TD no2	**Puissance 4 1ère partie**

Objectif de la séance :
Compléter le module `plateau_puissance4.py` pour se familiariser avec la programmation Python et préparer le TP d'IA qui saura bien jouer à ce jeu.

D'après Wikipédia, « Puissance 4 est un jeu de société combinatoire abstrait au tour par tour, édité pour la première fois en 1974 par MB (détenu de nos jours par Hasbro) et qui se joue à 2. Le but est de faire une ligne de 4 pions sur une grille comptant 6 rangées et 7 colonnes, les parties durant en général une dizaine de minutes. Chaque joueur dispose de 21 pions d'une couleur (jaune ou rouge). Tour à tour les joueurs posent un pion dans une colonne, le pion coulisse jusqu'à sa position la plus basse dans la colonne, et c'est à l'autre joueur de jouer. Le vainqueur est le premier joueur qui aligne quatre pions de sa couleur verticalement, horizontale ou diagonalement. »(Wikipedia)

La figure 1 montre un exemple de plateau de puissance 4.

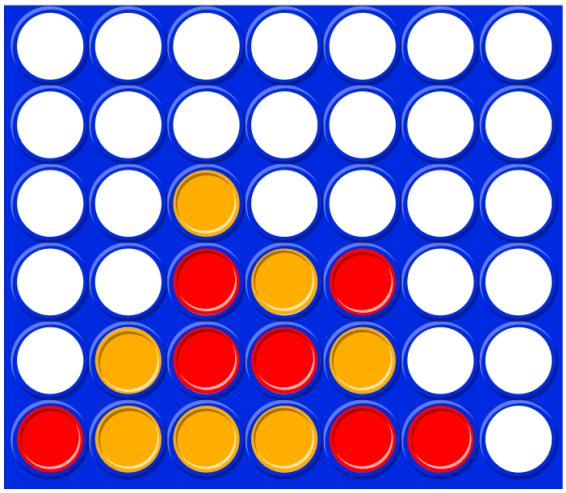

FIGURE 1 – Un plateau de puissance 4

L'archive du TP disponible sur Moodle est constituée des fichiers suivants :

- `plateau_puissance4.py` : module qui propose des fonctions permettant de manipuler un plateau de puissance 4 ;
- `puissance4.py` : module qui propose la fonction `jouer_une_partie` qui permet de jouer une partie de puissance 4 (logique métier) ;
- `ia_puissance4.py` qui propose la fonction `obtenir_meilleur_coup` qui permet à une IA de proposer un coup. Pour l'instant cette IA choisit un coup au hasard. Coder une IA plus intelligente sera l'objet d'un futur TP.
- `puissance4_ihm_txt.py`, un script, qui propose à des humains et/ou des IA de jouer l'un contre l'autre au jeu du puissance 4 en mode texte.

L'objectif de ce TP est de finir le développement du module `plateau_puissance4.py`. On décide de représenter un plateau de puissance 4 à l'aide d'une liste contenant `HAUTEUR_`

`PLATEAU` listes elles-même contenant `LARGEUR_PLATEAU` `int`. Chaque `int` représente soit une case vide (utilisation de la constante explicite `VIDE`), soit une case avec un pion jaune (utilisation de la constante explicite `JAUNE`), soit une case avec un pion rouge (utilisation de la constante explicite `ROUGE`).

