import math

# Function to print the Tic-Tac-Toe board

def print\_board(board):

for row in board:

print(" | ".join(row))

print("-" \* 5)

# Check if any player has won

def check\_winner(board):

# Rows

for row in board:

if row[0] == row[1] == row[2] and row[0] != "\_":

return row[0]

# Columns

for col in range(3):

if board[0][col] == board[1][col] == board[2][col] and board[0][col] != "\_":

return board[0][col]

# Diagonals

if board[0][0] == board[1][1] == board[2][2] and board[0][0] != "\_":

return board[0][0]

if board[0][2] == board[1][1] == board[2][0] and board[0][2] != "\_":

return board[0][2]

return None

# Check if board is full

def is\_full(board):

for row in board:

if "\_" in row:

return False

return True

# Minimax Algorithm

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

winner = check\_winner(board)

if winner == "X": # Maximizing player wins

return 1

elif winner == "O": # Minimizing player wins

return -1

elif is\_full(board):

return 0 # Draw

if is\_maximizing:

best\_val = -math.inf

for i in range(3):

for j in range(3):

if board[i][j] == "\_":

board[i][j] = "X"

value = minimax(board, depth + 1, False)

board[i][j] = "\_"

best\_val = max(best\_val, value)

return best\_val

else:

best\_val = math.inf

for i in range(3):

for j in range(3):

if board[i][j] == "\_":

board[i][j] = "O"

value = minimax(board, depth + 1, True)

board[i][j] = "\_"

best\_val = min(best\_val, value)

return best\_val

# Find best move for maximizing player (X)

def best\_move(board):

best\_val = -math.inf

move = (-1, -1)

for i in range(3):

for j in range(3):

if board[i][j] == "\_":

board[i][j] = "X"

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

board[i][j] = "\_"

if move\_val > best\_val:

best\_val = move\_val

move = (i, j)

return move

# --- Driver code ---

board = [

["X", "O", "X"],

["O", "O", "\_"],

["\_", "X", "\_"]

]

print("Current Board:")

print\_board(board)

best = best\_move(board)

print(f"\nBest Move for X is at position: {best}")