

TIC TAC TOE

21CSS101J – PROGRAMMING FOR PROBLEM SOLVING

Mini Project Report

Submitted by

Parvathy V Nair [Reg. No : RA2211003010295]

B.Tech. CSE - Core

Anyesha Biswas [Reg. No : RA2211003010298]

B.Tech. CSE - Core



**SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

S.R.M. NAGAR, KATTANKULATHUR – 603 203

KANCHEEPURAM DISTRICT

December 2022

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	Problem Statement	3
2.	Methodology / Procedure	4
3.	Coding (C or Python)	5
4.	Results	8
5.	Conclusion	9

PROBLEM STATEMENT:

Program a two-person game of tic-tac-toe played on a three-by-three board. each person has a marker, one has 'X' and other 'O'. Players alternate their turns to place the marker on the board. The first player to get three in a row either diagonally, horizontally, or vertically, wins the games. In the event all squares are taken on the board without a winner then it is a tie. Player one should be assigned an 'O' as their marker, player two should be assigned the 'X'. After the game has been completed, the program should congratulate the winner by name.

PROCEDURE / METHODOLOGY:

1. Create a board using a 2-dimensional array and initialize each element as empty.
2. You can represent empty using any symbol you like. Here, we are going to use a hyphen. '- '.
3. Write a function to check whether the board is filled or not.
4. Iterate over the board and return false if the board contains an empty sign or else return true.
5. Write a function to check whether a player has won or not.
6. We have to check all the possibilities that we discussed in the previous section.
7. Check for all the rows, columns, and two diagonals.
8. Write a function to show the board as we will show the board multiple times to the users while they are playing.
9. Write a function to start the game.
10. Select the first turn of the player randomly.
11. Write an infinite loop that breaks when the game is over (either win or draw).
12. Show the board to the user to select the spot for the next move.
13. Ask the user to enter the row and column number.
14. Update the spot with the respective player sign.
15. Check whether the current player won the game or not.
16. If the current player won the game, then print a winning message and break the infinite loop.
17. Next, check whether the board is filled or not.
If the board is filled, then print the draw message and break the infinite loop.
18. Finally, show the user the final view of the board.

CODING (PYTHON):

```
main.py +
1
2 import random
3
4
5 class TicTacToe:
6
7     def __init__(self):
8         self.board = []
9
10    def create_board(self):
11        for i in range(3):
12            row = []
13            for j in range(3):
14                row.append('-')
15            self.board.append(row)
```

Ln: 4, Col: 1

```
main.py +
17    def get_random_first_player(self):
18        return random.randint(0, 1)
19
20    def fix_spot(self, row, col, player):
21        self.board[row][col] = player
22
23    def is_player_win(self, player):
24        win = None
25
26        n = len(self.board)
27
28        # checking rows
29        for i in range(n):
30            win = True
31            for j in range(n):
```

Ln: 4, Col: 1

```
main.py +
31        for j in range(n):
32            if self.board[i][j] != player:
33                win = False
34                break
35        if win:
36            return win
37
38        # checking columns
39        for i in range(n):
40            win = True
41            for j in range(n):
42                if self.board[j][i] != player:
43                    win = False
44                    break
45        if win:
```

Ln: 4, Col: 1

```
main.py +
45     if win:
46         return win
47
48     # checking diagonals
49     win = True
50     for i in range(n):
51         if self.board[i][i] != player:
52             win = False
53             break
54     if win:
55         return win
56
57     win = True
58     for i in range(n):
59         if self.board[i][n - 1 - i] != player:
```

Ln: 4, Col: 1

```
main.py +
60         win = False
61         break
62     if win:
63         return win
64     return False
65
66     for row in self.board:
67         for item in row:
68             if item == '-':
69                 return False
70     return True
71
72     def is_board_filled(self):
73         for row in self.board:
74             for item in row:
```

Ln: 4, Col: 1

```
main.py +
74         for item in row:
75             if item == '-':
76                 return False
77     return True
78
79     def swap_player_turn(self, player):
80         return 'X' if player == 'O' else 'O'
81
82     def show_board(self):
83         for row in self.board:
84             for item in row:
85                 print(item, end=" ")
86             print()
87
88     def start(self):
```

