

main.py

```
1 PLAYER_X = 1
2 PLAYER_O = -1
3 EMPTY = 0
4 def evaluate(board):
5     for row in range(3):
6         if board[row][0] == board[row][1] == board[row][2] != EMPTY:
7             return board[row][0]
8     for col in range(3):
9         if board[0][col] == board[1][col] == board[2][col] != EMPTY:
10            return board[0][col]
11    if board[0][0] == board[1][1] == board[2][2] != EMPTY:
12        return board[0][0]
13    if board[0][2] == board[1][1] == board[2][0] != EMPTY:
14        return board[0][2]
15    return 0
16 def isMovesLeft(board):
17     for row in range(3):
18         for col in range(3):
19             if board[row][col] == EMPTY:
20                 return True
21     return False
22 def minimax(board, isMax):
23     score = evaluate(board)
24     if score == PLAYER_X: return score
25     if score == PLAYER_O: return score
26     if not isMovesLeft(board): return 0
```

Output

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
```

=== Code Execution Successful ===

Upcoming
Earnings

Search

```

25 11 score == PLAYER_O: return score
26 if not isMovesLeft(board): return 0
27 if isMax:
28     best = -float('inf')
29     for row in range(3):
30         for col in range(3):
31             if board[row][col] == EMPTY:
32                 board[row][col] = PLAYER_X
33                 best = max(best, minimax(board, not isMax))
34                 board[row][col] = EMPTY
35     return best
36 else:
37     best = float('inf')
38     for row in range(3):
39         for col in range(3):
40             if board[row][col] == EMPTY:
41                 board[row][col] = PLAYER_O
42                 best = min(best, minimax(board, not isMax))
43                 board[row][col] = EMPTY
44     return best
45 def findBestMove(board):
46     bestVal = -float('inf')
47     bestMove = (-1, -1)
48     for row in range(3):
49         for col in range(3):
50             if board[row][col] == EMPTY:

```

Output

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

=== Code Execution Successful ===

```

48- for row in range(3):
49-     for col in range(3):
50-         if board[row][col] == EMPTY:
51-             board[row][col] = PLAYER_X
52-             moveVal = minimax(board, False)
53-             board[row][col] = EMPTY
54-             if moveVal > bestVal:
55-                 bestMove = (row, col)
56-                 bestVal = moveVal
57-     return bestMove
58- def printBoard(board):
59-     for row in board:
60-         print(" ".join(["X" if x == PLAYER_X else "O" if x
61- ==PLAYER_O else "." for x in row]))
62- board = [
63-     [PLAYER_X, PLAYER_O, PLAYER_X],
64-     [PLAYER_O, PLAYER_X, EMPTY],
65-     [EMPTY, PLAYER_O, PLAYER_X]
66- ]
67- print("Current Board:")
68- printBoard(board)
69- move = findBestMove(board)
70- print(f"Best Move: {move}")
71- board[move[0]][move[1]] = PLAYER_X
72- print("\nBoard after best move:")
73- printBoard(board)

```

Output

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

=== Code Execution Successful ===