24/09/24

Tic Tac Toe

1) Write an algorithm

Step 1 :- Create a 2D array

Step 2 :- 2D array = 3x3 and initializing empty spaces '-'

⇒ Create 3 rows a 3 different lists [[ ], [ ], [ ]]

```
[[-, -, -]
 [-, -, -]
 [-, -, -]]
```

Step 3 :- Use of random function to choose first move.
The first move will always have X as start.

Display the board :

```
def print_board(board):
    for row in board:
        print (' '.join(row))
print ('_') for empty spaces.
```

Step 4 :- Check for a winner

⇒ 8 possibilities (3 rows, 3 col, 3 diagonals)

```
def check_winner(board):
for i in range(3):
    if board[i][0] == board[i][1] == board[i][2] != " "
        return board[i][0]
```

Similarly check for Col & diag

Step 5:- Player move :- The player should enter row & col index to make a move

```
def player_move(board):
    if board[row][col]=='-':
        board[row][col]='x'
        break
    else
        print("Try again.");
```

Step 6:- Computer move :-

if check for winning move

```
for i in range(3):
    for j in range(3):
        if board(i)[j]=='-':
            board[i][j]=='o';
        if check_winner(board)=='o';
            return
```

→ If no winning move, pick a random move such that player can't win.

```
import random
def print_board(board):
    for row in board:
        print("|".join(row))
        print("-"*9)

def check_winner(board):
    for i in range(3):
        if board[i][0]==
            return board
        if board[0][i] ==
            return board
    if board[0][0] ==boar
        return board[
    if board[0][2] == ba
        return boa
    return None
def is_full(board):
    return all(cell!=
def find_winning_
    for i in range(3
        for j in ra
            if board
                board
                if
```

```
import random

def print_board (board):
    for row in board:
    print ("|". join (row))
    print ("_"* 9)

def check_winner (board):
    for i in range (3):
        if board [i][0] == board (i)(i] == board (i)(2] ! = " ":
            return board (i)[0]

    if board [0] (i] == board [i](i] == board [2](i] ! = " ";
        return board [0] Ci]

if board [0](0] == board (1](1] = board [2](2] ! = " ":
    return board [0](0]

if board [0] [2] == board (1](1] == boad [2](0]. ! = " ":
    return bond (0) [2]

return None

def is_full (board):
    return all ( cell ( = " " for row in board for cell in row)

def find_winning_move (board, player):
    for i in range (3):
        for j in range (3):
            if board [i] (j] = = " ";
                boound [i](j] = = player
                if check_winner (board) = = player;
                    board (i) (j) = " "
                    return (i, j)
```

```
            board[i][j] = " "

    return None

def get_computer_move(board):
    move = find_winning_move(board, "O")
    if move:
        return move

    if board[1][1] == " ":
        return (1, 1)

    corners = [(0,0), (0,2), (2,0), (2,2)]
    for corner in corners:
        if board[corner[0]][corner[1]] == " ":

    for i in range(3):
        for j in range(3):
            if board[i][j] == " ":
                return (i, j)


def tic_tac_toe():
    board = [[" " for _ in range(3)] for _ in range(3)]
    current_player = "X"
    computer_player = "O"
    print("Player X goes first.")
```

```
while True :
    print_board (board)
    if current_player == "X":
        while True :
            try :
                row = int (input ( " Player X, enter the row (0-2): " )))
                col = int (input ("player X, enter the col (0-2): ")))
                if board [row] [col] == " ":
                    break
```

```
else:
    print (" Cell is already taken! Try again ")
except (ValueError, IndexError) :
    print (" Invalid input ! Please enter number b/w 0 & 2. ")
```

```
else :
    print (" Computer's turn .. ")
    row, col = get_computer_move (board)
    print (f" Computer chooses row {row}, col {col} ")

    board [row] [col] = current_player

    winner = check_winner (board)
    if winner :
        print_board (board)
        print (f" Player {winner} wins ! ")
        break
if is_full (board):
    print_board (board)
    print (" It's a tie ! ")
    break
```

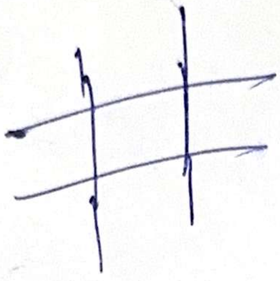
current_player = comput... / c...

if __name__ == "__main__":

    tic_tac_toe()

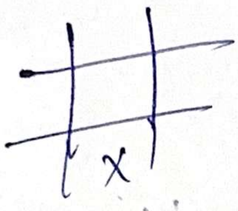
Output :-

player X goes first



player X, enter the row (0-2) : 2

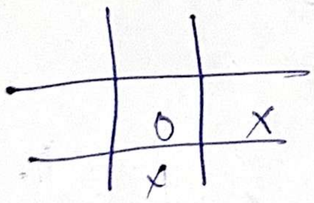
player X, enter the col (0-2) : 1



Computer's turn :

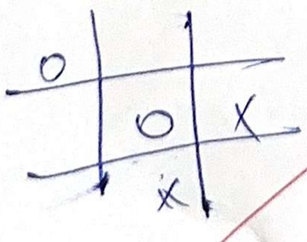
row 1, col = 1

player row = 1 col = 2



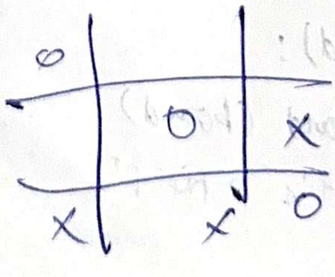
Computer turn row = 0, col = 0

from 24/9/24



player O wins

player X row 2 col = 0



Computer row = 2 col = 2