Date 24/01/24
Page

Algorithm for Tic Tac Toe :-

→ Player → ['x']
&
AI → ['o']

→ Analyse the current state of the board.

> Chance → Random
> Assuming only 9 boxes.
> Totally 8 match winning combinations {3×3 grid}
> random performance

→ Player need to place ['x'] randomly on the board.

→ After every move check for winning or draw
so, // i ranges (3)
def checkWin(board):
// mainly 3 rows, 3 columns, 2 diagonals.
if board[i][0] == board[i][1] == board[i][2] != " ":
return True;
if board[0][i] == board[1][i] == board[2][i] != " ":
return True;
if board[0][0] == board[1][1] == board[2][2] != " ":
return True;
if board[0][2] == board[1][1] == board[2][0] != " ":
return True

→ AI moves in following preference

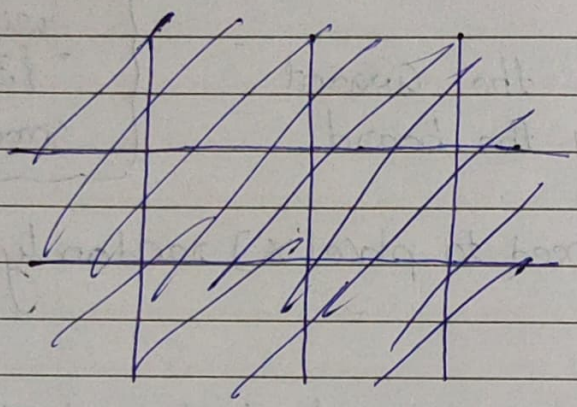
first AI checks if it can win
second to block the player winning.
Neither selecting random available cell, which may also lead to draw.

Reapeat.

Date / /
Page

→ def check_draw(board):

if is_board_full(board):
    // its draw
    break



{0,0}	{0,1}	{0,2}
{1,0}	{1,1}	{1,2}
{2,0}	{2,1}	{2,2}

Sub

O	O	X
	X	
X	O	

Date __/__/__
Page _____

//program code

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

def check_winner(board):
    lines = []
    lines.extend(board)
    lines.extend([[board[i][j] for i in range(3)]
                  for j in range(3)])
    lines.append([board[i][i] for i in range(3)])
    lines.append([board[i][2-i] for i in range(3)])

    for line in lines:
        if line[0] == line[1] == line[2] != "":
            return line[0]
    return None

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

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

def player_move(board):
    while True:
        try:
```

```
row = int(input("Enter row(0-2): "))
col = int(input("Enter column(0-2): "))
if board[row][col] == ' ':
    board[row][col] = 'X'
    break
else:
    print("Position already taken. Try again")
except(ValueError, IndexError):
    print("Invalid input")

def ai_move(board):
    for row, col in get_empty_positions(board):
        board[row][col] = 'O'
        if check_winner(board) == 'O':
            return
        board[row][col] = ' '

def main():
    board = [[' ' for _ in range(3)]
             for _ in range(3)]
    current_player = 'X'

while True:
    print_board(board)

    if current_player == 'X':
        player_move(board)
    else:
        ai_move(board)
```

Date __/__/__
Page _____

```
winner = check_winner(board)
if winner:
    print_board(board)
    print(f"{winner} wins!")
    break
elif is_board_full(board):
    print_board(board)
    print("It's a draw!")
    break

current_player = 'O' if current_player == 'X' else 'X'

if __name__ == "__main__":
    main()
```

Output:

```
 |  |
-----
 |  |
-----
 |  |
```

```
Enter row (0-2): 1
Enter column (0-2): 1
```

```
 |  |
-----
 | X |
-----
 |  |
```

```
 |  |
-----
 | X |
-----
O|  |
```

```
Enter row(0-2):0
Enter column(0-2):0

 x |  |
   |x |
 0 |  |

 x |  |
   |x |
 0 |  | 0

Enter row(0-2):2
Enter column(0-2):1

 x |0 |
   |x |
 0 |x | 0

Enter row(0-2):1
Enter column(0-2):2

 x |0 | x
 0 |x | 0
 0 |x | 0

Its a draw!
```