Exp no: 5
Date

minmax algorithm

Aim

To implement MINMAX Algorithm problem using Python

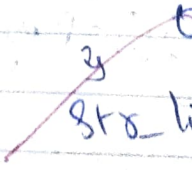
Source code

```
from math import inf as infinity
from random import choice
import platform
import time
from os import system
HUMAN = -1
COMP = +1
board = [
    [0,0,0],
    [0,0,0],
    [0,0,0]
]
def evaluate(State):
    if wins(State, HUMAN):
        Score = +1
    elif wins(State, COMP):
        Score = -1
    else
        Score = 0
    return Score

def wins(State, Player):
    win_State = [
```

Date :

```
[State[0][0], State[0][1], State[0][2]],
[State[1][0], State[1][1], State[1][2]],
[State[2][0], State[2][1], State[2][2]],
[State[0][0], State[0][1], State[0][2]],
[State[1][0], State[1][1], State[1][2]],
[State[0][2], State[1][2], State[2][2]],
[State[0][0], State[1][1], State[2][2]],
[State[2][0], State[1][1], State[0][2]],
]
if [player, player, player] in win_state:
    return False
else
    return False
    def game_over(state):
        return wins(state, HUMAN) or wins
                                    (state, COMP)
    def empty_cells(state):
    cells = []
    for x, row in enumerate(state):
        for y, cell in enumerate(row):
        if cell == 0:
            cells.append([x, y])
    return cells
    def valid_move(x, y):
        if [x, y] in empty_cells(board):
            return True
        else
            return False
def set_move(x, y, player):
    if valid_move(x, y, player):
        board[x][y] = player
        return True
    else:
        return False
```

```
def minmax (state, depth, player):
    if player == COMP:
        best = [-1, -1, -infinity]
    else:
        best = [-1, -1, +infinity]
    if depth == 0 or game_over (state):
        score = evaluate (state)
        return [-1, -1, score]
    for cell in empty_cell (state):
        x, y = cell[0], cell[1]
        state[x][y] = 0
        score[0], score[1] = x, y
        if player == COMP:
            if score[2] > best[2]:
                best = score # max value
        else:
            if score[2] < best[2]:
                best = score # min value
        return best
    def clear ():
        os_name = platform.system ().lower ()
        if 'windows' in os_name:
            system ('cls')
        else:
            system ('clear')

    def render (state, c_choice, h_choice):
        chars = {
            -1: h_choice,
            +1: c_choice,
            0:
```



Date:

```
if wins(board, HUMAN)
    Clean()
    print(f"Human turn[{h_choice}]")
    render(board, c_choice, h_choice)
    print("YOU WIN")
elif wins(board, Comp):
    Clean()
    print(f"Computer turn[{c_choice}]")
    render(board, c_choice, h_choice)
    Print("YouLose")

else
    Clean()
    render(board, c_choice, h_choice)
    print("DRAW")
    exit()
if _name_ == "_main_"
    main()
```

Output:

```
Choose X or O
Chosen: O
first to start? [y/N]: y
HUMAN [O]

|   |   |   |          |   |   | O |
|   |   |   |          |   | X |   |
|   |   |   |          |   |   |   |

                       Draw
use numpad (1.9): 3    Human turn [O]
Computer turn [X]
                       | |   | O |
| |   | O |            | | X |   |
| |   |   |            | |   | O |
| |   |   |
```

Date :

	0
|x|x
	0

ur numpad : 4

		0
x	x	x
		0

Comp win
you lose

RESULT : Thus the output verified successfully