TD 2 : Introduction à Python

Projet Programmation - Geipi 2A IT

ESIREM GEIPI2 IT - Mars 2025

1 Création d'un jeu

Les questions de ce travail dirigé doivent vous permettre de mettre en oeuvre les concepts que vous avez utilisés lors du travail dirigé précédent, d'introduction au langage Python.

A travers ce TD, nous vous proposons de programmer une version minimaliste d'un puissance 4. La structure de votre programme sera basée sur des classes Python fournies et organiser de la manière suivantes :

- **Classe Joueur:** Mémorisation des statistiques de victoire du joueur, de son identifiant
- **Classe Board:** Gestion de la grille de jeu et son affichage
- **Classe Puissance4** : Gestion des règles du jeu et de la partie

L'objet Puissance 4 est l'élément central de ce programme, il régit le fonctionnement global du programme à travers une gestion des objets Joueurs et Board. Afin de vous faire gagner du temps dans l'implémentation du jeu, nous vous fournissons des script python pré-remplis.

Listing 1: Classe Board et Joueur

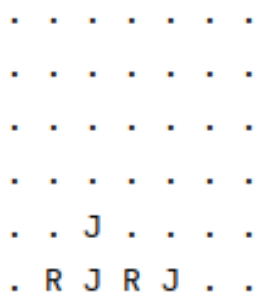
```
class Joueur:
    ...

class Board:
    def __init__(self, size):
        self.size = size
        self.board = None

    def clear(self):
        ...
    def fullColumn(self, idx):
        if self.board[0][idx] != '.':
            return True
        else:
            return False
    def __repr__(self):
        return ...
```

1. Écrivez une classe joueur avec éléments suivant:
   (a) : Un attribut *Nom* et *Nombre de victoire*
   (b) : Une initialisation des attributs lors de la création de l'objet

2. Compléter la classe Board tel que:
   (a) : La méthode **clear** réinitialise l'attribut *board*, un tableau 2D de caractère, avec le caractère par défaut '.' symbolisant une case libre.
   (b) : La méthode **__repr__** permet d'obtenir l'affichage de l'objet Board.



Listing 2: Classe Puissance 4

```
class Puissance4::
     def __init__(self):
        self.joueurA = ...
        self.joueurB= ...
        self.board = ...

    def startGame(self):
        while(True):
            self.board.clear()
            self.game()
            if(input("Tappez N pour rejouer") != "N"):
                break

    #Entree clavier d'un entier entre 0 - 6
    def inputValidValue(self):
        while True:
            value = input('Value between 0 and 6:\n')
            # Si le caractere n'est pas un entier lever une exception
            try:
                value = int(value)
            except ValueError:
                print('Not integer value')
                continue
            if 0 <= value < 7 and not self.board.fullColumn(value):
                break
            else:
                print("Invalid value")
        return value

    # Gestion du jeu
    def game(self):
        gameOver = False
        joueur = self.joueurA # Le JoueurA debute la partie
        while(not gameOver):
            # Index de la colonne entree par le joueur
            idx = self.inputValidValue()
            # Verification si 4 jetons sont alignes
                # Si oui le joueur a gagne
                    # Mise a jour du compteur de victoire
                    # Fin de partie
            # Sinon verifier si la grille est complete
                # Si oui Fin de partie
            # Sinon changement de joueur

    # Met un Jeton au nom de joueur sur
```

```
        #   la ligne la plus basse la colonne Idx
        # Retourne l'index de la ligne
        def setTokenBoard(self, idx, name):
            ...
            return row

        # Retourne le maximum de jeton aligne parmi une direction
        # Position est un tuple tel que (Ligne, Col)

        def numberTokenDir(self, position, dir,name):
            count = 1
            # Sens positif
            row = position[0] + dir[0]
            col = position[1] + dir[1]
            while row < self.board.size[1] and col < self.board.size[0]
            and row >= 0 and col >= 0 and self.board[(row,col)] == name:
                count = count + 1
                row = row + dir[0]
                col = col + dir[1]
            # Sens inverse
            row = position[0] - dir[0]
            col = position[1] - dir[1]
            while row >= 0 and col >= 0 and row < self.board.size[1]
            and col < self.board.size[0] and self.board[(row, col)] == name:
                count = count + 1
                row = row - dir[0]
                col = col - dir[1]
            return count

        # Retourne le maximum de jeton aligne
        def alignToken(self, position, name):
            numToken = self.numberTokenDir(position, (1,0), name)
            ...
            return numToken
```

3 Compléter la classe Puissance 4 telle que:

(a) : La méthode **d'initialisation** instancie des objets Joueurs et Board.

(b) : La méthode d'ajout de jeton (**setTokenBoard**) en respectant les commentaires du code.

(c) : La méthode **alignToken** retourne le maximum de jeton aligné parmi les 4 directions

(d) : La méthode de gestion de partie (**game**) en respectant les commentaires du code.

