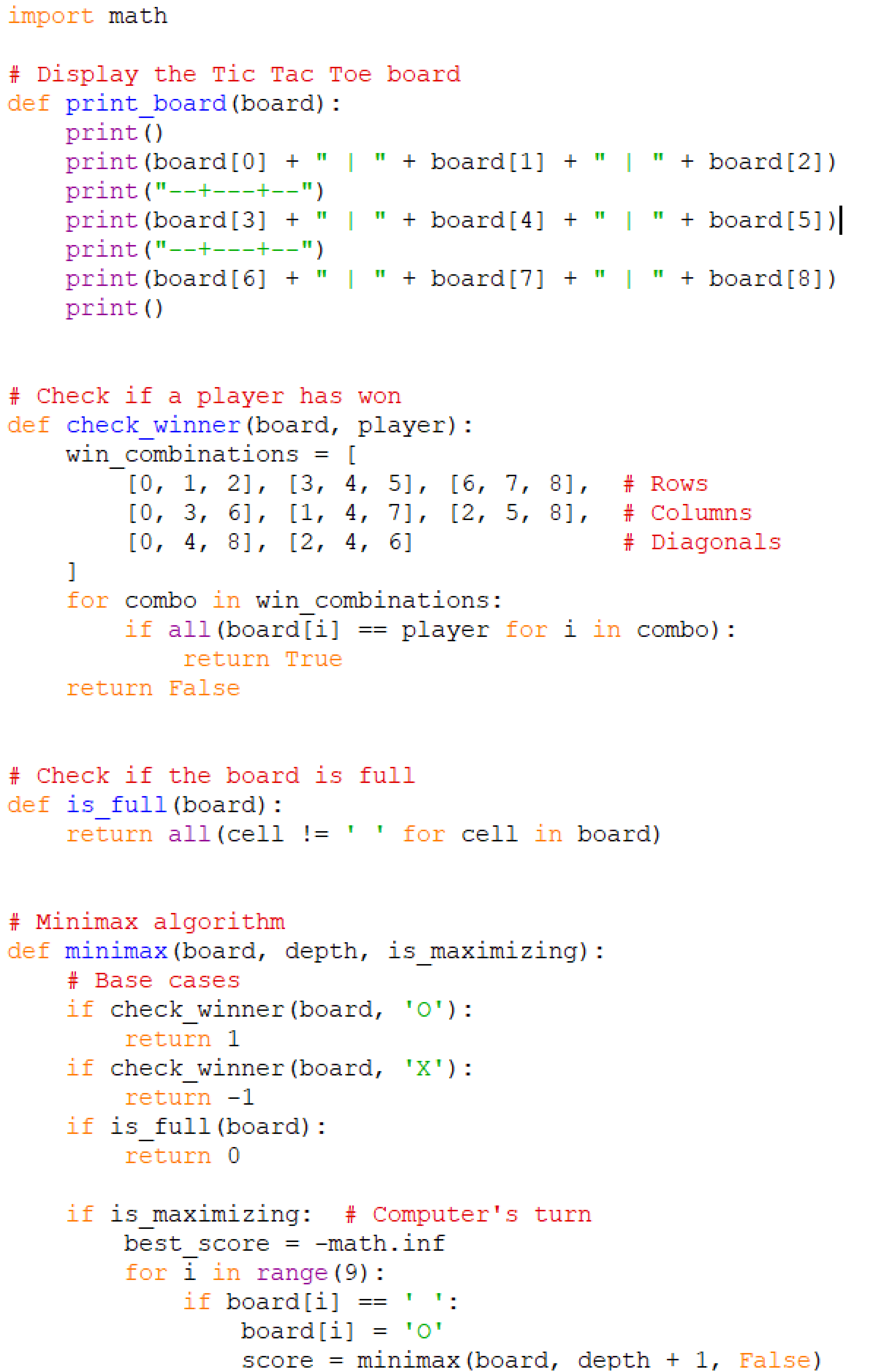
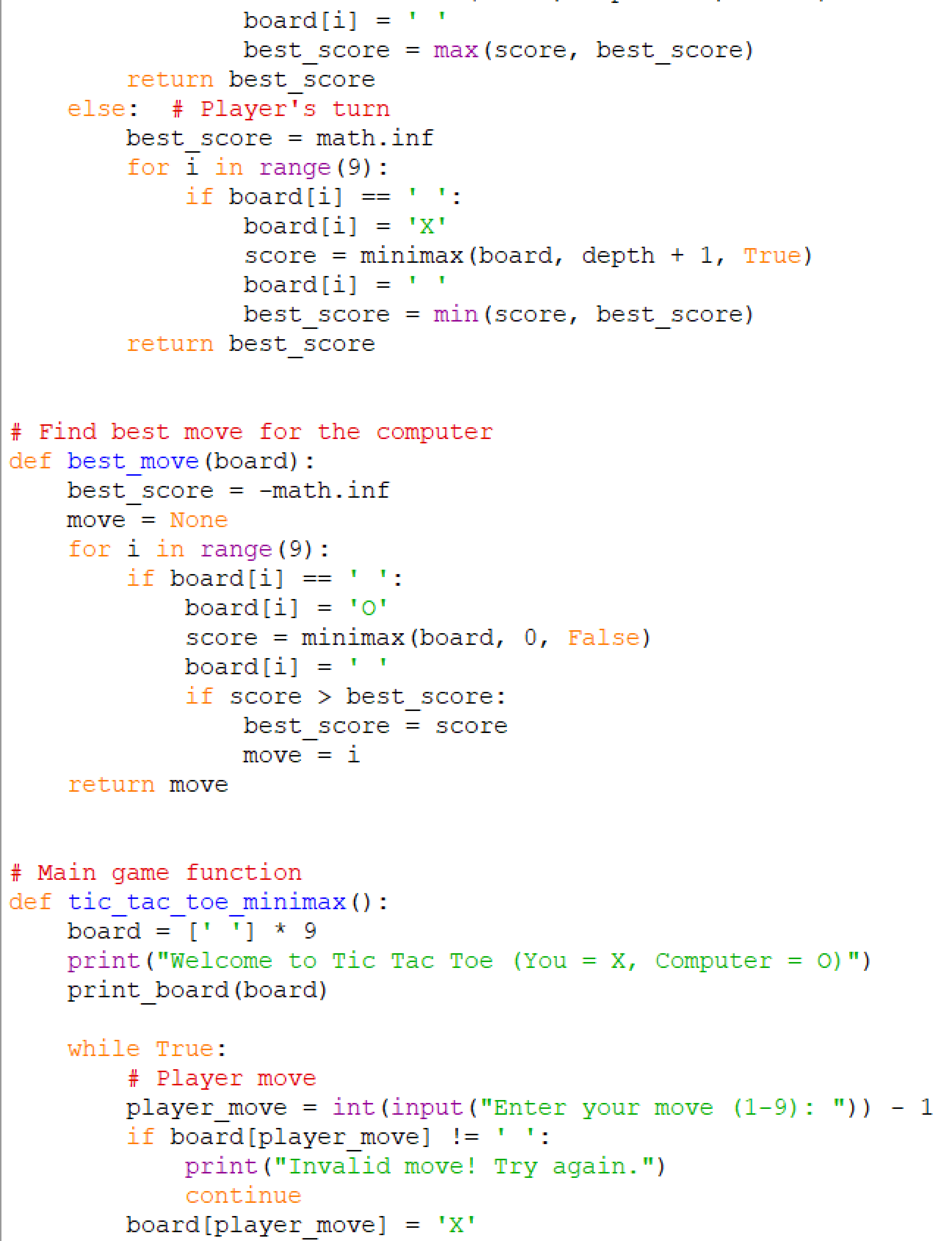
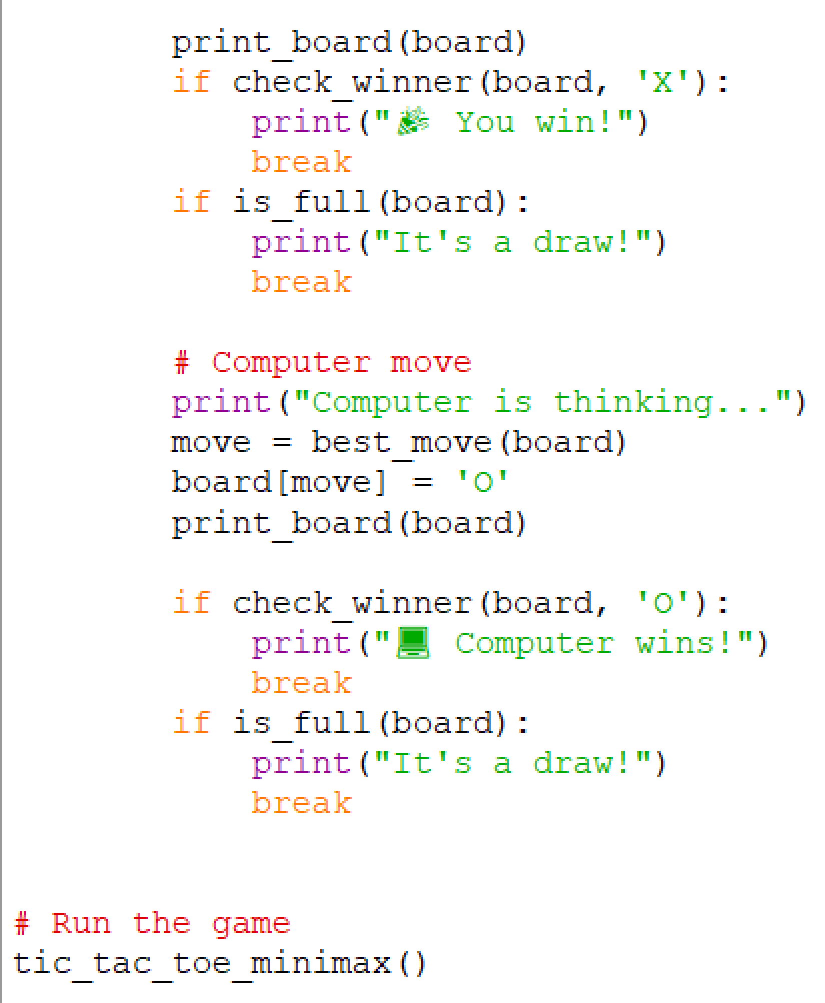


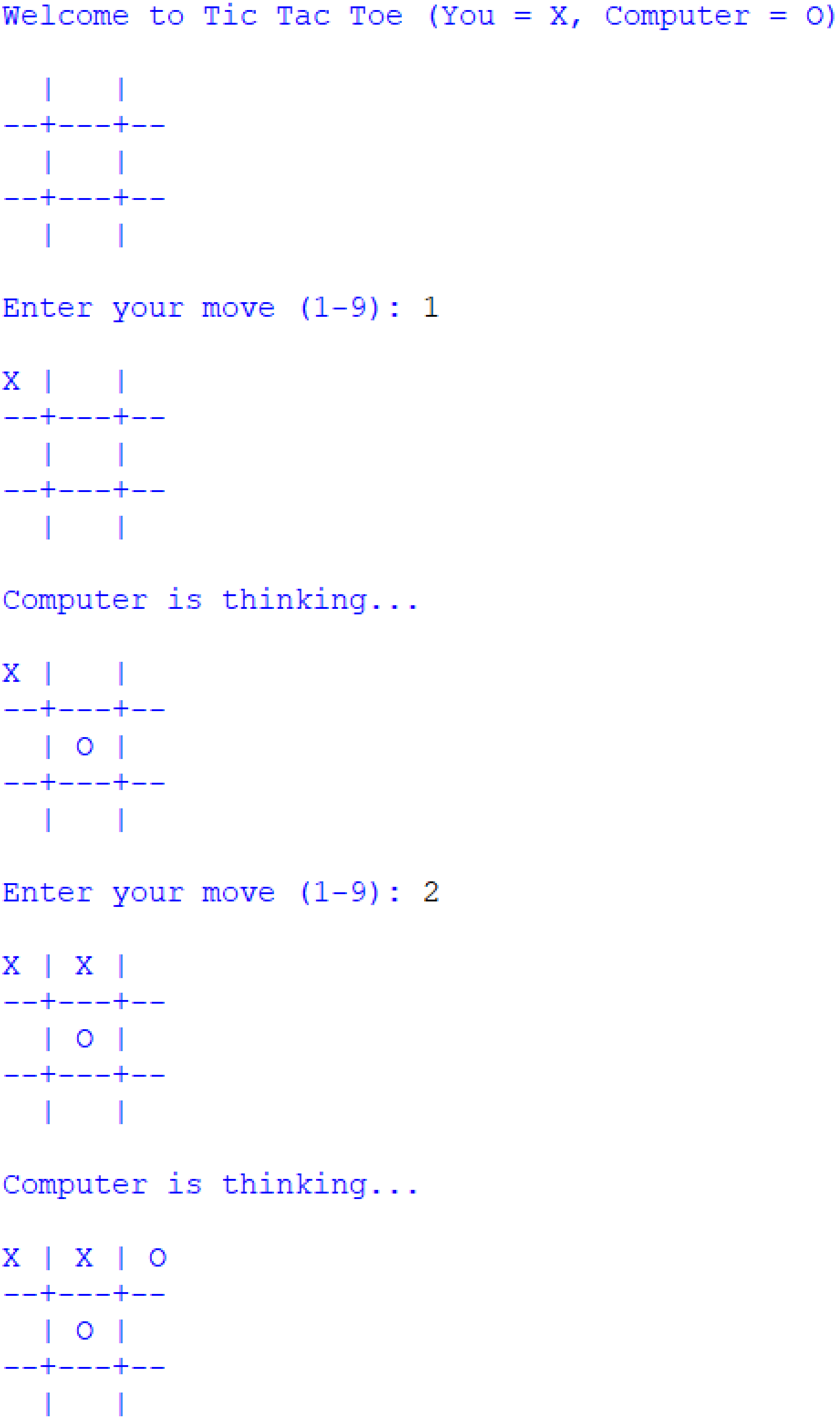
```
import math

# Display the Tic Tac Toe board
def print_board(board):
    print()
    print(board[0] + " | " + board[1] + " | " + board[2])
    print("--+---+--")
    print(board[3] + " | " + board[4] + " | " + board[5])
    print("--+---+--")
    print(board[6] + " | " + board[7] + " | " + board[8])
    print()


# Check if a player has won
def check_winner(board, player):
    win_combinations = [
        [0, 1, 2], [3, 4, 5], [6, 7, 8],   # Rows
        [0, 3, 6], [1, 4, 7], [2, 5, 8],   # Columns
        [0, 4, 8], [2, 4, 6]               # Diagonals
    ]
    for combo in win_combinations:
