Workshop: Console Tic-Tac-Toe

In this workshop, we will create a simple two-player tic-tac-toe game. Here is how the game is going to look in the end:

```
Player one name: peter
Player two name: george
peter would you like to play with 'X' or 'O'? X
This is the numeration of the board:
|  1  |  2  |  3  |
|  4  |  5  |  6  |
|  7  |  8  |  9  |
peter starts first!
peter choose a free position [1-9]: 1
|  X  |     |     |
|     |     |     |
|     |     |     |
george choose a free position [1-9]: 5
|  X  |     |     |
|     |  O  |     |
|     |     |     |
peter choose a free position [1-9]:
```

```
peter choose a free position [1-9]: 2
|  X  |  X  |     |
|     |  O  |     |
|     |     |     |
george choose a free position [1-9]: 4
|  X  |  X  |     |
|  O  |  O  |     |
|     |     |     |
peter choose a free position [1-9]: 3
|  X  |  X  |  X  |
|  O  |  O  |     |
|     |     |     |
peter won!

Process finished with exit code 0
```

The Main Logic

Global Variables

The global variables will be **player_one**, **player_two**, **board** (the state of the game) and **loop** (boolean to check if the game should continue or not)

Implementation

Let us now create our **main** logic of the game

```
player_one = None
player_two = None
board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]
setup()
current = player_one
other = player_two
loop = True

while loop:
    play(current, board)
    current, other = other, current
```

- We create our **global** variables **player_one**, **player_two** (**None** to start with), **board** (empty to start with), and **loop** (game loop)
- We also create variables **current** and **other** (to **switch turns** of the players)
- We call a **setup()** function, which we will implement later (it should take the **info** of the **players** and **draw the initial state** of the board)
- We create a **while** loop to keep playing until a player wins
- In there, we call a function called **play()** which will take the **choice** of the current player and **apply** his choice **on the board**
- Finally, we **switch the players**, so in the next iteration, the other player should choose

Creating the Setup() Function

```
def setup():
    global player_one, player_two
    player_one_name = input("Player one name: ")
    player_two_name = input("Player two name: ")
    player_one_sign = input(f"{player_one_name} would you like to play with 'X' or 'O'? ")
    player_two_sign = 'X' if player_one_sign == 'O' else 'O'
    player_one = [player_one_name, player_one_sign]
    player_two = [player_two_name, player_two_sign]
    print("This is the numeration of the board:")
    print("|  1  |  2  |  3  |")
    print("|  4  |  5  |  6  |")
    print("|  7  |  8  |  9  |")
    print(f"{player_one_name} starts first!")
```

- We take the **names** of the two players
- Then we ask **player one** for his **sign** and set the sign of the **second player**
- We save the info in the global variables **player_one** and **player_two** as a **list** of their **names** and **signs**
- We display some **info** of the game **rules** and start with player one

Creating the Play() Function

Now, let us implement the **`play()`** function, which will ask the current player to choose his following action and apply his sign on the board

```
def play(current, board):
    choice = int(input(f"{current[0]} choose a free position [1-9]: "))
    row = ceil(choice / 3) - 1
    col = choice % 3 - 1
    board[row][col] = current[1]
    draw_board(board)
    check_if_won(current, board)
```

- Here we ask the player to choose a **free space** to apply his sign
- We create the variables **`row`** and **`col`**, which calculate the **row** and **col** of the selected **label** (don't forget to **import ceil** from the math library)
- Then we **apply** the sign of the current player on the board
- We call **two functions** which we will implement next:
    - **`draw_board(board)`** - loops through the board and draws its current state
    - **`check_if_won(current, board)`** - checks if the current player has won after choosing his action

Creating the Draw_board() Function

```
def draw_board(board):
    for row in board:
        print('|   ', end="")
        print('   |   '.join([str(x) for x in row]), end="")
        print('   |')
```

- Here we **loop** through each **row** in the board and print its **state**

Creating the Check_if_won() Function

```
def check_if_won(current, board):
    global loop
    first_row = all([x == current[1] for x in board[0]])
    second_row = all([x == current[1] for x in board[1]])
    third_row = all([x == current[1] for x in board[2]])
    first_column = all(x == current[1] for x in [board[0][0], board[1][0], board[2][0]])
    second_column = all(x == current[1] for x in [board[0][1], board[1][1], board[2][1]])
    third_column = all(x == current[1] for x in [board[0][2], board[1][2], board[2][2]])
    first_diagonal = all(x == current[1] for x in [board[0][0], board[1][1], board[2][2]])
    second_diagonal = all(x == current[1] for x in [board[2][0], board[1][1], board[0][2]])
    if any([first_row, second_row, third_row, first_column, second_column,
            third_column, first_diagonal, second_diagonal]):
        print(f"{current[0]} won!")
        loop = False
```

- In this function, we firstly use the **global loop** variable, because we will use it later
- Then we create a **boolean** variable for each **win condition**
- We then check if any of these conditions is `True` and if there are, we print that the **current player has won** and then **stop the loop** (we set the loop variable to `False`)

1. BONUS

- Try writing validation logic for:
  - The signs can only be "X" and "O"
  - The users can only choose from the numbers 1 to 9
  - The users can only choose a free space
- Try adding error messages for those validations