Python OOP Tic-Tac-Toe

By: Harsimran Singh (2401030148), Arayan Sharma (2401030159)

---

game.py

This module contains the core game logic, including the `Board`, `Player`, and `Game` classes.

```
class Board:
   def __init__(self, size=3):
       self.size = size
       self.board = [[' ' for _ in range(size)] for _ in range(size)]

   def display(self):
       for row in self.board:
           print('|'.join(row))
           print('-' * (self.size * 2 - 1))

   def update(self, row, col, player):
       if self.is_valid_move(row, col):
           self.board[row][col] = player.symbol
           return True
       return False

   def is_valid_move(self, row, col):
       return 0 <= row < self.size and 0 <= col < self.size and self.board[row][col] == ' '

   def is_full(self):
       for row in self.board:
           if ' ' in row:
               return False
       return True


class Player:
   def __init__(self, symbol):
       self.symbol = symbol


class Game:
   def __init__(self, size=3):
       self.board = Board(size)
       self.player1 = Player('X')
       self.player2 = Player('O')
       self.current_player = self.player1

   def start_game(self):
       while True:
           self.board.display()
           row, col = self.get_player_move()
           if self.board.update(row, col, self.current_player):
               if self.check_win():
                   self.board.display()
                   print(f"Player {self.current_player.symbol} wins!")
                   break
               elif self.board.is_full():
                   self.board.display()
                   print("It's a draw!")
                   break
               else:
                   self.switch_player()
           else:
               print("Invalid move. Try again.")


   def switch_player(self):
       self.current_player = self.player2 if self.current_player == self.player1 else self.player1
```

```
    def check_win(self):
        # Check rows
        for row in self.board.board:
            if all(cell == self.current_player.symbol for cell in row):
                return True

        # Check columns
        for col in range(self.board.size):
            if all(self.board.board[row][col] == self.current_player.symbol for row in range(self.board.size)):
                return True

        # Check diagonals
        if all(self.board.board[i][i] == self.current_player.symbol for i in range(self.board.size)):
            return True
        if all(self.board.board[i][self.board.size - 1 - i] == self.current_player.symbol for i in range(self.board.size)):
            return True

        return False

    def get_player_move(self):
        while True:
            try:
                row, col = map(int, input(f"Player {self.current_player.symbol}, enter your move (row, column) (1-indexed): ").
                row -= 1
                col -= 1
                if self.board.is_valid_move(row,col):
                    return row, col
                else:
                    print("Invalid move. Try again.")

            except ValueError:
                print("Invalid input. Please enter row and column numbers separated by a comma.")

if __name__ == "__main__":
    game = Game()
    game.start_game()
```