\\---/'
```

Et `print(dessine_plateaut(pla1))` affiche :

```
=== Evaluation de : 'print(dessine_plateaut(pla1))' ===
/---\
|O  |
| X |
|   |
\---/
============================================================
```

Question 5. Ecrire une fonction `gagnet` qui prend en entrée un plateau `pla` de *tic-tac-toe*, un symbole `s` (soit `"X"`, soit `"O"`) et qui décide si le plateau est gagnant pour `s`, c'est-à-dire s'il existe une ligne ou une diagonale de `pla` composée uniquement de `s`.

```
assert gagnet([["O", " ", "X"], ["O", "X", " "], ["X", " ", " "]], "X") == True
assert gagnet([["O", " ", "X"], ["O", "X", " "], ["X", " ", " "]], "O") == False
assert gagnet([["X", " ", "O"], ["X", "O", " "], ["X", " ", " "]], "X") == True
assert gagnet([["X", " ", "O"], ["O", "X", " "], ["X", " ", " "]], "X") == False
assert gagnet([["O", "O", "O"], ["O", "O", "O"], ["O", "O", "O"]], "O") == True
```

Question 6. Ecrire une fonction `pleint` qui prend en entrée un plateau `pla` de *tic-tac-toe* et qui décide si le plateau est plein, c'est-à-dire si aucune case n'est `" "`.

```
assert pleint([["X", " ", "O"], ["O", "X", " "], ["X", " ", " "]]) == False
assert pleint([["O", "X", "O"], ["X", "O", "X"], ["O", "X", "O"]]) == True
```

Question 7. Ecrire une **procédure** `tourt` qui prend en entrée un plateau de jeu `pla`, deux entiers `i` et `j`, et qui joue un tour entier de *tic-tac-toe* en supposant que le joueur joue en (i, j) et en faisant jouer l'ordinateur au hasard.

Cette procédure devra, dans l'ordre :

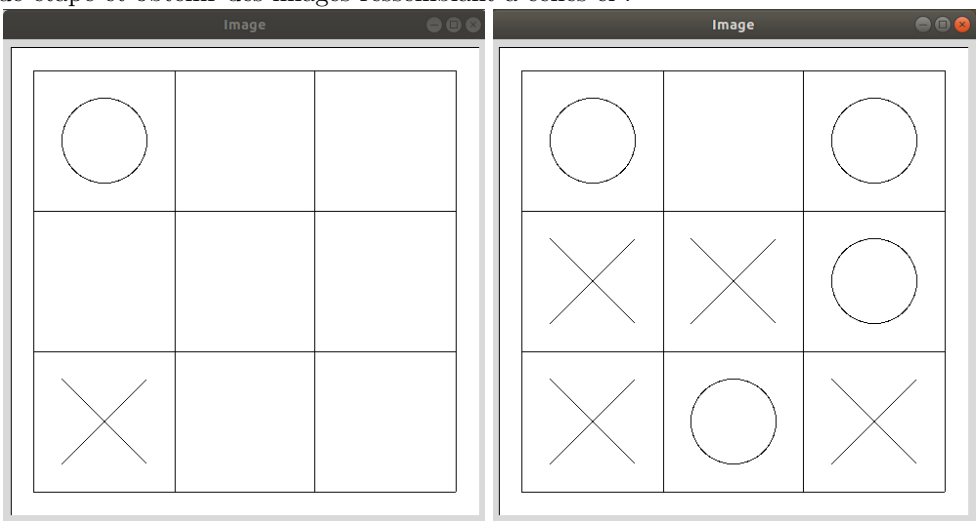
1. vérifier si le coup est légal (pas dans une case déjà occupée),
2. mettre le symbole `"X"` en (i, j) dans `pla`,
3. vérifier si le coup est gagnant,
4. vérifier si le coup est une égalité (tableau plein),
5. tirer au hasard une case vide pour l'ordinateur,
6. mettre le symbole `"O"` dans cette case,

7. vérifier si le coup est gagnant pour l'ordinateur,
8. vérifier si le coup est une égalité.

Voici un exemple de partie depuis `plat_essai`, un plateau vide :