Ainsi le plateau présenté par la figure 1 est représenté par la liste :

```
[[ROUGE, JAUNE, JAUNE, JAUNE, ROUGE, ROUGE, VIDE],
 [VIDE, JAUNE, ROUGE, ROUGE, JAUNE, VIDE, VIDE],
 [VIDE, VIDE, ROUGE, JAUNE, ROUGE, VIDE, VIDE],
 [VIDE, VIDE, JAUNE, VIDE, VIDE, VIDE, VIDE],
 [VIDE, VIDE, VIDE, VIDE, VIDE, VIDE, VIDE],
 [VIDE, VIDE, VIDE, VIDE, VIDE, VIDE, VIDE]]
```

Afin d'appliquer le principe d'encapsulation, ce module contient les fonctions suivantes :

- `plateau_vide` qui permet d'obtenir un plateau vide ;
- `jouer` qui permet de jouer un pion dans une colonne (`int` compris entre `1` et `LARGEUR_PLATEAU`) ;
- `contenu_case` qui permet d'obtenir le contenu d'une case du plateau (VIDE, JAUNE ou ROUGE) ;
- `hauteur_colonne` qui permet d'obtenir la hauteur d'une colonne ;
- `colonne_remplie` qui permet de savoir si une colonne est remplie (plus de case vide) ;
- `plateau_rempli` qui permet de savoir si un plateau est rempli ;
- `nb_pions_alignes_horizontalement` qui permet de connaitre le nombre de pions alignés horizontalement depuis une case donnée ;
- `nb_pions_alignes_verticalement` qui permet de connaitre le nombre de pions alignés verticalement depuis une case donnée ;
- `nb_pions_alignes_diagonalement_SONE` qui permet de connaître le nombre de pions alignés diagonalement (du Sud Ouest au Nord Est) depuis une case donnée ;
- `nb_pions_alignes_diagonalement_NOSE` qui permet de connaîtrez le nombre de pions alignés diagonalement (du Nord Ouest au Sud Est) depuis une case donnée.

Développez ces fonctions dans l'ordre suivant :

1. `contenu_case`
2. `hauteur_colonne`
3. `colonne_remplie`
4. `plateau_rempli`
5. `plateau_vide`
6. `jouer`
7. `nb_pions_alignes_horizontalement`
8. `nb_pions_alignes_verticalement`
9. `nb_pions_alignes_diagonalement_SONE`
10. `nb_pions_alignes_diagonalement_NOSE`

Pour développer ces 4 dernières fonctions, afin d'éviter les copier/coller, vous pouvez généraliser le problème en développpant une fonction privée (la bonne pratique veut dans ce cas que le nom de la fonction commence par un _) qui permet de calculer le nombre de pions alignés depuis une position vers une direction, direction représentée par deux paramètres (`pas_x` et `pas_y`) qui peuvent prendre comme valeur -1, 0 ou 1.

Utilisé en tant que script, ce module lance des tests unitaires. Une fois les fonctions précédentes convenablement codées, tous les tests unitaires devraient être `OK` :

```
$ python3 plateau_puissance4.py
  contenu_case vide OK
  contenu_case JAUNE OK
  contenu_case ROUGE OK
```

```
  hauteur_colonne vide OK
  hauteur_colonne non vide OK
  colonne_remplie non remplie OK
  colonne_remplie remplie OK
  plateau_rempli non rempli OK
  plateau_rempli rempli OK
  plateau_vide OK
  jouer OK
  nb_pions_alignes_horizontalement OK
  nb_pions_alignes_verticalement OK
  nb_pions_alignes_diagonalement_SONE OK
  nb_pions_alignes_diagonalement_NOSE OK
```

Une fois que c'est le cas, vous devriez pouvoir jouer au puissance 4 :

```
$ python puissance4_ihm_txt.py
Le joueur ayant les jaune sera un humain 0 ou un l'IA 1:
0
Le joueur ayant les rouge sera un humain 0 ou un l'IA 1:
1
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
+-+-+-+-+-+-+-+
 1 2 3 4 5 6 7
Vous avez les o, où voulez vous jouer :
3
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | |o| | | | |
+-+-+-+-+-+-+-+
 1 2 3 4 5 6 7
J'ai les x, je réfléchie
Je joue en 2
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| |x|o| | | | |
+-+-+-+-+-+-+-+
 1 2 3 4 5 6 7
Vous avez les o, où voulez vous jouer :
```