**Minimax on Connect-4**

import numpy as np  
  
def main():  
 print("Connect 4!")  
 global board  
 global human   
 global computer  
 global column  
 board = np.array([["[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "1"],   
 ["[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "2"],  
 ["[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "3"],  
 ["[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "4"],  
 ["[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "5"],  
 ["[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "[ ]", "6"],  
 [" a ", " b ", " c ", " d ", " e ", " f ", " g ", " "]])  
 column = {"a":0, "b":1, "c":2, "d":3, "e":4, "f":5, "g":6}  
 human = input("Type 'X' , or 'O'.\n").upper()  
 if human == 'X':  
 print("You are X.")  
 human = "[X]"  
 computer = "[O]"  
 print(board)  
 humanmove()  
 elif human == 'O':  
 print("You are O.")  
 human = "[O]"  
 computer = "[x]"  
 else:  
 print("Invalid Character. \nComputer will move first.")  
 human = "[O]"  
 computer = "[X]"  
 while True:  
 print(board)  
 computermove()  
 if win() != False:  
 print(board)  
 print("computer Wins")  
 break  
 if checktie() == True:  
 print(board)  
 print("Tie. No winner.")  
 break  
 print(board)  
 humanmove()  
 if win() != False:  
 print(board)  
 print("You Win")  
 break  
 if checktie() == True:  
 print(board)  
 print("Tie. No winner.")  
 break  
  
def win():  
  
 for i in range(6):  
 for j in range(4):  
 if board[i][j] != "[ ]" and board[i][j] == board[i][j + 1] == board[i][j + 2] == board[i][j + 3]:  
 return board[i][j]  
  
 for i in range(3):  
 for j in range(7):  
 if board[i][j] != "[ ]" and board[i][j] == board[i + 1][j] == board[i + 2][j] == board[i + 3][j]:  
 return board[i][j]  
  
 for i in range(3):  
 for j in range(4):  
 if board[i][j] != "[ ]" and board[i][j] == board[i + 1][j + 1] == board[i + 2][j + 2] == board[i + 3][j + 3]:  
 return board[i][j]  
  
 for i in range(3):  
 for j in range(3, 7):  
 if board[i][j] != "[ ]" and board[i][j] == board[i + 1][j - 1] == board[i + 2][j - 2] == board[i + 3][j - 3]:  
 return board[i][j]  
  
 return False  
  
def checktie():  
 count = 0  
 for i in range(6):  
 for j in range(7):  
 if board[i][j] == "[ ]":  
 return False  
 else:  
 count = count + 1  
 if count == 42:  
 return True  
   
 return False  
  
def cspace(row, column):  
 if board[row + 1][column] != "[ ]" and board[row][column] == "[ ]":  
 return True   
 else:  
 return False  
  
def humanmove():  
 global board  
 move = input("Your Move. Enter a square <column><row> ex: 'c2'").lower()  
 try:  
 if cspace(int(move[1]) - 1,column[move[0]]) == True:  
 board[int(move[1]) - 1][column[move[0]]] = human  
 else:  
 print("invalid move")  
 humanmove()  
 except:  
 print("invalid move")  
 humanmove()  
  
def computermove():  
 global board  
 alpha = -99999  
 beta = 99999  
 depth = 8  
 print("computer move")  
 bestscore = -9999  
 for i in range(5, 0 , -1):  
 for j in range(6, 0, -1):  
 if cspace(i, j) == True:  
 board[i][j] = computer  
 score = minimax(depth, alpha, beta, False)  
 board[i][j] = "[ ]"  
 if score > bestscore:  
 bestscore = score  
 row = i  
 column = j  
 board[row][column] = computer  
  
def minimax(depth, alpha, beta, maximize):  
 global board  
 depth = depth - 1  
 if win() == computer:  
 return 4  
 elif win() == human:  
 return -4  
 elif checktie() == True:  
 return 0  
 elif depth == 0:  
 return func\_eval()  
   
 if maximize == True:   
 maxscore = -9999  
 for i in range(5, 0, -1):   
 for j in range(6, 0, -1):  
 if cspace(i, j) == True:  
 board[i][j] = computer  
 score = minimax(depth, alpha, beta, False)  
 board[i][j] = "[ ]"  
 maxscore = max(maxscore, score)  
 alpha = max(alpha, maxscore)  
 if alpha >= beta:  
 return maxscore  
 return maxscore  
  
 else: #if humans turn  
 minscore = 9999  
 for i in range(5, 0, -1):   
 for j in range(6, 0, -1):  
 if cspace(i, j) == True:  
 board[i][j] = human  
 score = minimax(depth, alpha, beta, True)   
 board[i][j] = "[ ]"  
 minscore = min(minscore, score)   
 beta = min(beta, minscore)   
 if beta <= alpha:  
 return minscore  
 return minscore  
  
def func\_eval():  
 eval = 0   
   
  
 for i in range(6):  
 for j in range(4):  
 if board[i][j] == computer:   
 if board[i][j] == board[i][j + 1]:  
 eval = eval + 1  
 if board[i][j + 1] == board[i][j + 2]:  
 eval = eval + 1  
  
 for i in range(3):  
 for j in range(7):  
 if board[i][j] == computer:  
 if board[i][j] == board[i + 1][j]:  
 eval = eval + 1  
 if board[i + 1][j] == board[i + 2][j]:  
 eval = eval + 1  
  
 for i in range(3):  
 for j in range(4):  
 if board[i][j] == computer:  
 if board[i][j] == board[i + 1][j + 1]:  
 eval = eval + 1  
 if board[i + 1][j + 1] == board[i + 2][j + 2]:  
 eval = eval + 1  
  
 for i in range(3):  
 for j in range(3, 7):  
 if board[i][j] == computer:  
 if board[i][j] == board[i + 1][j - 1]:  
 eval = eval + 1  
 if board[i + 1][j - 1] == board[i + 2][j - 2]:  
 eval = eval + 1  
   
  
 for i in range(6):  
 for j in range(4):  
 if board[i][j] == human:   
 if board[i][j] == board[i][j + 1]:  
 eval = eval - 1  
 if board[i][j + 1] == board[i][j + 2]:  
 eval = eval - 1  
 #vertical  
 for i in range(3):  
 for j in range(7):  
 if board[i][j] == human:  
 if board[i][j] == board[i + 1][j]:  
 eval = eval - 1  
 if board[i + 1][j] == board[i + 2][j]:  
 eval = eval - 1  
 #right diagonals  
 for i in range(3):  
 for j in range(4):  
 if board[i][j] == human:  
 if board[i][j] == board[i + 1][j + 1]:  
 eval = eval - 1  
 if board[i + 1][j + 1] == board[i + 2][j + 2]:  
 eval = eval - 1  
 #left diagonals  
 for i in range(3):  
 for j in range(3, 7):  
 if board[i][j] == human:  
 if board[i][j] == board[i + 1][j - 1]:  
 eval = eval - 1  
 if board[i + 1][j - 1] == board[i + 2][j - 2]:  
 eval = eval - 1  
 return eval  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()