**Introduction**

This Python script implements a simple console-based Tic-Tac-Toe game for two players, where one player is a human and the other is a computer. The game allows for a choice of one or two human players and employs a strategic algorithm for the computer player to make its moves. The primary goal is to create a functional and entertaining game with clear user instructions.

**Functions**

**1. display\_board()**

Purpose: Displays the current state of the Tic-Tac-Toe board.

Logic: Utilizes a formatted print statement to visualize the board with its current markings.

**2. is\_full()**

Purpose: Checks if the Tic-Tac-Toe board is completely filled.

Logic: Returns True if no empty spaces are left on the board; otherwise, returns False.

**3. has\_won(symbol, current\_board)**

Purpose: Checks if a player with a given symbol has won the game.

Logic: Examines the rows, columns, and diagonals of the board to determine if any of them contain three consecutive symbols of the same player. Returns True if a winning condition is found; otherwise, returns False.

**4. get\_valid\_moves()**

Purpose: Retrieves a list of valid moves on the current board.

Logic: Iterates through the board and appends the indices of empty spaces to the list.

**5. get\_human\_move(symbol)**

Purpose: Gets the human player's move input.

Logic: Uses a while loop to continually prompt the user for input until a valid move is entered. Validates input for both numeric format and availability on the board.

**6. get\_computer\_move(symbol)**

Purpose: Determines the computer player's move using a strategic algorithm.

Algorithm:

Winning Move Check: Checks if the computer can win in the next move and plays accordingly.

Blocking Move Check: Checks if the human can win in the next move and blocks the winning move.

Center Move Priority: Attempts to take the center if available.

Corner Move Priority: Attempts to take a corner if available.

Random Move: Chooses a random move if no strategic moves are available.

**7. main()**

Purpose: Orchestrates the main game loop, alternating turns between human and computer players until a win, a tie, or the board is full.

Logic: Uses a while loop to iterate through the game turns, updating the board based on user input and computer moves.

**Usage**

Run the script.

Choose the number of human players (1 or 2).

The game begins, and players take turns making moves.

The game ends when a player wins, it's a tie, or the board is full.

The result is displayed at the end of the game.