import math

# Initialize the Tic-Tac-Toe board

def create\_board():

    return [' ' for \_ in range(9)]  # A list of 9 spaces representing the board

# Display the board

def display\_board(board):

    print("\n")

    for i in range(0, 9, 3):

        print(f" {board[i]} | {board[i+1]} | {board[i+2]} ")

        if i < 6:

            print("---|---|---")

    print("\n")

# Check if a player has won

def check\_winner(board, player):

    win\_patterns = [

        [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

    ]

    return any(all(board[pos] == player for pos in pattern) for pattern in win\_patterns)

# Check if the board is full

def is\_draw(board):

    return ' ' not in board

# Evaluate the board (used by Minimax)

def evaluate\_game(board):

    if check\_winner(board, 'O'):  # AI wins

        return 1

    elif check\_winner(board, 'X'):  # Human wins

        return -1

    else:  # Draw

        return 0

# Minimax decision-making algorithm

def minimax\_decision(board, depth, maximizing\_player):

    if check\_winner(board, 'O') or check\_winner(board, 'X') or is\_draw(board):

        return evaluate\_game(board)

    if maximizing\_player:

        best\_score = -math.inf

        for i in range(9):

            if board[i] == ' ':

                board[i] = 'O'

                score = minimax\_decision(board, depth + 1, False)

                board[i] = ' '

                best\_score = max(best\_score, score)

        return best\_score

    else:

        best\_score = math.inf

        for i in range(9):

            if board[i] == ' ':

                board[i] = 'X'

                score = minimax\_decision(board, depth + 1, True)

                board[i] = ' '

                best\_score = min(best\_score, score)

        return best\_score

# Determine the best move for the AI

def ai\_best\_move(board):

    optimal\_score = -math.inf

    optimal\_move = -1

    for i in range(9):

        if board[i] == ' ':

            board[i] = 'O'

            score = minimax\_decision(board, 0, False)

            board[i] = ' '

            if score > optimal\_score:

                optimal\_score = score

                optimal\_move = i

    return optimal\_move

# Main game loop

def play\_game():

    board = create\_board()

    print("Welcome to Tic-Tac-Toe!")

    display\_board(board)

    while True:

        # Human move

        while True:

            try:

                human\_move = int(input("Enter your move (1-9): ")) - 1

                if board[human\_move] == ' ':

                    board[human\_move] = 'X'

                    break

                else:

                    print("That spot is already taken!")

            except (ValueError, IndexError):

                print("Invalid move. Try again!")

        display\_board(board)

        if check\_winner(board, 'X'):

            print("You win!")

            break

        if is\_draw(board):

            print("It's a draw!")

            break

        # AI move

        print("AI is thinking...")

        ai\_move = ai\_best\_move(board)

        board[ai\_move] = 'O'

        display\_board(board)

        if check\_winner(board, 'O'):

            print("AI wins!")

            break

        if is\_draw(board):

            print("It's a draw!")

            break

# Run the game

if \_\_name\_\_ == "\_\_main\_\_":

    play\_game()