13/11/2020 AI - LAB - TEST - 1 REVANTH.R

1BM18CS082

5-B

Q) Implement Tic-tac-toe using 2-agent algorithm (Computer v/s Computer).

→ Tic Tac Toe (Computer vs Computer)

```
import numpy as np
import random
from time import sleep

# Creating an empty board
def create_board():
    return (np.array([[0, 0, 0],
                      [0, 0, 0],
                      [0, 0, 0]]))

# Check for empty places on board
def chances(board):
    l = []
    for i in range(len(board)):
        for j in range(len(board)):
            if board[i][j] == 0:
```


Revanth

REVANTH. R
1BM18CS082

```
l.append [ (i, j] ]
    return (l)
```

Selecting random place for player

```
def random_place ( board, player )
    selection = # chances (board)
    current_place = random.choice ( selection )
    board [ current_place ] = player
    return (board)
```

Checking whether the player has 3 of their marks in horizontal

```
def row_win ( board, player ) :
    for x in range (len (board)) :
        win = True
    for y in range (len( board )) :
        if board [x, y] != player :
            win = False
            continue
```

```
if win = True:
    return (win)        return (win)
```

Checking whether player has 3 of their marks in vertical row

```
def col_win (board, player):
    for x in range [len (board)]:
        win = True
    for y in range (len (board)):
        if board [y] [x] != player:
            win = False
            continue
    if win = True:
        return (win)        return (win)
```

Check whether player has 3 of marks in a diagonal row.

```
def diag_win (board, player):
    win = true          y = 0
    for x in range (len (board)):
```

```
if board [x, x] != player:
    win = False
if win:                    return win
    win = True
if win:                    for x in range (len (board)):
    y = len (board) - 1 - x
    if board [x, y] != player:
        win = False
    return win
# evaluating whether there is a winner or not is
the
    def evaluate (board):
        winner = 0
    for player in [1, 2]:
if (row_win (board, player) or
    col_win (board, player) or
        diag_win (board, player)):
    winner = player
```

```
if np.all(board != 0) and
        winner == 0:
            winner = -1
        return winner.

# Main function.
def play_game():
    board, winner, counter = create_board(), 0, 1
    print(board)          sleep(2)
    while winner == 0:
        for player in [1, 2]:
            board = random_place(board, player)
            print("Board after " + str(counter) + " move")
            print(board)          sleep(2)
            counter += 1          winner = evaluate(board)
            if winner != 0:       break
    return(winner)
print("Winner is: " + str(play_game()))
```