

```
1 import math
2
3 # Board is a 3x3 list of lists
4 def create_board():
5     return [[' ' for _ in range(3)] for _ in range(3)]
6
7 def print_board(board):
8     for row in board:
9         print('|'.join(row))
10        print('-' * 5)
11
12 def is_winner(board, player):
13     # Rows, columns and diagonals
14     for i in range(3):
15         if all(cell == player for cell in board[i]): # Row
16             return True
17         if all(board[j][i] == player for j in range(3)): # Column
18             return True
19     if all(board[i][i] == player for i in range(3)): # Diagonal
20         return True
21     if all(board[i][2 - i] == player for i in range(3)): # Anti-diagonal
22         return True
23     return False
24
25 def is_full(board):
26     return all(cell != ' ' for row in board for cell in row)
27
28 def get_available_moves(board):
29     return [(i, j) for i in range(3) for j in range(3) if board[i][j] == ' ']
30
31 def minimax(board, is_maximizing):
32     if is_winner(board, 'O'):
33         return 1
34     if is_winner(board, 'X'):
35         return -1
36     if is_full(board):
37         return 0
38
39     if is_maximizing:
40         best_score = -math.inf
41         for (i, j) in get_available_moves(board):
42             board[i][j] = 'O'
43             score = minimax(board, False)
44             board[i][j] = ' '
45             best_score = max(score, best_score)
46         return best_score
47     else:
48         best_score = math.inf
49         for (i, j) in get_available_moves(board):
50             board[i][j] = 'X'
```

```
51         score = minimax(board, True)
52         board[i][j] = ' '
53         best_score = min(score, best_score)
54     return best_score
55
56 def best_move(board):
57     best_score = -math.inf
58     move = None
59     for (i, j) in get_available_moves(board):
60         board[i][j] = 'O'
61         score = minimax(board, False)
62         board[i][j] = ' '
63         if score > best_score:
64             best_score = score
65             move = (i, j)
66     return move
67
68 def play_game():
69     board = create_board()
70     print("Welcome to Tic-Tac-Toe! You are X, AI is O.")
71     print_board(board)
72
73     while True:
74         # Human move
75         while True:
76             try:
77                 x, y = map(int, input("Enter your move (row and column: 0-2):
78 ").split())
79                 if board[x][y] == ' ':
80                     board[x][y] = 'X'
81                     break
82                 else:
83                     print("Cell already taken.")
84             except:
85                 print("Invalid input. Try again.")
86
87     print_board(board)
88     if is_winner(board, 'X'):
89         print("You win!")
90         break
91     if is_full(board):
92         print("It's a draw!")
93         break
94
95     # AI move
96     ai_move = best_move(board)
97     if ai_move:
98         board[ai_move[0]][ai_move[1]] = 'O'
99         print("AI played:")
100         print_board(board)
```

```
101         if is_winner(board, 'O'):  
102             print("AI wins!")  
103             break  
104         if is_full(board):  
105             print("It's a draw!")  
106             break  
107  
108 if __name__ == "__main__":  
109     play_game()  
110
```