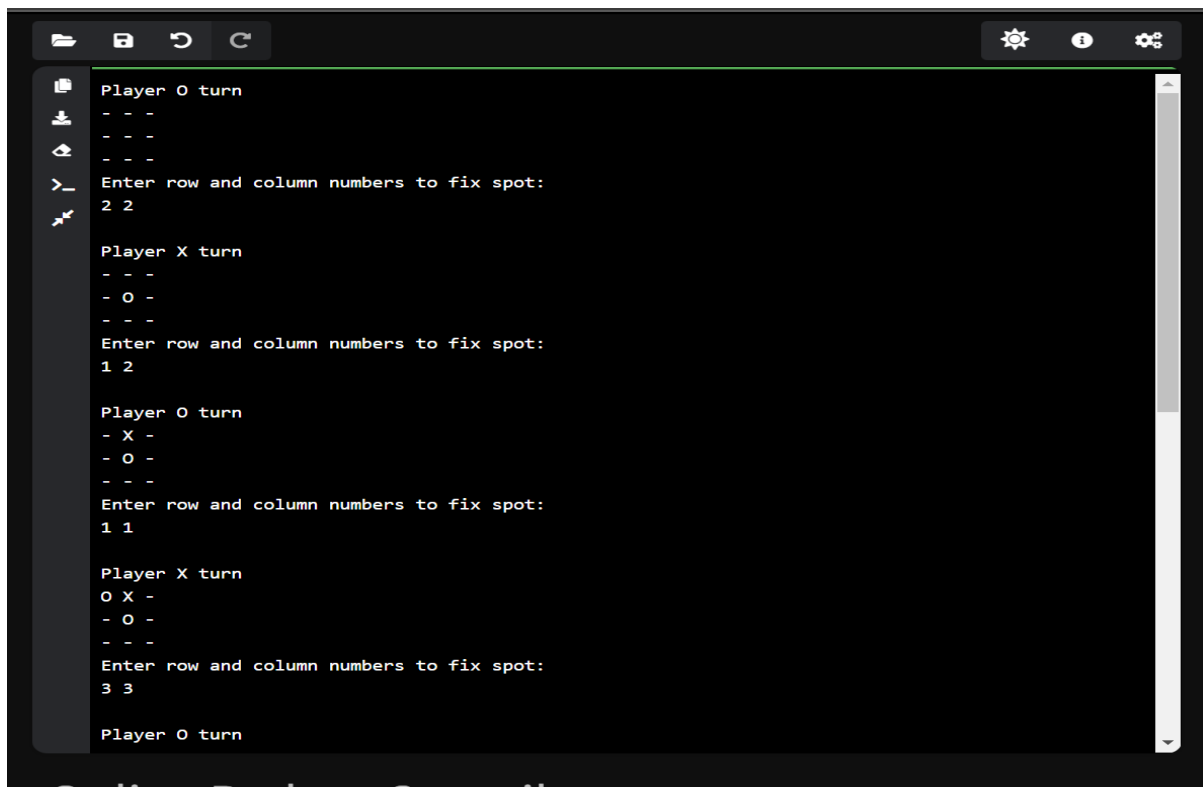
Ln: 4, Col: 1

```
main.py +
88 ~ def start(self):
89     self.create_board()
90
91     player = 'X' if self.get_random_first_player() == 1 else 'O'
92 ~ while True:
93     print(f"Player {player} turn")
94
95     self.show_board()
96
97     # taking user input
98     row, col = list(
99         map(int, input("Enter row and column numbers to fix spot: ").split()))
100     print()
101
102     # fixing the spot
Ln: 4, Col: 1
```

```
main.py +
103     self.fix_spot(row - 1, col - 1, player)
104
105     # checking whether current player is won or not
106 ~ if self.is_player_win(player):
107     print(f"Player {player} wins the game!")
108     break
109
110     # checking whether the game is draw or not
111 ~ if self.is_board_filled():
112     print("Match Draw!")
113     break
114
115     # swapping the turn
116     player = self.swap_player_turn(player)
117
Ln: 4, Col: 1
```

```
main.py +
113         break
114
115         # swapping the turn
116         player = self.swap_player_turn(player)
117
118         # showing the final view of board
119         print()
120         self.show_board()
121
122
123 # starting the game
124 tic_tac_toe = TicTacToe()
125 tic_tac_toe.start()
126
127
Ln: 4, Col: 1
```

RESULTS:



