

Le jeu puissance 4

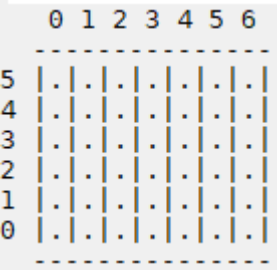
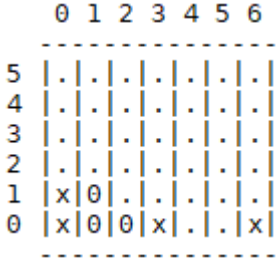
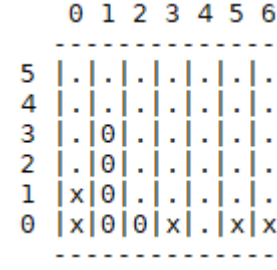
Cahier des charges

Les contraintes

- Langage pour le codage : Python 3 et supérieure
- Nombre de participants au projet : 2
- Temps disponible en semaines : 2

Les principes du jeu

Illustration par quelques images :

Représentation de la grille vide	Représentation de la grille en cours de partie	Représentation de la grille en cours de partie
		

Liste des fonctions à créer : 11 fonctions

Nom de la fonction	Difficulté estimée De 0 à 5	Codeur 1	Codeur 2
grille_vide()	0		
def affiche(gril)	1		
coup_possible(gril, col)	2		
jouer(gril, j, col)	1		
horiz (gril, j, lig, col)	3		
vert(gril, j, lig, col)	3		
diag_haut(gril, j, lig, col:	4		
diag_bas(gril, j, lig, col)	4		
victoire(gril, j)	5		
match_nul(gril)	1		
coup_aleatoire(gril,j)	3		

Spécification des fonctions

```
def grille_vide():
    """
    Fonction grille_vide():
    La fonction construit un tableau à deux dimensions de taille 6 x 7 : 6 lignes et 7 colonnes.
    Chaque case contient la valeur 0.
    La fonction ne prend pas d'argument.
    La fonction renvoie le tableau.
    """
```

```
def affiche(gril):
    """
    Fonction affiche(gril): affiche une grille de 6 lignes sur 7 colonnes.
    La fonction prend en argument un tableau de taille 6 x 7.
    Une ligne est notée lig et prend une valeur entre 0 et 5, la ligne 0
    est située en bas.
    Une colonne est notée col et prend une valeur entre 0 et 6, la
    colonne 0 est située à gauche.
    Dans la grille:
    la valeur 0 représente une case vide, représentée par un .
    la valeur 1 représente un pion du joueur 1, représenté par un x.
    la valeur 2 représente un pion du joueur 2, représenté par un o.
    """
```

	0	1	2	3	4	5	6
5
4
3
2
1
0

```
def coup_possible(gril, col):
    """
    Fonction coup_possible(gril, col) :
    Détermine s'il est possible de jouer dans la colonne col.
    Prend en argument la grille, tableau de 5x6, avec la position des pions des joueurs et un
    entier, le numéro de colonne entre 0 et 6.
    Renvoie True s'il est possible de jouer dans la colonne col, False sinon.
    Il est possible de jouer dans la colonne col, si il existe une case avec la valeur 0 dans cette
    colonne.
    """
```

```
def jouer(gril, j, col):
    """
    Fonction jouer(gril, j, col):
    Fonction qui joue un coup du joueur j dans la colonne col de la grille.
    Arguments:
    gril est la grille de 5 x 6 avec les pions des joueurs
    j est un entier qui a la valeur 1 ou 2 suivant le joueur.
    col est un entier entre 0 et 6 et désigne une colonne non pleine de la grille.
    Si j vaut 1 la première case vide de la colonne col prendra la valeur 1
    Si j vaut 2 la première case vide de la colonne col prendra la valeur 2
    """
```

```
'''
```

```
def horiz (gril, j, lig, col):
```

```
'''
```

```
Fonction horiz(gril, j, lig, col):
```

```
Détermine si il y a un alignement horizontal de 4 pions du joueur j  
à partir de la case (lig,col).
```

```
arguments: gril la grille avec les pions.
```

```
j le joueur, un entier avec la valeur 1 ou 2
```

```
lig la ligne, un entier avec la valeur entre 0 et 5
```

```
col la colonne, un entier avec la valeur entre 0 et 3
```

```
Renvoie True si c'est le cas.
```

```
'''
```

```
def vert(gril, j, lig, col):
```

```
'''
```

```
Fonction vert(gril, j, lig, col):
```

```
Détermine si il y a un alignement vertical de 4 pions du joueur j  
à partir de la case (lig,col).
```

```
arguments: grill la grille avec les pions.
```

```
j le joueur, un entier avec la valeur 1 ou 2
```

```
lig la ligne, un entier avec la valeur entre 0 et 2
```

```
col la colonne, un entier avec la valeur entre 0 et 6
```

```
Renvoie True si c'est le cas.
```

```
'''
```

```
def diag_haut(gril, j, lig, col):
```

```
'''
```

```
Fonction diag_haut(gril, j, lig, col):
```

```
Détermine si il y a un alignement diagonal vers le haut de 4 pions du joueur j à partir de la  
case (lig,col).
```

```
Arguments:
```

```
gril la grille avec les pions.
```

```
j le joueur, un entier avec la valeur 1 ou 2
```

```
lig la ligne, un entier avec la valeur entre 0 et 2
```

```
col la colonne, un entier avec la valeur entre 0 et 6
```

```
Renvoie True si c'est le cas, False sinon.
```

```
'''
```

```
def diag_bas(gril, j, lig, col):
```

```
'''
```

```
Fonction diag_bas(gril, j, lig, col):
```

```
Détermine si il y a un alignement diagonal vers le haut de 4 pions du joueur j à partir de la  
case (lig,col).
```

```
Arguments:
```

```

        gril la grille avec les pions.
        j le joueur, un entier avec la valeur 1 ou 2
        lig la ligne, un entier avec la valeur entre 0 et 2
        col la colonne, un entier avec la valeur entre 0 et 6
    Renvoie True si c'est le cas, False sinon.
'''

