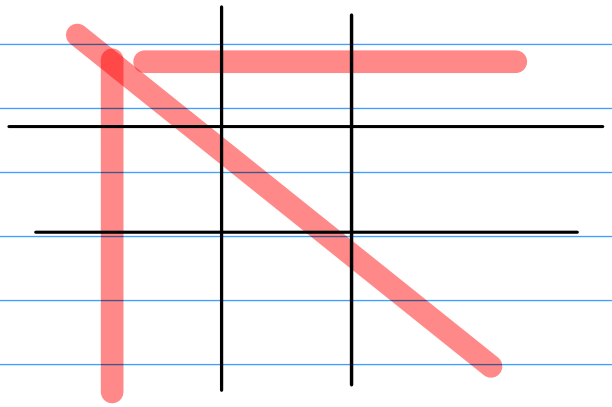
Game Building
Tic - tac - toe

Date
10th Nov 24

player 1 - X

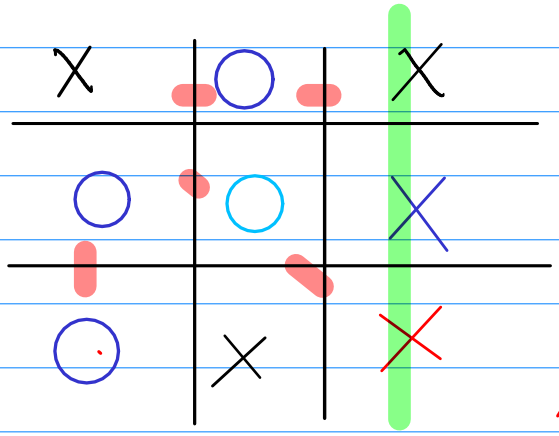
player 2 - O



Any complete row
— you win

Any complete col
- you win

1	2	3
4	5	6
7	8	9



X wins

Understanding the logic

→ Analyzing the flow of code

↳ Required functions

↓

Terminal

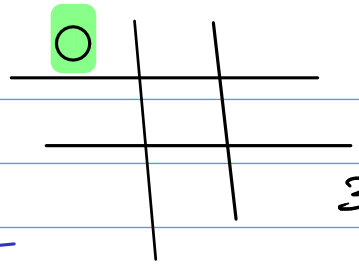
Second part

→ python libraries

↓

To make an interface for the game

Requirements

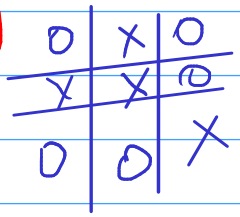


3x3 grid.

→ Board

→ valid location or not

→ Check winner ✓ (check_winner())



→ Board is full not

(is_board_full()) ↳ tie

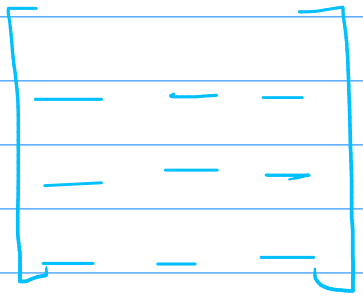
Setting the board

0 to 8

empty string

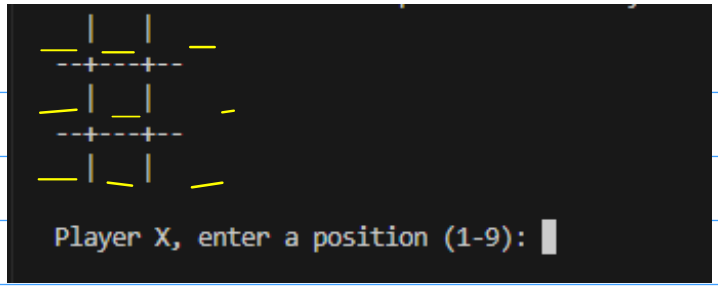
```
board = [" " for _ in range(9)]
```

↳ not using any variable, just running loop & assigning empty string.



```
 |   |
--+---+--
 |   |
--+---+--
 |   |

Player X, enter a position (1-9):
```




```
def print_board():
    print(f"{board[0]} | {board[1]} | {board[2]}")
    print("--+---+--")
    print(f"{board[3]} | {board[4]} | {board[5]}")
    print("--+---+--")
    print(f"{board[6]} | {board[7]} | {board[8]}")
    print()
```

name = "himanshu"

print ("Hello" , name)

o/p - Hello himanshu

print (f"Hello {name}")

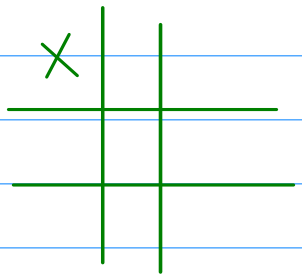
Input from user

move = 1 to 9

```
# Validate the move
    if board[move] == " ":
        board[move] = current_player
    else:
        print("This spot is already taken. Try again.")
        continue
```

board[1] == empty

board[1] = X

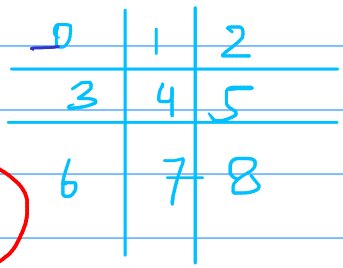



```
# Check for a win
    if check_winner(current_player):
        print_board()
        print(f"Player {current_player} wins!")
        break
```