```
=== Evaluation de : 'tourt(plat_essai, 0, 0)' ===
/---\
|   |
|  O|
|X  |
\---/
L'ordinateur joue en ( 2 , 1 ).
================================================
=== Evaluation de : 'tourt(plat_essai, 2, 2)' ===
/---\
|  X|
|  O|
|XO |
\---/
L'ordinateur joue en ( 1 , 0 ).
================================================
=== Evaluation de : 'tourt(plat_essai, 0, 2)' ===
/---\
|X X|
|  O|
|XOO|
\---/
L'ordinateur joue en ( 2 , 0 ).
================================================
=== Evaluation de : 'tourt(plat_essai, 1, 1)' ===
/---\
|X X|
| XO|
|XOO|
\---/
*** GAGNE ***
================================================
```

Question 9. (Suggestion) On pourra utiliser les primitives graphiques pour réprésenter le plateau de jeu à chaque étape et obtenir des images ressemblant à celles-ci :



2 Suggestion : Le Jeu de Dames

Le jeu de dames (*draughts* ou *checkers* en anglais) un un jeu à deux joueurs se jouant sur un damier carré de 8 cases de côté.

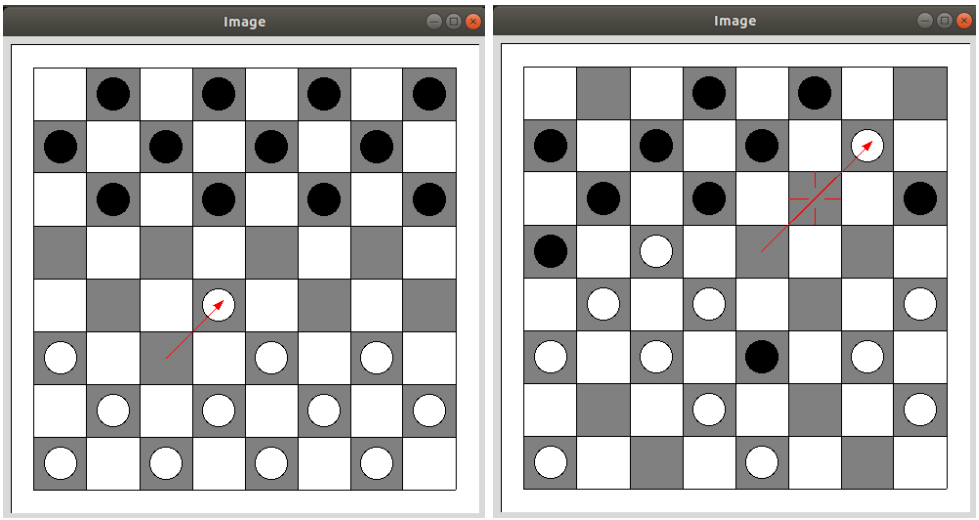
Les pions des deux joueurs (blancs et noirs) sont initialement disposés sur les cases noirs en haut et en bas du damier. Un pion ne peut se placer qu'en diagonale, dans une case vide, mais il peut "sauter" par dessus un pion adverse. Dans ce cas, le pion adverse est pris (retiré du plateau).

D'autres règles plus compliquées existent, par exemple :

- quand on prend un pion adverse, on peut continuer son mouvement si une autre "prise" est possible.
- quand un pion finit son mouvement sur la ligne opposée à son côté de départ, il se transforme en *dame*, un pion plus puissant qui se déplacer de n'importe quel nombre de cases pour prendre,
- on doit faire une prise, si au moins une prise une possible.

On les consultera sur *Wikipedia*, par exemple :
`https://fr.wikipedia.org/wiki/Dames`

On pourra faire jouer l'ordinateur, soit en tirant des coups possibles au hasard, soit en utilisant une "intelligence artificielle" (en se renseignant sur les differentes techniques applicables, comme le *min-max*), et réaliser un affichage du damier et des mouvements successifs.



3 Suggestion : 2048

2048 est un jeu monojoueur qui se joue sur un plateau de taille 4×4. Certaines cases contiennent des nombres qui sont toujours des puissances de 2. Initialement, deux cases au hasard contiennent les valeurs 2 ou 4 :

```
=== Evaluation de : 'print(dessine_plateau2048(essai2048))' ===
-------------------------
|     |     |     |     |
|     |  2  |     |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|  4  |     |     |     |
|     |     |     |     |
-------------------------
=====================================================================
```

A chaque tour, le joueur choisit une direction (haut, bas, gauche ou droite) et les nombres du plateaus sont "poussés" dans cette direction. Il s'arrêtent s'ils rencontrent le bord du plateau, ou un autre nombre, et si deux nombre identiques se recontrent à cette occasion, ils fusionnent dans leur somme. Deplus à chaque tour, apres ce processus, un 2 ou un 4 apprait dans une case vide.

Par exemple en jouant vers le bas depuis l'état précédent (l'ordinateur a fait apparaitre un 4 en $(2, 1)$):

```
=== Evaluation de : 'tour2048(essai2048, "b")' ===
-------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|     |     |  4  |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|  4  |  2  |     |     |
|     |     |     |     |
-------------------------
=====================================================
```

Puis en jouant vers la gauche (un 2 est apparu en $(3, 1)$):

```
=== Evaluation de : 'tour2048(essai2048, "g")' ===
------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
------------------------
|     |     |     |     |
|  4  |     |     |  2  |
|     |     |     |     |
------------------------
|     |     |     |     |
|  4  |  2  |     |     |
|     |     |     |     |
------------------------
==================================================
```

Puis vers le vas:

```
=== Evaluation de : 'tour2048(essai2048, "b")' ===
------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
------------------------
|     |     |     |     |
|     |     |  4  |     |
|     |     |     |     |
------------------------
|     |     |     |     |
|  8  |  2  |     |  2  |
|     |     |     |     |
------------------------
==================================================
```

Puis la gauche :

```
=== Evaluation de : 'tour2048(essai2048, "g")' ===
-------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|     |     |     |     |
|     |     |     |     |
-------------------------
|     |     |     |     |
|  4  |     |     |  2  |
|     |     |     |     |
-------------------------
|     |     |     |     |
|  8  |  4  |     |     |
|     |     |     |     |
-------------------------
==================================================
```

Le jeu est gagné quand le nombre `2048` apparaît et perdu quand l'écran est rempli et qu'aucun coup ne fait progresser le plateau.

Pour comprendre les subtilités des déplacements des nombres, on pourra essayer la version en ligne du jeu ;
`https://play2048.co/`

L'objectif est, évidemment, d'implémenter une version de *2048* jouable directement dans la zone d'évaluation de *MrPython*.

4 Suggestion : Autres jeux de plateaux

On pourra essayer d'implémenter différents jeux de plateau, plus (les échecs) ou moins (le jeu de l'oie) intéressants, de travailler sur la représentation graphique des jeux, et sur la programmation d'un "joueur ordinateur" plus ou moins intelligent.