Jenna Leali - Project 1 - Sep 18, 2023

```
def print_app_purpose():
    print("Welcome to Tic-Tac-Toe!")
    print("Play against the computer and try to win.")

print_app_purpose()
```

```
Welcome to Tic-Tac-Toe!
Play against the computer and try to win.
```

Importing the random Module

```
import random
```

Initializing the Tic-Tac-Toe Board

```
board = [[' ' for _ in range(3)] for _ in range(3)] # Each cell is initially fillec
```

Display Tic-Tac-Toe Board

```
def display_board(board):
    for row in board:
        print(" | ".join(row)) # Join each cell in a row with " | "
        print("-" * 9) # Print a horizontal line to separate rows
```

Check Win Condition

```
def check_win(board, player):
    # Check rows
    for row in board:
        if all(cell == player for cell in row):
            return True

    # Check columns
    for col in range(3):
        if all(board[row][col] == player for row in range(3)):
            return True

    # Check diagonals
    if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == pla
        return True

    return False
```

Check for a Tie

```
# Function to check if the board is full (tie)
def check_tie(board):
    return all(cell != ' ' for row in board for cell in row)
```

Function to allow user to make a move on the tic-tac-toe board

```
# Function for the user's move
def user_move():
    while True:
        try:
            move = int(input("Enter your move (1-9): ")) # Get user input
            if 1 <= move <= 9:
                row, col = divmod(move - 1, 3) # Calculate row and column
                if board[row][col] == ' ': # Check if the spot is available
                    board[row][col] = 'X' # Update the board with 'X'
                    break # Exit the loop if the move is valid
                else:
                    print("That spot is already taken. Try again.") # Display a m
            else:
                print("Invalid input. Please enter a number between 1 and 9.") # I
        except ValueError:
            print("Invalid input. Please enter a number between 1 and 9.") # Disp
```

Computer's Move Using Minimax

```
# Function for the computer's move using Minimax
def computer_move():
    best_score = -float('inf') # Initialize the best score as negative infinity
    best_move = None # Initialize the best move as None

    for row in range(3):
        for col in range(3):
            if board[row][col] == ' ':
                board[row][col] = 'O'
                score = minimax(board, 0, False) # Evaluate the move using Minima
                board[row][col] = ' ' # Reset the board

                if score > best_score:
                    best_score = score
                    best_move = (row, col) # Update the best move if a better mov

    if best_move:
        board[best_move[0]][best_move[1]] = 'O' # Make the best move for the comp
```

Minimax Algorithm

```
# Minimax algorithm
def minimax(board, depth, is_maximizing):
    scores = {'X': -1, 'O': 1, 'tie': 0}

    if check_win(board, 'O'): # Check if the game is won by 'O' or 'X' or if it's
        return scores['O']
    if check_win(board, 'X'):
        return scores['X']
    if check_tie(board):
        return scores['tie']

    if is_maximizing:
        best_score = -float('inf') # Initialize the best score for maximizing pla
        for row in range(3):
            for col in range(3):
                if board[row][col] == ' ':
                    board[row][col] = 'O' # Try making a move ('O')
                    score = minimax(board, depth + 1, False) # Recursively evalua
                    board[row][col] = ' ' # Reset the board
                    best_score = max(score, best_score) # Update the best score
        return best_score
    else:
        best_score = float('inf') # Initialize the best score for minimizing play
        for row in range(3):
            for col in range(3):
                if board[row][col] == ' ':
                    board[row][col] = 'X' # Try making a move ('X')
                    score = minimax(board, depth + 1, True) # Recursively evaluat
                    board[row][col] = ' ' # Reset the board
                    best_score = min(score, best_score) # Update the best score
        return best_score
```

Playing Multiple Rounds and Main Game Loop

```
# Outer loop for playing multiple rounds
while True:
    # Initialize the board for a new round
    board = [[' ' for _ in range(3)] for _ in range(3)]

    # Main game loop
    while True:
        display_board(board) # Display the current state of the board
        user_move()  # Handle the user's move
```

```
        if check_win(board, 'X'): # Check if the user has won or if it's a tie
            display_board(board)
            print("You win!")
            break
        elif check_tie(board):
            display_board(board)
            print("It's a tie!")
            break

        computer_move() # Handle the computer's move

        if check_win(board, 'O'): # Check if the computer has won or if it's a ti
            display_board(board)
            print("Computer wins!")
            break

        if check_tie(board):
            display_board(board)
            print("It's a tie!")

    # Prompt for playing another round
    play_again = input("Play another round? (yes/no): ")
    if play_again.lower() != "yes":
        break
```

```
  |   |
---------
  |   |
---------
  |   |
---------
Enter your move (1-9): 1
X |   |
---------
  | O |
---------
  |   |
---------
Enter your move (1-9): 9
X | O |
---------
  | O |
---------
  |   | X
---------
Enter your move (1-9): 7
X | O |
---------
  | O |
---------
X | O | X
---------
Computer wins!
Play another round? (yes/no): no
```