AI23231-PRINCIPLES OF ARTIFICIAL INTELLIGENCE LAB

Name : Vijay Antony.L

Roll no : 241801311

Ex no : 03

Ex name : IMPLEMENTATION OF MINIMAX ALGORITHM

Date : 26/03/2025

PROBLEM: import

math

PLAYER_X = 'X'

PLAYER_O = 'O' EMPTY

= '.' def

print_board(board):

for row in board:

    print(' | '.join(row))

print('-' * 5)

def evaluate(board): for row in board: if row[0] ==

row[1] == row[2] != EMPTY: return 1 if row[0] ==

PLAYER_X else -1 for col in range(3): if board[0][col] ==

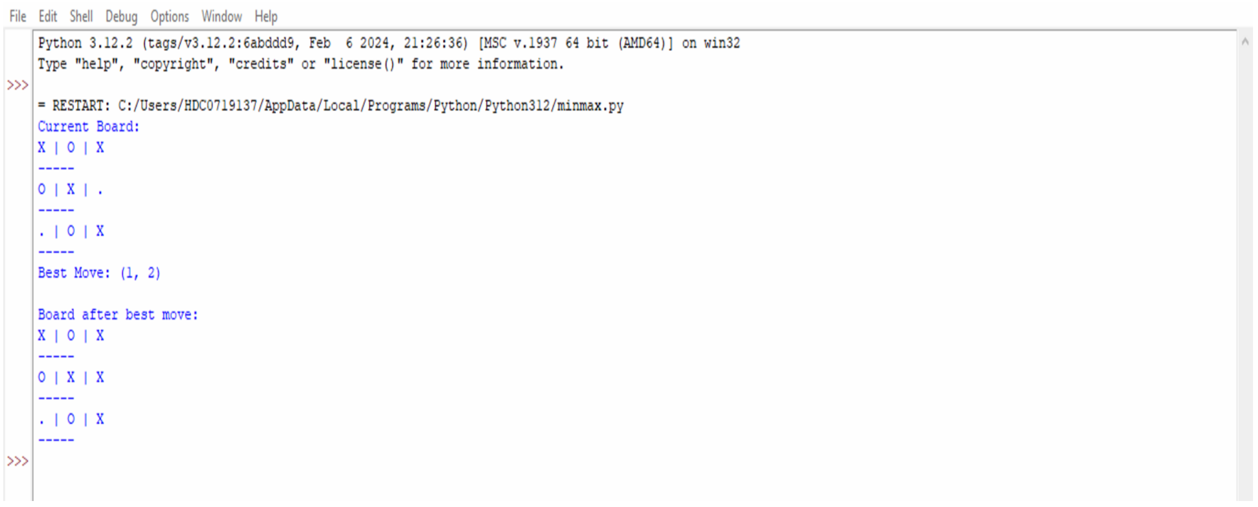
board[1][col] == board[2][col] != EMPTY:

```
            return 1 if board[0][col] == PLAYER_X else -1         if
board[0][0] == board[1][1] == board[2][2] != EMPTY:

            return 1 if board[0][0] == PLAYER_X else -1
    if board[0][2] == board[1][1] == board[2][0] != EMPTY:

        return 1 if board[0][2] == PLAYER_X else -1

        return 0

def is_moves_left(board):

        return any(EMPTY in row for row in board)  def
minimax(board, depth, alpha, beta, is_max):

     score = evaluate(board)

if score == 1:

        return score - depth

if score == -1:

        return score + depth     if
not is_moves_left(board):

        return 0

if is_max:

        best = -math.inf         for i in range(3):           for j in range(3):               if
board[i][j] == EMPTY:                    board[i][j] = PLAYER_X                   best =
max(best, minimax(board, depth + 1, alpha, beta, not is_max))

                    board[i][j] = EMPTY

alpha = max(alpha, best)

if beta <= alpha:
```

```
                    break
return best     else:

            best  =  math.inf
for     i     in     range(3):
for     j     in     range(3):
if board[i][j] == EMPTY:
board[i][j]  =  PLAYER_O
best       =       min(best,
minimax(board,   depth
+  1,  alpha,  beta,  not
is_max))
board[i][j]     =     EMPTY
beta  =  min(beta,  best)
if beta <= alpha:

                    break
return best

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] == EMPTY:               board[i][j]
= PLAYER_X                move_val = minimax(board, 0, -math.inf,
math.inf, False)

                    board[i][j]  =  EMPTY
if      move_val      >      best_val:
best_move         =         (i,         j)
best_val = move_val
```

```
    return best_move      if
__name__ == "__main__":

        board = [
        [PLAYER_X, PLAYER_O, PLAYER_X],
        [PLAYER_O, PLAYER_X, EMPTY],
        [EMPTY, PLAYER_O, PLAYER_X]
    ]
    print("Current Board:")
print_board(board)     move =
find_best_move(board)     print(f"Best Move:
{move}")     board[move[0]][move[1]] =
PLAYER_X     print("\nBoard after best
move:")     print_board(board)
```

OUTPUT:



```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/HDC0719137/AppData/Local/Programs/Python/Python312/minmax.py
Current Board:
X | O | X
-----
O | X | .
-----
. | O | X
-----
Best Move: (1, 2)

Board after best move:
X | O | X
-----
O | X | X
-----
. | O | X
-----
>>>
```