Write the python program to implement Minimax algorithm for gaming

AIM

To implement the **Minimax algorithm** in Python to determine the optimal move for player "O" in Tic Tac Toe.

ALGORITHM

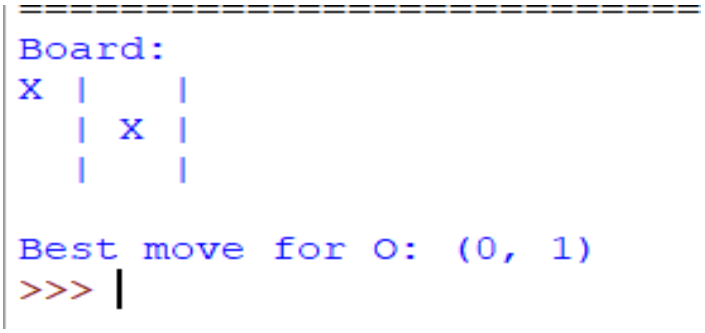
1. Represent the Tic Tac Toe board as a 3×3 grid.
2. Define functions to:
    a. Check if a player has won (is_winner).
    b. Check if the board is full (is_full).
3. Implement **Minimax**:
    a. If "O" wins → return +1.
    b. If "X" wins → return -1.
    c. If draw → return 0.
    d. Otherwise, recursively evaluate all possible moves for maximizer and minimizer.
4. Use best_move to select the move with the highest score for "O".
5. Output the best move coordinates.

map colouring.py - C:/Users/gayathri/map colouring.py (3.8.2)

File Edit Format Run Options Window Help

```
import math
def print_board(board):
    for row in board:
        print(" | ".join(row))
    print()
def is_winner(board, player):
    for row in board:
        if all(cell == player for cell in row):
            return True
    for col in range(3):
        if all(board[row][col] == player for row in range(3)):
            return True
    if all(board[i][i] == player for i in range(3)) or all(board[i][2-i] == player for i in range(3)):
        return True
    return False
def is_full(board):
    return all(cell != " " for row in board for cell in row)
def minimax(board, depth, is_maximizing):
    if is_winner(board, "O"): return 1
    if is_winner(board, "X"): return -1
    if is_full(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] = "O"
                    best = max(best, minimax(board, depth+1, False))
                    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] = "X"
                    best = min(best, minimax(board, depth+1, True))
                    board[i][j] = " "
        return best
def best_move(board):
    move = None
    best_val = -math.inf
    for i in range(3):
        for j in range(3):
            if board[i][j] == " ":
                board[i][j] = "O"
                move_val = minimax(board, 0, False)
                board[i][j] = " "
                if move_val > best_val:
                    best_val = move_val
                    move = (i, j)

    return move
board = [[" "]*3 for _ in range(3)]
board[0][0] = "X"
board[1][1] = "X"
print("Board:")
print_board(board)
move = best_move(board)
print("Best move for O:", move)
```



RESULT

The program successfully evaluates all possible moves and selects the **optimal move** for player "O".