        if all(board[i] == player for i in combo):
            return True
    return False


# Check if the board is full
def is_full(board):
    return all(cell != ' ' for cell in board)


# Minimax algorithm
def minimax(board, depth, is_maximizing):
    # Base cases
    if check_winner(board, 'O'):
        return 1
    if check_winner(board, 'X'):
        return -1
    if is_full(board):
        return 0

    if is_maximizing:  # Computer's turn
        best_score = -math.inf
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'O'
                score = minimax(board, depth + 1, False)
```



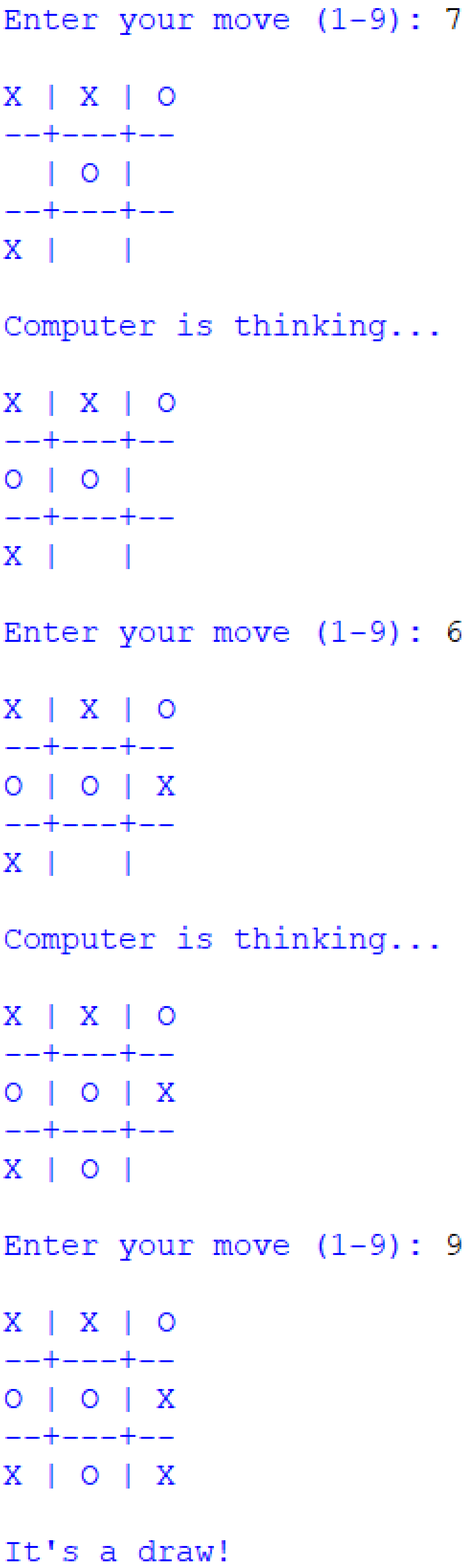
```
                board[i] = ' '
                best_score = max(score, best_score)
        return best_score
    else:  # Player's turn
        best_score = math.inf