4. Tester le bon fonctionnement de votre *Puissance 4* avec une instanciation d'un objet **Puissance4** et l'appel de la méthode **startGame**.

2 Librairie PyGame

PyGame est une librairie spécifique pour la création d'interface pour des jeux vidéo en langage Python. Cette librairie inclut de multiples tutoriels pour la gestion des événements claviers/souris et la création de forme. Au cours de cette section d'introduction à la librairie PyGame, vous serez amené à créer une interface graphique à votre puissance 4. N'oubliez pas d'importer la librairie *pygame* si cela n'est pas encore réalisé.

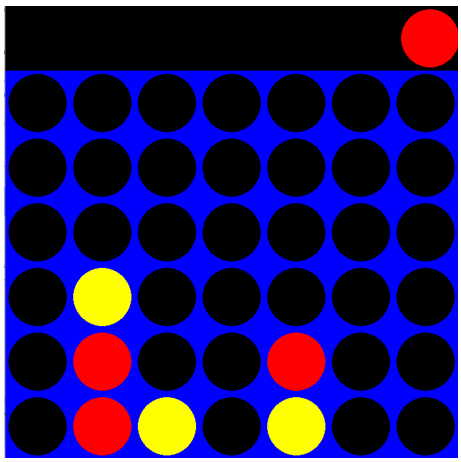
La première étape dans la construction d'une interface graphique fonctionnelle est la génération d'une fenêtre d'affichage relativement grande pour contenir notre grille de puissance 4. Pour cela, vous créerez une méthode au sein de la classe, où vous définirez des dimensions de fenêtre en pixel et vous initialiserez l'interface grâce à une méthode de **Puissance 4** nommée createWindow().

Listing 3: Initialisation d'une fenêtre PyGame

```
import pygame
pygame.init()
pixelSize = (WIDTH, HEIGHT)
self.window = pygame.display.set_mode(pixelSize)
```

5. Modifier la classe Puissance 4 telle que:
   (a) : La méthode **d'initialisation** definisse des dimensions de la fenêtre en pixel (Prenez par exemple 100 pixels de largeur et de hauteur par jeton dans la grille) .
   (b) : Créer la méthode (**createWindow**) de création de fenêtre avec l'aide du snippet code précédant.

Vous venez d'initialiser une fenêtre graphique. Cependant, la fenêtre est vide et doit donc intégrée des éléments graphiques élémentaires à votre jeu telle qu'une grille et des jetons. Une solution consiste à créer un rectangle bleu symbolisant la grille et d'ajouter sur ce rectangle noir des cercles de couleur symbolisant la présence de jeton ou non sur la grille. La figure ci-dessous est une représentation graphique de la solution graphique.



6. La librairie PyGame inclut une myriade de formes différentes que vous pourrez utiliser au cours de votre projet. En utilisant l'aide ci-dessous, réalisez une méthode de la classe **Puisssance 4** nommée **drawBoard** permettant d'obtenir une fenêtre graphique équivalente en fonction de l'état de la grille du jeu

Listing 4: Initialisation d'une fenêtre PyGame

```
# Dessiner un rectangle
pygame.draw.rect(SCREEN, Color, (Position X, Position Y, Largeur, Longueur))
# Dessiner un cercle
pygame.draw.circle(SCREEN, Color, (Centre X, CenterY)), rayon)
# Oubliez pas de rafraichir la fenetre pour afficher les modifications
pygame.display.update()
```

7. Maintenant que l'interface graphique est terminée, il ne reste plus qu'à gérer les événements utilisateurs (Clavier, souris, ...) pour ajouter un jeton à la grille et afficher le score. Pour cela, compléter de nouveau le snippet de la méthode **game** remplaçant les commandes consoles par la souris.

Listing 5: Modification de la méthode game

```
def game(self):
    gameOver = False
    joueur = self.joueurA # Le JoueurA debute la partie
    while(not gameOver):
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    sys.exit()
                # Si la souris est en mouvement
                if event.type == pygame.MOUSEMOTION:
                    # Affichage du rectangle noir superieur
                    pygame.draw.rect(...)
                    # Position de la souris en X
                    posx = event.pos[0]
                    # Afficher un jeton a la position de la souris
                    pygame.draw.circle(...)
                # Si le bouton de la souris est presse
                    # Position de la souris en X
                    posx = event.pos[0]
                    # Calcul de l'index de la colonne en fonction la position
                    # de la souris
                    idx = ...
                    # Verification si 4 jetons sont alignes
                        # Si oui le joueur a gagne
                            # Mise a jour du compteur de victoire
                            # Fin de partie
                    # Sinon verifier si la grille est complete
                        # Si oui Fin de partie
                    # Sinon changement de joueur
        self.drawBoard()
        pygame.display.update()
```