```

```

def victoire(gril, j):
    '''
    fonction victoire(gril, j):
        Renvoie un booléen True si le joueur j a gagné, False sinon.
        Fait appel aux fonctions horiz(), vert(), diag_haut() et diag_bas()
    '''

```

```

def match_nul(gril):
    '''
    Fonction match_nul(gril):
        Renvoie True si la partie est nulle, c'est à dire si la ligne du haut est remplie, False sinon.
    '''

```

```

def coup_aleatoire(gril,j):
    '''
    Fonction coup_aleatoire(gril,j):
        Joue un coup aléatoire pour le joueur j.
        On suppose la grille non pleine, condition indispensable pour ne pas se trouver dans une
        boucle infinie !
    '''

```

Les étapes et la répartition des tâches

Équipier 1 :

Équipier 2 :

Les étapes

1. L'analyse du travail à faire
Lecture du sujet
Réflexion sur le travail demandé :
2. On peut ici définir pour chaque fonction demandée les tests à effectuer, cela permet d'approfondir le travail demandé et d'avoir une première approche des éventuelles difficultés.
Algorithme général (d'ensemble).
3. La répartition des tâches entre coéquipiers.
Un tableau avec qui fait quoi (répartition du codage des fonctions) et estimation du temps...
Les tâches à faire en commun :
Définir :

Le lancement du jeu, l'accueil au lancement du code

Les choix proposés à l'utilisateur

Les variables (globales) à initialiser en début de code

4. Coder et tester les fonctions en respectant les spécifications *et les conditions avant et après.*

Les tests à effectuer et résultats à obtenir doivent être relevés : soit dans une documentation technique à part, soit dans le code lui-même. □ voir les assertions.

La documentation du code doit se faire durant la phase de codage : voir la docstring et les commentaires techniques.

Les tests unitaires à effectuer : test individuelle des fonctions avec relevé des valeurs testés et des résultats obtenus.

5. La mise en commun

Les réglages nécessaires....

Les tests finaux....

Illustration d'une partie jouée dans l'éditeur python choisi

```
#####
#           Puissance 4                               #
#####
La partie se joue à deux : joueur 1 et joueur 2
Le joueur 1 est celui qui commencera la partie.
```

Quel joueur êtes-vous (1 ou 2) ? 2

Bienvenue joueur 2, votre nom est (2-0) Luke.
 Votre adversaire est l'ordinateur et porte le nom (1-x) Aléa.

	0	1	2	3	4	5	6
5
4
3
2
1
0

Début de la partie dans 1 seconde !

	0	1	2	3	4	5	6
5
4
3
2
1
0	x

A vous de jouer !

Dans quelle colonne jouez-vous [0,6] ? 1

	0	1	2	3	4	5	6
5
4
3
2
1
0	x	0

	0	1	2	3	4	5	6
5
4
3
2
1	x
0	x	0

A vous de jouer !

Dans quelle colonne jouez-vous [0,6] ? 2

	0	1	2	3	4	5	6
5
4
3
2
1	x
0	x	0	0

	0	1	2	3	4	5	6
5
4
3
2
1	x
0	x	0	0	x	.	.	.

A vous de jouer !

Dans quelle colonne jouez-vous [0,6] ? 1

	0	1	2	3	4	5	6
5
4
3
2
1	x	0
0	x	0	0	x	.	.	.

	0	1	2	3	4	5	6
5
4
3
2
1	x	0
0	x	0	0	x	.	.	x

A vous de jouer !

Dans quelle colonne jouez-vous [0,6] ? 1

	0	1	2	3	4	5	6
5
4
3
2	.	0
1	x	0
0	x	0	0	x	.	.	x

	0	1	2	3	4	5	6
5
4
3
2	.	0
1	x	0
0	x	0	0	x	.	x	x

A vous de jouer !

Dans quelle colonne jouez-vous [0,6] ? 1

	0	1	2	3	4	5	6
5
4
3	.	0
2	.	0
1	x	0
0	x	0	0	x	.	x	x

#####

Bravo ! Humain (2-0) Luke vous avez gagné !

La partie est terminée !

#####