

Partie 1 : La Base de Données SQLite



Nous allons créer une classe pour gérer toutes les interactions avec la base de données. Elle contiendra deux tables : Joueurs (pour les noms d'utilisateurs) et Scores (pour l'historique des parties).

Fichier : `db_manager.py`

Complétez les méthodes de la classe `DBManager` :

```
import sqlite3

class DBManager:
    """Gère la connexion à la base de données et les opérations CRUD."""

    def __init__(self, db_name="morpion.db"):
        self.db_name = db_name
        self.conn = None
        self.cursor = None
        self.connect()
        self.creer_tables()

    def connect(self):
        """Établit la connexion à la base de données."""
        # Établir la connexion

    def creer_tables(self):
        """Crée les tables Joueurs et Scores si elles n'existent pas."""
        try:
            # Créer la table Joueurs (id INTEGER PRIMARY KEY, nom TEXT UNI

            # Créer la table Scores (id INTEGER PRIMARY KEY, joueur_id INT
```

```

def get_or_create_joueur(self, nom_joueur):
    """Récupère l'ID d'un joueur ou le crée s'il n'existe pas."""
    # Chercher Le joueur
    # Insérer Le nouveau joueur
def enregistrer_score(self, joueur_id, resultat):
    """Enregistre le résultat d'une partie pour un joueur (e.g., 'Vict
    # Insérer Le score
def get_classement(self):
    """Affiche le classement (nombre de victoires par joueur)."""
    # Requête SQL pour compter Les 'Victoire' et joindre avec Joueurs

def close(self):
    """Ferme la connexion à la base de données."""

if __name__ == '__main__':
    # Testez votre classe DBManager
    db = DBManager()
    joueur1_id = db.get_or_create_joueur("Alice")
    joueur2_id = db.get_or_create_joueur("Bob")

    db.enregistrer_score(joueur1_id, "Victoire")
    db.enregistrer_score(joueur1_id, "Nul")
    db.enregistrer_score(joueur2_id, "Victoire")
    db.enregistrer_score(joueur2_id, "Victoire")

    classement = db.get_classement()
    print("\n🏆 Classement des Victoires :")
    for nom, victoires in classement:
        print(f"- {nom} : {victoires} victoires")

    db.close()

```

Partie 2 : La Logique du Jeu (POO)

Nous allons créer une classe JeuMorpion pour gérer l'état du plateau et les règles.

Fichier : morpion.py

Complétez les méthodes de la classe JeuMorpion :

```

class JeuMorpion:
    """Représente le jeu du Morpion."""

```

```

def __init__(self):
    # Initialiser le plateau (liste de 9 éléments)
    self.plateau = ...
    self.joueur_actuel = ...
    self.partie_finie = False

def afficher_plateau(self):
    """Affiche le plateau de jeu formaté."""

def faire_coup(self, position):
    """Joue un coup à la position donnée (0 à 8)."""
def verifier_victoire(self):
    """Vérifie si le joueur actuel a gagné."""

def verifier_nul(self):
    """Vérifie si le plateau est plein sans victoire."""

def reset(self):
    """Réinitialise le plateau pour une nouvelle partie."""

```

Partie 3 : L'Exécution du Jeu et l'Intégration SQLite

Nous allons maintenant utiliser les classes DBManager et JeuMorpion pour créer le script principal du jeu.

Fichier : main_morpion.py

```

from morpion import JeuMorpion
from db_manager import DBManager

def boucle_jeu(db, joueur_x_id, joueur_o_id, nom_x, nom_o):
    """Contient la boucle principale d'une partie de Morpion."""

    #Demander la position (de 1 à 9) et la convertir en indice (0 à 8)
    # Fin de partie : Enregistrement et affichage
    # Gérer l'enregistrement du score et l'affichage du résultat
        # Le perdant n'enregistre rien, ou on pourrait enregistrer une déf

        # On pourrait enregistrer un nul pour les deux joueurs, mais pour

```

```
def main():
    """Fonction principale du programme."""
    db = DBManager()

    #Demander Le nom des deux joueurs et récupérer leurs IDs
    # Afficher Le classement après chaque partie

if __name__ == "__main__":
    # Lancement du jeu
    main()
```