Our logic

0 to 8

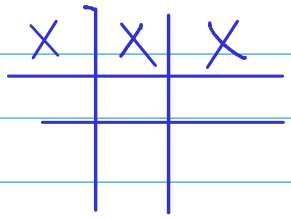
1 to 9




```
def check_winner(player):
    winning_combinations = [
        (0, 1, 2), (3, 4, 5), (6, 7, 8),  # rows
        (0, 3, 6), (1, 4, 7), (2, 5, 8),  # columns
        (0, 4, 8), (2, 4, 6)              # diagonals
    ]
    for combo in winning_combinations:
        if board[combo[0]] == board[combo[1]] == board[combo[2]] == player:
            return True
    return False
```

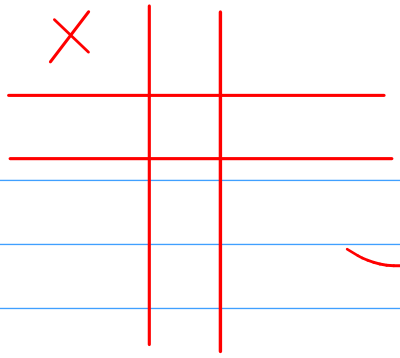
lists

board



false

0, 1, 2



is_board_full( )

↳ There is no place to enter the i/p., It's a tie

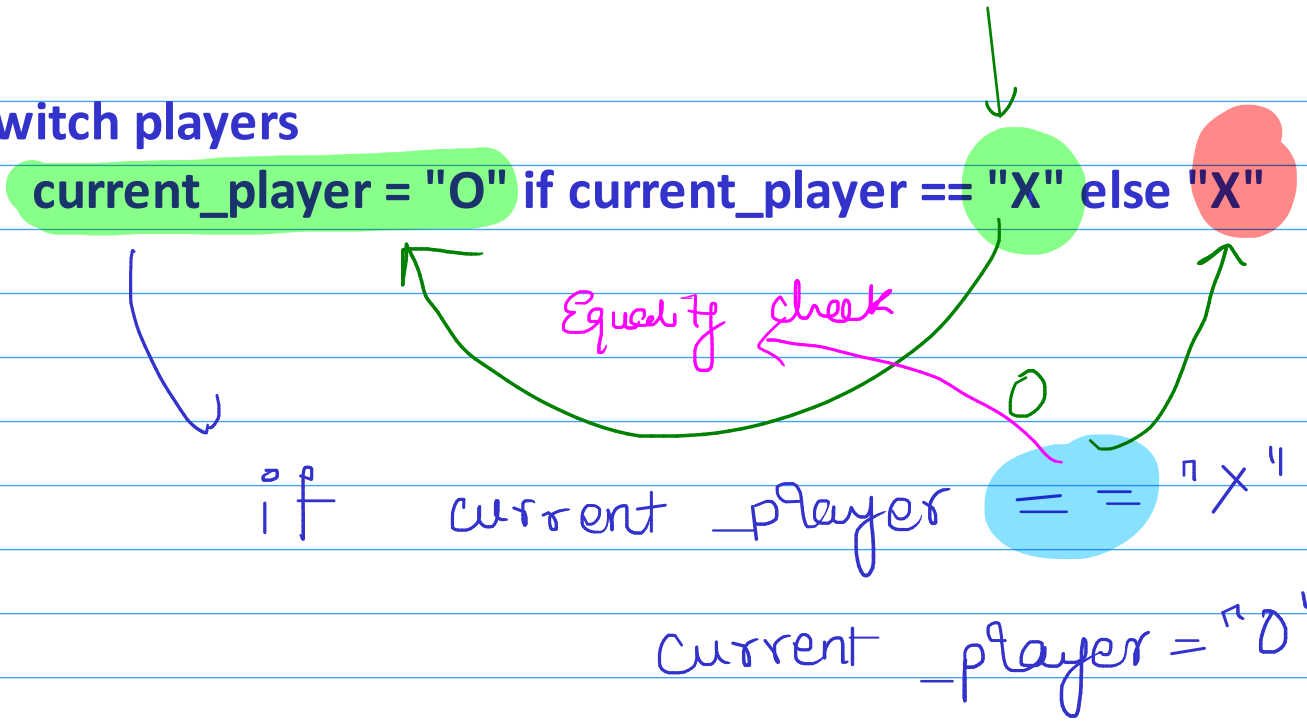
```
# Check for a tie
    if is_board_full():
        print_board()
        print("It's a tie!")
        break
```

True

```
def is_board_full():
    return " " not in board
```

if no empty string in the board, return true

```
# Switch players
current_player = "O" if current_player == "X" else "X"
```



list=[ ]
sets={ }
tuple=( )

[( ),]

( ) tuple

[ ]

```
def check_winner(player):
    winning_combinations = [
        (0, 1, 2), (3, 4, 5), (6, 7, 8),  # rows
        (0, 3, 6), (1, 4, 7), (2, 5, 8),  # columns
        (0, 4, 8), (2, 4, 6)              # diagonals
    ]
    for combo in winning_combinations:
        if board[combo[0]] == board[combo[1]] == board[combo[2]] == player:
            return True
    return False
```

0 1 2

if board[combo[0]] == "O"

4

if board[combo[1]] == "O"

6

if board[combo[2]] == "O"

X wins.

(0, 4, 8)

0 4 8