```
import math

# Initialize empty Tic-Tac-Toe board
board = [[" " for _ in range(3)] for _ in range(3)]

# Function to print the Tic-Tac-Toe board
def print_board(board):
    for row in board:
        print("|".join(row))
        print("-" * 5)

# Function to check if any moves are left
def is_moves_left(board):
    return any(" " in row for row in board)

# Function to evaluate the board
def evaluate(board):
    for i in range(3):
        # Check rows and columns
        if board[i][0] == board[i][1] == board[i][2] != " ":
            return 10 if board[i][0] == "O" else -10
        if board[0][i] == board[1][i] == board[2][i] != " ":
            return 10 if board[0][i] == "O" else -10

    # Check diagonals
    if board[0][0] == board[1][1] == board[2][2] != " ":
        return 10 if board[0][0] == "O" else -10
    if board[0][2] == board[1][1] == board[2][0] != " ":
        return 10 if board[0][2] == "O" else -10

    return 0  # No winner yet

# Minimax function with Alpha-Beta Pruning
def minimax(board, depth, is_max, alpha, beta):
    score = evaluate(board)

    # Base cases: AI wins, Player wins, or Draw
    if score == 10:
        return score - depth  # Favor quicker wins
    if score == -10:
        return score + depth  # Favor delaying losses
    if not is_moves_left(board):
        return 0  # Draw

    if is_max:  # AI's turn (Maximizing)
        best = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == " ":
```

```
                        board[i][j] = "O"
                        best = max(best, minimax(board, depth + 1, False, alpha, beta))
                        board[i][j] = " "  # Undo move
                        alpha = max(alpha, best)
                        if beta <= alpha:
                            break  # Prune
            return best
        else:  # Human's turn (Minimizing)
            best = math.inf
            for i in range(3):
                for j in range(3):
                    if board[i][j] == " ":
                        board[i][j] = "X"
                        best = min(best, minimax(board, depth + 1, True, alpha, beta))
                        board[i][j] = " "  # Undo move
                        beta = min(beta, best)
                        if beta <= alpha:
                            break  # Prune
            return best

    # Function to find the best move for AI
    def find_best_move(board):
        best_val = -math.inf
        best_move = (-1, -1)

        for i in range(3):
            for j in range(3):
                if board[i][j] == " ":
                    board[i][j] = "O"
                    move_val = minimax(board, 0, False, -math.inf, math.inf)
                    board[i][j] = " "  # Undo move

                    if move_val > best_val:
                        best_val = move_val
                        best_move = (i, j)

        return best_move

    # Main game loop
    def play_game():
        print("Welcome to Tic-Tac-Toe! You are 'X', AI is 'O'.")
        print_board(board)

        while True:
            # Human player's move
            try:
                row, col = map(int, input("Enter row and column (0-2, space-separated): ").split())
                if board[row][col] != " ":
                    print("Invalid move! Try again.")
                    continue
                board[row][col] = "X"
```

```
            except (ValueError, IndexError):
                print("Invalid input! Enter two numbers between 0 and 2.")
                continue

            print_board(board)

            if evaluate(board) == -10:
                print("Congratulations! You win! 🎉")
                break
            if not is_moves_left(board):
                print("It's a draw! 🤝")
                break

            # AI's move
            print("AI is making a move...")
            ai_move = find_best_move(board)
            board[ai_move[0]][ai_move[1]] = "O"
            print_board(board)

            if evaluate(board) == 10:
                print("AI wins! Better luck next time. 🤖")
                break
            if not is_moves_left(board):
                print("It's a draw! 🤝")
                break

    # Start the game
    play_game()
```

```
••• Welcome to Tic-Tac-Toe! You are 'X', AI is 'O'.
     | |
    -----
     | |
    -----
     | |
    -----
    X| |
    -----
     | |
    -----
     | |
    -----
    AI is making a move...
    X| |
    -----
     |O|
    -----
     | |
    -----
    X|X|
    -----
     |O|
```

```
-----
 | |
-----
AI is making a move...
X|X|O
-----
 |O|
-----
 | |
-----
Enter row and column (0-2, space-separated):
```