LMI 1 - Semestre 1

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO1 – Algorithmique et Programmation 1 – Groupe 4

Fiche de TP numéro 5 - Le jeu du "tic-tac-toe" – Fiche Bonus

Aujourd'hui, vous allez programmer le jeu "tic-tac-toe". Voici un exemple d'exécution du jeu :

```
TIC-TAC-TOE

 | |
-----
 | |
-----
 | |
Votre coup ("q" pour abandonner) : 1 1
 | |
-----
 |X|
-----
 | |
Coup de l'ordinateur : 0 2
 | |O
-----
 |X|
-----
 | |
Votre coup ("q" pour abandonner) : 0 0
X| |O
-----
 |X|
-----
 | |
Coup de l'ordinateur : 0 1
X|O|O
-----
 |X|
-----
 | |
Votre coup ("q" pour abandonner) : 2 2
X|O|O
-----
 |X|
-----
 | |X
Félicitations. Vous avez gagné.
Voulez-vous rejouer (oui/non) ? non
Au revoir.
```

Comme d'habitude, nous allons décomposer le problème en plusieurs sous-problèmes, où chacun correspond à une fonction différente.

N'oubliez pas de documenter et de tester vos fonctions !

1 Le plateau du jeu

La première chose dont nous avons besoin est de définir quelques constantes pour notre jeu. Les constantes ci-dessous sont utilisées pour définir le symbole représentant le joueur humain (`'X'`), le symbole représentant l'ordinateur (`'O'`) ainsi que le symbole d'une case vide (`' '`). Nous définissons aussi la taille du plateau dans une constante. Le tic-tac-toe se joue sur un plateau carré de dimension 3. Déclarez ces constantes au début de votre programme.

```
HUMAIN = 'X'       # Le symbole de l'humain.
ORDI = 'O'         # Le symbole de l'ordinateur.
VIDE = ' '         # Le symbole de case vide.
T_PLATEAU = 3      # Taille du plateau de jeu
```

Maintenant, nous allons représenter le plateau de jeu dans la mémoire de l'ordinateur. On peut représenter une ligne du plateau par une liste de trois caractères. Par exemple, dans la situation ci-dessous :

```
X| |O
-----
 |X|
-----
 | |O
```

La première ligne est représentée par la liste `['X', ' ', 'O']`, la deuxième ligne par la liste `[' ', 'X', ' ']`, et la troisième ligne par la liste `[' ', ' ', 'O']`.

Le plateau de jeu sera donc représenté par la liste de ses trois lignes, c'est-à-dire par une liste de trois listes. Dans notre exemple précédent, le plateau de jeu est représenté par :

```
[['X', ' ', 'O'], [' ', 'X', ' '], [' ', ' ', 'O']]
```

D'une manière générale, le plateau de jeu du tic-tac-toe est représenté par une liste de `T_PLATEAU` listes, chacune avec `T_PLATEAU` éléments. Chaque sous-liste correspond à une ligne et chaque élément d'une ligne est soit une case vide, soit un coup de l'utilisateur (`X`) ou bien un coup de l'ordinateur (`O`).

On peut ainsi accéder facilement à chaque case du plateau à partir de son numéro de ligne (entre 0 et `T_PLATEAU - 1`) et de son numéro de colonne (entre 0 et `T_PLATEAU - 1` également). Ainsi, si on veut accéder à la case de coordonnées $(0, 2)$ (ligne 0 et colonne 2), dans `plateau` défini comme dans l'exemple précédent :

```
>>> plateau = [['X', ' ', 'O'], [' ', 'X', ' '], [' ', ' ', 'O']]
>>> plateau[0]       # accède à la ligne numéro 0 du plateau
['X', ' ', 'O']
>>> plateau[0][2]    # accède à la case numéro 2 de la ligne 0 du plateau
'O'
```

Exercice 1 : Spécifiez et écrivez la fonction `init_plateau`, sans argument, qui créé le plateau initial du jeu où toutes les cases sont vides. La fonction doit **retourner** (pas afficher) le plateau initial du jeu :

```
>> init_plateau()
[[' ', ' ', ' '], [' ', ' ', ' '], [' ', ' ', ' ']]
```