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'X'
                score = minimax(board, depth + 1, True)
                board[i] = ' '
                best_score = min(score, best_score)
        return best_score


# Find best move for the computer
def best_move(board):
    best_score = -math.inf
    move = None
    for i in range(9):
        if board[i] == ' ':
            board[i] = 'O'
            score = minimax(board, 0, False)
            board[i] = ' '
            if score > best_score:
                best_score = score
                move = i
    return move


# Main game function
def tic_tac_toe_minimax():
    board = [' '] * 9
    print("Welcome to Tic Tac Toe (You = X, Computer = O)")
    print_board(board)

    while True:
        # Player move
        player_move = int(input("Enter your move (1-9): ")) - 1
        if board[player_move] != ' ':
            print("Invalid move! Try again.")
            continue
        board[player_move] = 'X'
```



```python
        print_board(board)
        if check_winner(board, 'X'):
            print("🎉 You win!")
            break
        if is_full(board):
            print("It's a draw!")
            break

        # Computer move
        print("Computer is thinking...")
        move = best_move(board)
        board[move] = 'O'
        print_board(board)

        if check_winner(board, 'O'):
            print("💻 Computer wins!")
            break
        if is_full(board):
            print("It's a draw!")
            break


# Run the game
tic_tac_toe_minimax()
```



```
Welcome to Tic Tac Toe (You = X, Computer = O)

  |   |  
--+---+--
  |   |  
--+---+--
  |   |  

Enter your move (1-9): 1

X |   |  
--+---+--
  |   |  
--+---+--
  |   |  

Computer is thinking...

X |   |  
--+---+--
  | O |  
--+---+--
  |   |  

Enter your move (1-9): 2

X | X |  
--+---+--
  | O |  
--+---+--
  |   |  

Computer is thinking...

X | X | O
--+---+--
  | O |  
--+---+--
  |   |  
```



```
Enter your move (1-9): 7

X | X | O
--+---+--
  | O |
--+---+--
X |   |

Computer is thinking...

X | X | O
--+---+--
O | O |
--+---+--
X |   |

Enter your move (1-9): 6

X | X | O
--+---+--
O | O | X
--+---+--
X |   |

Computer is thinking...

X | X | O
--+---+--
O | O | X
--+---+--
X | O |

Enter your move (1-9): 9

X | X | O
--+---+--
O | O | X
--+---+--
X | O | X

It's a draw!
```