### **Overview**

The program implements a Tic-Tac-Toe game where:

* A human player plays against an AI.
* The AI uses the **Minimax algorithm** to evaluate the best possible moves, ensuring it cannot be beaten.
* A graphical interface (UI) built with tkinter allows players to interact with the game.

### **Breaking Down the Implementation**

#### **1. The TicTacToe Class**

The class encapsulates all the logic for the game, including the UI, AI, game rules, and player interactions.

##### **Attributes**

* **self.board**: A 3x3 grid (2D list) representing the game board.
  + None: Empty cell.
  + "X": Player's move.
  + "O": AI's move.
* **self.current\_player**: Tracks whose turn it is ("X" for player, "O" for AI).
* **self.buttons**: A 2D list of tk.Button objects representing clickable cells in the UI.

##### **Methods**

Each method in the class handles a specific part of the game logic.

#### **2. UI Creation**

The **create\_board()** method initializes a 3x3 grid of buttons for the UI:

* Each button corresponds to a cell in self.board.
* Clicking a button triggers the player\_move() function, allowing the player to make a move.

#### **3. Player Moves**

The **player\_move()** method:

1. Ensures the clicked cell is empty.
2. Updates the board and the button with "X".
3. Checks for a win or draw condition:
   * If the player wins, the game ends with a message.
   * If the game is a draw, it ends with a draw message.
4. If the game continues, it switches the turn to the AI ("O").

#### **4. AI Moves**

The **ai\_move()** method:

1. Calculates the best possible move for the AI using the **Minimax algorithm**.
2. Updates the board and UI with the AI's move.
3. Checks for a win or draw:
   * Ends the game if the AI wins or if there's a draw.
4. If the game continues, switches the turn back to the player ("X").

#### **5. Minimax Algorithm**

The Minimax algorithm ensures the AI plays optimally. It evaluates moves recursively:

##### **Key Ideas**

* The AI assumes the opponent will also play optimally.
* Assign scores to board states:
  + **+10**: AI wins.
  + **-10**: Player wins.
  + **0**: Draw.
* The AI tries to maximize its score while minimizing the player's.

##### **How It Works**

1. **Base Case**:
   * If the game ends (win, loss, or draw), return the score based on the outcome.
2. **Recursive Case**:
   * Simulate all possible moves for the current player.
   * For each move:
     + Temporarily update the board.
     + Recursively call minimax() for the next player.
     + Undo the move (backtracking).
     + Record the best score:
       - Maximizing for the AI.
       - Minimizing for the player.

##### **Alpha-Beta Pruning (Optional)**

* This technique reduces the number of nodes evaluated by pruning branches that won't affect the outcome.
* It speeds up the decision-making process for the AI.

#### **6. Checking Game Status**

Two helper methods determine if the game is over:

* **check\_winner(player)**:
  + Checks rows, columns, and diagonals for a win.
  + Returns True if the specified player wins.
* **is\_draw()**:
  + Checks if all cells are filled and there’s no winner.
  + Returns True if the game is a draw.

#### **7. Ending the Game**

The **end\_game()** method:

* Disables all buttons to prevent further moves.
* Displays a message indicating the winner or a draw.

### **How the Program Runs**

1. **Initialization**:
   * The game initializes with an empty board and player "X"'s turn.
   * The UI is displayed using tkinter.
2. **Gameplay**:
   * Player clicks a button to make a move.
   * AI calculates its best move using Minimax and plays.
3. **Game End**:
   * If a player wins or the game draws, a message is displayed, and the game stops.

### **Why This AI is Unbeatable**

The AI uses the Minimax algorithm to evaluate all possible moves and outcomes:

* It calculates the optimal move based on future possibilities.
* It avoids mistakes and always makes the best possible decision.
* This guarantees that the AI cannot lose, though the game might end in a draw if the player also plays optimally.

### **Running the Program**

1. Save the code to a .py file (e.g., tic\_tac\_toe.py).
2. Ensure Python and tkinter are installed.
3. Run the file in a Python environment.

You’ll see a 3x3 grid. Click cells to make your move and watch the AI respond.

### **Future Enhancements**

* Add a restart button to play multiple games.
* Allow the player to choose who starts first.
* Add difficulty levels by limiting the depth of the Minimax algorithm for easier AI.