GUI-based Tic Tac Toe

1. Overview

This Python script implements a GUI-based **Tic Tac Toe** game using tkinter. It supports:

- Single-player mode (player vs. computer).
- Three difficulty levels (Easy, Medium, Hard).
- Score tracking (Player, Computer, Draws).
- Choice of who starts first (player or computer).
- Highlighting the winning combination.
- Game reset and score reset functionality.

2. Structure & Components

2.1 Class: TicTacToe

The core game logic is encapsulated inside the TicTacToe class, with the following responsibilities:

- **UI Setup** (setup_ui)
- **Game State Management** (board, score, highlighted)
- **Player Actions** (player_move)
- **Computer AI** (get_computer_move, find_best_move, minimax)
- **Game Flow Control** (check_game_over, end_game, reset_game)
- **Utility Functions** (score display, start order handling)

3. Main Functionalities

3.1 User Interface

- **Layout**
    - **Header Frame**:
        - Difficulty dropdown (Easy, Medium, Hard).
        - Start order buttons (Player Starts, Computer Starts).

- **Game Board**:
  - 3×3 grid of buttons.
- **Result & Score Display**:
  - result_label for win/draw messages.
  - score_label for cumulative scores.
- **Control Buttons**:
  - Play Again
  - Reset Scores

3.2 Gameplay Flow

1. **Starting Player Selection**
   - Player or computer is chosen to move first.
   - If computer starts, after(500, self.computer_move) delays AI's first move.
2. **Player Move (player_move)**
   - Updates the button to 'X'.
   - Disables the clicked button.
   - Checks for win/draw before triggering computer's move.
3. **Computer Move (computer_move)**
   - Chooses a position based on difficulty:
     - **Easy**: random move.
     - **Medium**: 50% random, 50% optimal (minimax).
     - **Hard**: optimal move only.
   - Places 'O' and disables the cell.
4. **Win/Draw Check (check_game_over)**
   - Calls is_winner for both players.
   - Updates scores.
   - Highlights winning cells.
   - Disables further input.

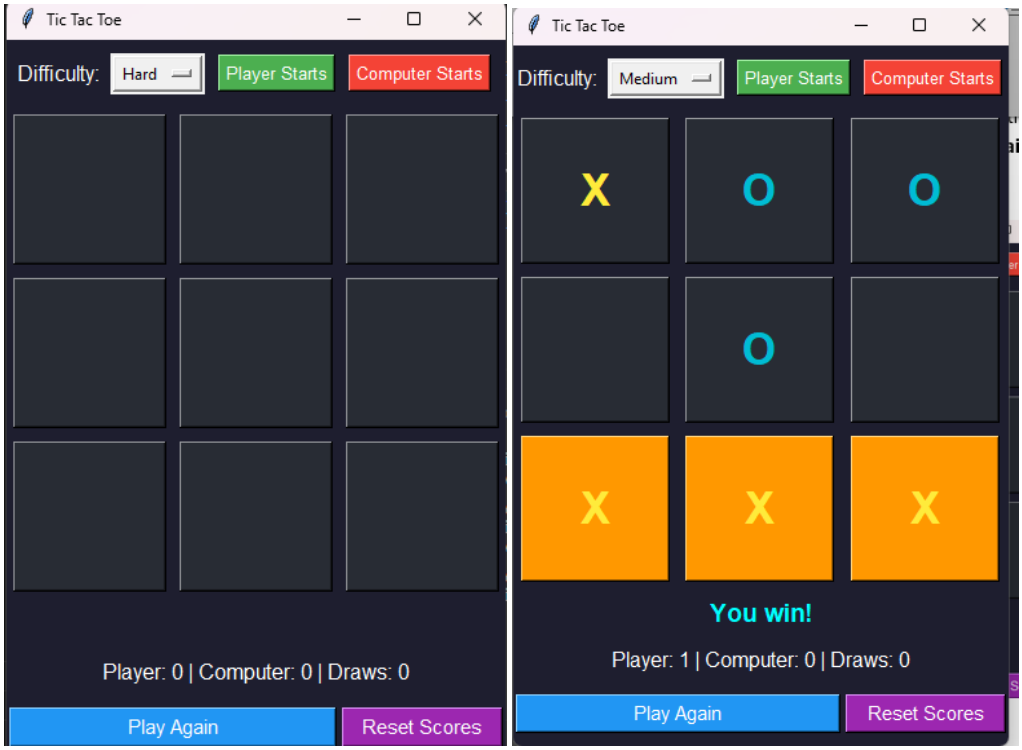
3.3 AI (Minimax Implementation)

- **Easy**: Random empty spot.
- **Medium**: 50% random, otherwise minimax.
- **Hard**: Always minimax.
- **Minimax Scoring**:
    - O wins → +1
    - X wins → -1
    - Draw → 0
- **Recursion** explores all possible moves until terminal state.

4. Strengths

Well-structured: Good use of an object-oriented approach.
User-friendly UI: Dark theme, clear layout, responsive controls.
Multiple difficulty levels: Makes the game more engaging.
Win highlighting: Visually satisfying.
Score persistence during multiple rounds.
Separate control for **"Play Again"** and **"Reset Scores".**

5.Output:



6.Conclusion

Your game is **functional, visually clear, and enjoyable**. It's an excellent example of combining **GUI programming** with **AI logic** in Python. The structure is clean enough to extend with more features (themes, multiplayer, network play, animations). A few optimizations and UI refinements could make it even more polished.