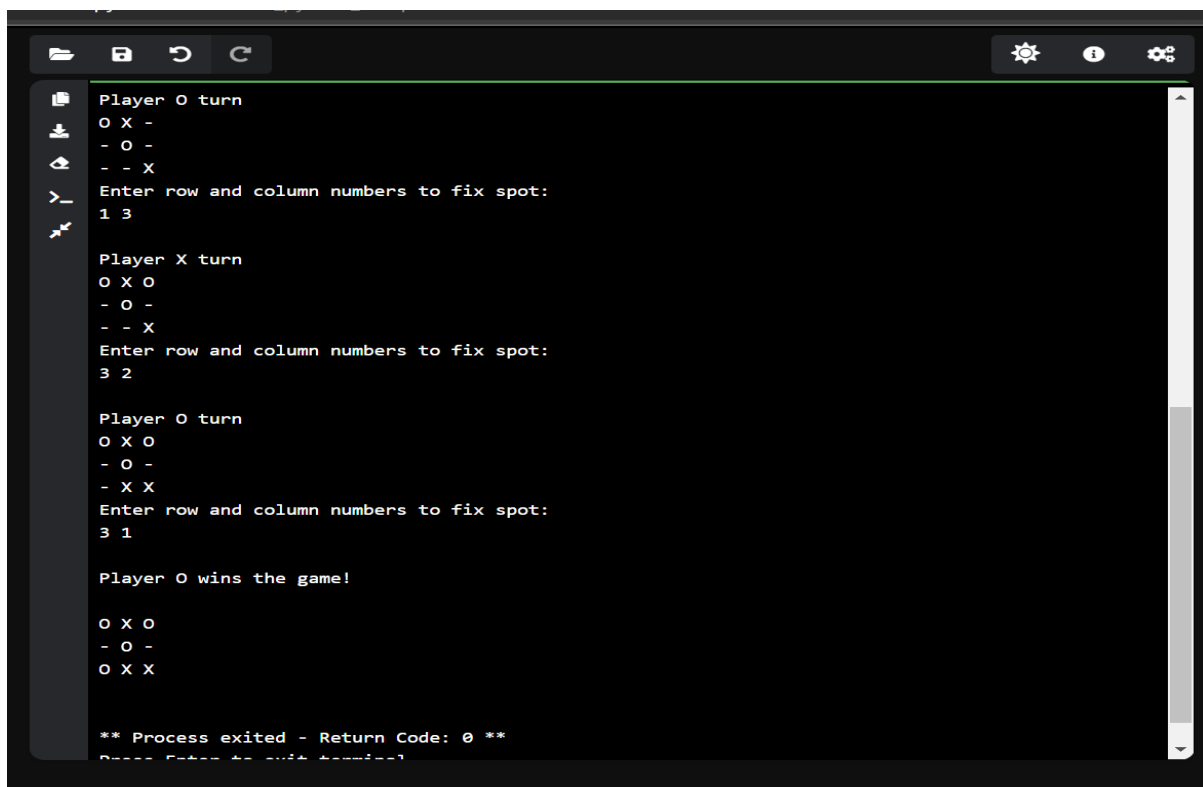
```
Player O turn
- - -
- - -
- - -
>_ Enter row and column numbers to fix spot:
2 2

Player X turn
- - -
- O -
- - -
>_ Enter row and column numbers to fix spot:
1 2

Player O turn
- X -
- O -
- - -
>_ Enter row and column numbers to fix spot:
1 1

Player X turn
O X -
- O -
- - -
>_ Enter row and column numbers to fix spot:
3 3

Player O turn
```



```
Player O turn
O X -
- O -
- - X
>_ Enter row and column numbers to fix spot:
1 3

Player X turn
O X O
- O -
- - X
>_ Enter row and column numbers to fix spot:
3 2

Player O turn
O X O
- O -
- X X
>_ Enter row and column numbers to fix spot:
3 1

Player O wins the game!

O X O
- O -
O X X

** Process exited - Return Code: 0 **
Press Enter to exit Terminal
```


CONCLUSION:

The Python Tic Tac Toe game is created using the basic features of Python. We have used basic if-else conditions along with function calls based on user input. In future, we can also add the names for the player. we can also keep track of the number of times the players play and the number of wins using a scoreboard. We can also change the markers from the typical 'X' and 'O' to the desired characters. We can also have a time limit for each turn. The board can be filled with a desired color. It can be enhanced by using animation and have different levels according to the difficulty.