Meeting Minutes

Meeting Minutes: Tic-Tac-Toe Game Implementation

Date: [Date of Code Creation/Review - Not explicitly stated, infer from context]

Attendees: Krishna Thakkar (C075, 60004220, 250) [Assuming sole developer/author]

Subject: Implementation of a Tic-Tac-Toe game with AI using the Minimax algorithm.

Key Discussion Points:

Game Board Initialization:

The game board is represented as a list of 9 elements, initialized with empty spaces (' ').

`board = [' ' for _ in range(9)]`

Board Printing:

The `print_board()` function formats and displays the Tic-Tac-Toe board in a user-friendly manner.

It iterates through the board in rows of three and prints each row with separators.

Win Condition Check:

The `check_winner(player)` function determines if a player (either 'X' or 'O') has won the game.

It checks all possible win conditions (rows, columns, and diagonals) against the current board state.

`win_conditions` list stores tuples representing the indices of cells that form a winning line.

Tie Condition Check:

The `is_board_full()` function checks if all spaces on the board are filled, indicating a tie game.

It returns `True` if there are no empty spaces (' ') left on the board.

Available Moves:

The `available_moves()` function returns a list of indices representing the empty spaces on the board, which are the valid moves for the current turn.

Minimax Algorithm:

The `minimax(is_maximizing)` function implements the Minimax algorithm to determine the optimal move for the AI player.

It recursively explores all possible game states, assigning scores based on the outcome (win, lose, or tie).

`is_maximizing` parameter indicates whether the current move is for the AI (maximizing player) or the human player (minimizing player).

Base cases:

AI wins: returns 1

Human wins: returns -1

Tie: returns 0

Recursive step:

Maximizing player: chooses the move that maximizes the score.

Minimizing player: chooses the move that minimizes the score.

AI Move Selection:

The `ai_move()` function uses the Minimax algorithm to select the best move for the AI player.

It iterates through the available moves, calculates the Minimax score for each move, and chooses the move with the highest score.

Player Move Handling:

The `player_move()` function handles the human player's move.

It prompts the player to enter a move (1-9), validates the input, and updates the board with the player's mark ('X').

Includes input validation to prevent invalid moves.

Game Over Check:

The `is_game_over()` function checks if the game is over, either due to a win or a tie.

It returns `True` if either `check_winner()` returns `True` for either player or `is_board_full()` returns `True`.

Main Game Loop:

The `play_game()` function implements the main game loop.

It initializes the game, prints the board, and alternates between the player's move and the AI's move until the game is over.

After the game is over, it announces the winner or declares a tie.

Decisions Made:

The Tic-Tac-Toe game will be implemented using Python.

The AI will use the Minimax algorithm to make its moves.

The board will be represented as a list of 9 elements.

The human player will be represented by 'X' and the AI player by 'O'.

Action Items:

Krishna Thakkar: Review and refine the code for potential optimizations and edge cases. Consider adding features like difficulty levels or a graphical user interface.