Présentation de notre Projet TicTacToe.

## Tic Tac Toe?

Le morpion (Tic Tac Toe) est un jeu de réflexion se pratiquant à deux joueurs au tour par tour et dont le but est de créer en premier un alignement de 3 symboles identiques sur une grille de 9 cases.

# A quoi sert ce code?

Ce code est un jeu de morpion (ou tic-tac-toe) en Python, permettant de jouer soit entre deux joueurs, soit contre une intelligence artificielle (IA).

# Etape

## Explication détaillée de chaque fonction :

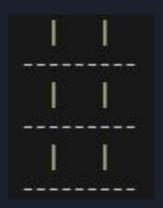
```
def afficher_plateau(plateau):
    for ligne in plateau:  # Affiche chaque ligne du plateau
        print(" | ".join(ligne)) # Joint les éléments de la ligne avec des "|"
        print("-" * 9) # Affiche une ligne de séparation
```

1. afficher\_plateau(plateau)

But : Afficher le plateau de jeu. Fonctionnement :

- La fonction prend en entrée une liste plateau, qui est une matrice 3x3 représentant le jeu.
- Pour chaque ligne du plateau, elle imprime les éléments de la ligne, séparés par un " | ".
- Ensuite, elle imprime une ligne de tirets pour séparer les lignes visuellement.

Tableau sur le Terminal



#### 2. verifier\_victoire(plateau)

But : Vérifier si un joueur a gagné.

Fonctionnement:

- La fonction vérifie toutes les lignes, colonnes et deux diagonales pour voir si un joueur (X ou
   O) a aligné trois de ses symboles.
- Elle retourne True si un joueur a gagné, sinon False.

#### 3. plateau\_rempli(plateau)

But: Vérifier si toutes les cases du plateau sont remplies.

#### Fonctionnement:

- La fonction parcourt chaque cellule du plateau.
- Si elle trouve une cellule vide (représentée par un espace ' '), elle retourne False.
- Si toutes les cellules sont remplies, elle retourne True.

```
det ia(plateau, signe):
   opponent = "X" if signe == "0" else "0" # Une IA très simple pour jouer contre le joueur
   for i in range(3):
       for j in range(3):
           if plateau[i][j] == ' ':
               plateau[i][j] = signe
               if verifier victoire(plateau):
                   return i * 3 + j
                                      # Retourne le mouvement en 1D
               plateau[i][j] = ' '
                                         # Annule le mouvement
   for i in range(3):
                                              # Si pas de victoire, joue au premier endroit disponible
       for j in range(3):
           if plateau[i][j] == ' ':
               return i * 3 + j
                                              # Aucun mouvement possible
   return -1
```

#### 4. ia(plateau, signe)

But : Logique de l'IA pour jouer.

#### Fonctionnement:

- La fonction prend le plateau et le symbole du joueur courant (X ou O).
- Elle vérifie si le joueur peut gagner en jouant dans une case vide. Si c'est le cas, elle retourne l'index de cette case.
- Si aucune victoire n'est trouvée, elle joue dans la première case vide qu'elle trouve.
- Si le plateau est plein, elle retourne -1.

#### 5. play\_game()

But : Gérer la logique du jeu.

#### Fonctionnement:

 Le code initialise un plateau vide et définit le joueur courant comme X. Il demande à l'utilisateur de choisir entre Joueur contre Joueur ou Joueur contre IA. Dans une boucle continue, il affiche le plateau et gère les entrées des joueurs. Après chaque coup, il vérifie si un joueur a gagné ou si le plateau est plein, puis il change de joueur.

```
def play_game():
   plateau = [[' ']*3 for _ in range(3)]
   current player = "X"
   mode = input("Choisissez le mode (1: JvsJ, 2: JvsIA): ")
   while True:
       afficher plateau(plateau)
       if mode == "1":
                                                                                                       # Mode Joueur contre Joueur
           ligne, col = map(int, input(f"Joueur {current_player}, choisissez (ligne col): ").split())
           if plateau[ligne][col] == ' ':
               plateau[ligne][col] = current_player
                print("Case déjà prise, réessayez.")
                                                                                                       # Mode Joueur contre IA
            if current player == "X":
                move = int(input(f"Joueur {current player}, choisissez (0-8): "))
                if move >= 0 and move < 9:
                   ligne, col = move // 3, move % 3
                    if plateau[ligne][col] == ' ':
                        plateau[ligne][col] = current_player
                        print("Case déjà prise, réessayez.")
                move = ia(plateau, current player)
                if move != -1:
                   ligne, col = move // 3, move % 3
                   plateau[ligne][col] = current_player
                   print(f"L'IA a joué en {move}.")
       if verifier victoire(plateau):
            afficher_plateau(plateau)
           print(f"Félicitations, joueur {current player} ! Vous avez gagné !")
           break
        if plateau rempli(plateau):
           afficher_plateau(plateau)
           print("Match nul !")
           break
       current_player = "0" if current_player == "X" else "X"
```

```
if __name__ == "__main__": # Démarre le jeu
   play_game()
```

#### 6. Point d'entrée du programme

But : Lancer le jeu lorsque le script est exécuté.

Fonctionnement : Si le script est exécuté directement (et non importé), il appelle la fonction play\_game() pour démarrer le jeu.

### **Exécution dans le Terminal**

```
Choisissez le mode (1: JvsJ, 2: JvsIA):
```

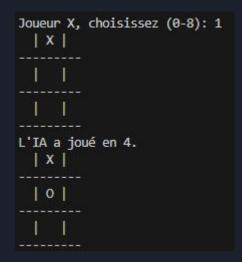
1. Il nous pose la question dans quel mode souhaitant jouer?

Si on choisit le premier mode Joueur vs Joueur, il nous posera la question :

Mais si on choisit le mode Joueur vs IA, il nous posera la question :

```
Joueur X, choisissez (0-8): 1
| X |
------
| |
------
| |
-----
L'IA a joué en 4.
| X |
------
| 0 |
```

(IA) Après que le joueur a choisi sa casse, l'IA choisira automatiquement la sienne et la partie continue.



(IA) Après que le joueur a choisi sa casse, l'IA choisira automatiquement la sienne et la partie continue.

(JvsJ) Après que le premier joueur a choisi sa casse, le tour passe au 2ème joueur.

