import math

def print\_board(board):

    for row in board:

        print(row)

def evaluate(board):

    for row in range(3):

        if board[row][0] == board[row][1] == board[row][2]:

            if board[row][0] == 'X':

                return 10

            elif board[row][0] == 'O':

                return -10

    for col in range(3):

        if board[0][col] == board[1][col] == board[2][col]:

            if board[0][col] == 'X':

                return 10

            elif board[0][col] == 'O':

                return -10

    if board[0][0] == board[1][1] == board[2][2]:

        if board[0][0] == 'X':

            return 10

        elif board[0][0] == 'O':

            return -10

    if board[0][2] == board[1][1] == board[2][0]:

        if board[0][2] == 'X':

            return 10

        elif board[0][2] == 'O':

            return -10

    return 0

def is\_moves\_left(board):

    for row in board:

        if ' ' in row:

            return True

    return False

def minimax(board, depth, is\_maximizing):

    score = evaluate(board)

    if score == 10:

        return score - depth

    if score == -10:

        return score + depth

    if not is\_moves\_left(board):

        return 0

    if is\_maximizing:

        best = -math.inf

        for i in range(3):

            for j in range(3):

                if board[i][j] == ' ':

                    board[i][j] = 'X'

                    best = max(best, minimax(board, depth + 1, not is\_maximizing))

                    board[i][j] = ' '

        return best

    else:

        best = math.inf

        for i in range(3):

            for j in range(3):

                if board[i][j] == ' ':

                    board[i][j] = 'O'

                    best = min(best, minimax(board, depth + 1, not is\_maximizing))

                    board[i][j] = ' '

        return best

def find\_best\_move(board):

    best\_value = -math.inf

    best\_move = (-1, -1)

    for i in range(3):

        for j in range(3):

            if board[i][j] == ' ':

                board[i][j] = 'X'

                move\_value = minimax(board, 0, False)

                board[i][j] = ' '

                if move\_value > best\_value:

                    best\_move = (i, j)

                    best\_value = move\_value

    return best\_move

board = [

    ['X', 'O', 'X'],

    ['O', 'O', 'X'],

    [' ', ' ', ' ']

]

print("Current board:")

print\_board(board)

best\_move = find\_best\_move(board)

print(f"\nBest move for X: {best\_move}")