Exercice 2 : Spécifiez et écrivez la fonction `print_plateau` qui reçoit un plateau (donc une liste de listes) comme argument et qui **affiche** (mais ne retourne rien) le plateau sous la forme usuelle pour le jeu tic-tac-toe. Par exemple :

```
>>> plateau = init_plateau()
>>> print_plateau(plateau)
 | |
-----
 | |
-----
 | |
```

Vous devez aussi tester si votre fonction affiche correctement le plateau lorsque des coups ont déjà été joués. Par exemple :

```
>> plateau = init_plateau()
>> plateau[0][1] = 'X'
>> plateau[2][2] = 'O'
>> print_plateau(plateau)
 |X|
-----
 | |
-----
 | |O
```

2 Les actions

Maintenant, vous allez gérer les actions des joueurs.

Exercice 3 : Spécifiez et écrivez la fonction `input_humain` qui reçoit un plateau (liste de listes) comme argument. La fonction doit demander le coup de l'utilisateur, c'est-à-dire les coordonnées auxquelles il veut placer son symbole. Tant que le coup saisi par l'utilisateur n'est pas valide, ou bien que l'utilisateur n'a pas décidé de quitter, la fonction doit lui redemander son choix. Le coup n'est pas valide si :

- les valeurs entrées ne sont pas deux entiers ; ou
- les valeurs entrées sont négatives ; ou
- les valeurs entrées sont à l'extérieur du plateau (`>= T_PLATEAU`) ; ou
- la position correspondante aux valeurs entrées n'est pas vide (`!= VIDE`).

La fonction doit **retourner** un couple (tuple de deux valeurs) contenant les coordonnées choisies par l'utilisateur.

Exercice 4 : Spécifiez et écrivez la fonction `coords_vides` qui reçoit un plateau (liste de listes). Cette fonction doit **retourner** une liste de couples correspondant aux coordonnées des positions vides sur le plateau.

Exercice 5 : Spécifiez et écrivez la fonction `input_ordi` qui reçoit un plateau (liste de listes). À l'aide de la fonction `coords_vides`, cette fonction doit tirer au hasard une des positions vides du plateau qui correspondra au coup de l'ordinateur. La fonction doit **retourner** un couple contenant les coordonnées du coup de l'ordinateur.

3 Qui a gagné ?

Vous allez maintenant vérifier s'il y a un gagnant dans le jeu.

Exercice 6 : Spécifiez et écrivez la fonction `est_victoire` qui reçoit un symbole (`HUMAIN` ou `ORDI`) et un plateau (liste de listes). Cette fonction doit **retourner** `True` si le joueur passé en argument a gagné la partie (c'est-à-dire, s'il a aligné `T_PLATEAU` coups sur le plateau) et `False` sinon.
Piste : Il est souvent plus facile d'implémenter cette fonction avec l'aide d'une ou plusieurs fonctions qui cherchent les coups du joueur dans différentes directions.

4 On joue maintenant

Exercice 7 : Spécifier et écrivez la fonction `joue_partie`, sans argument et sans valeur de retour. Cette fonction doit :

1. Initialiser le plateau du jeu (fonction `init_plateau`);
2. Afficher le plateau (fonction `print_plateau`);
3. Tant que la partie n'est pas terminé :
   (a) Demander le coup de l'utilisateur (fonction `input_humain`)
   (b) Placer le coup de l'utilisateur sur le plateau.
   (c) Afficher le plateau.
   (d) Vérifier si l'utilisateur a gagné (fonction `est_victoire`).
   (e) Si c'est le cas, féliciter l'utilisateur et terminer la partie.
   (f) Calculer le coup de l'ordinateur (fonction `input_ordi`).
   (g) Placer le coup de l'ordinateur sur le plateau.
   (h) Afficher le plateau.
   (i) Vérifier si l'ordinateur a gagné (fonction `est_victoire`).
   (j) Si c'est le cas, se moquer de l'utilisateur :) et terminer la partie.

Exercice 8 : Spécifiez et écrivez la fonction `input_rejouer`, sans argument. Cette fonction doit demander à l'utilisateur s'il veut jouer une nouvelle partie. La fonction doit retourner `True` si l'utilisateur a décidé de jouer une nouvelle partie et `False` sinon.

Exercice 9 : Spécifier et écrivez la fonction principale du programme (`main`) et son appel.