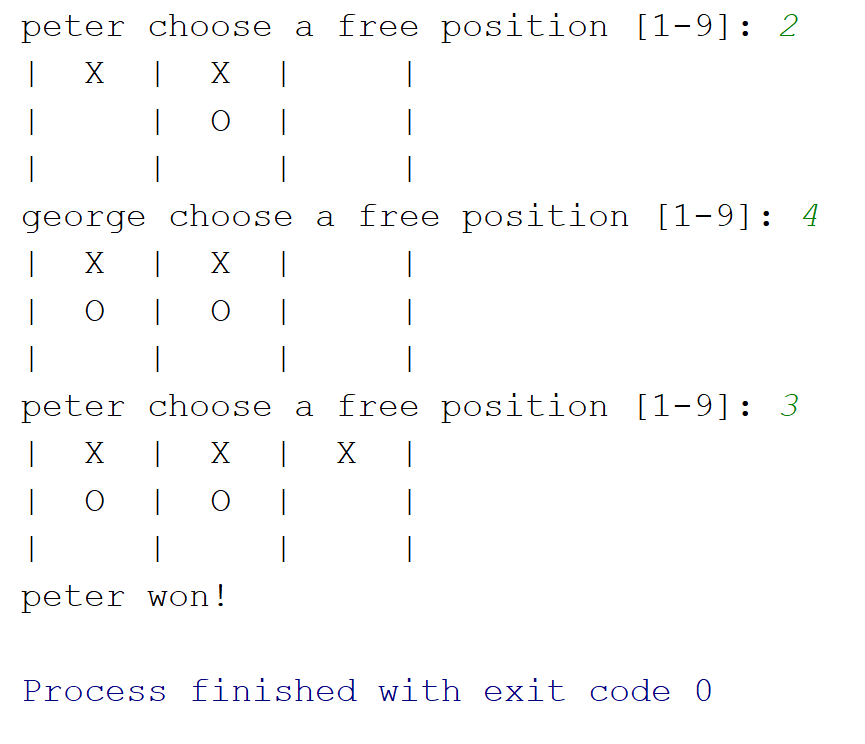
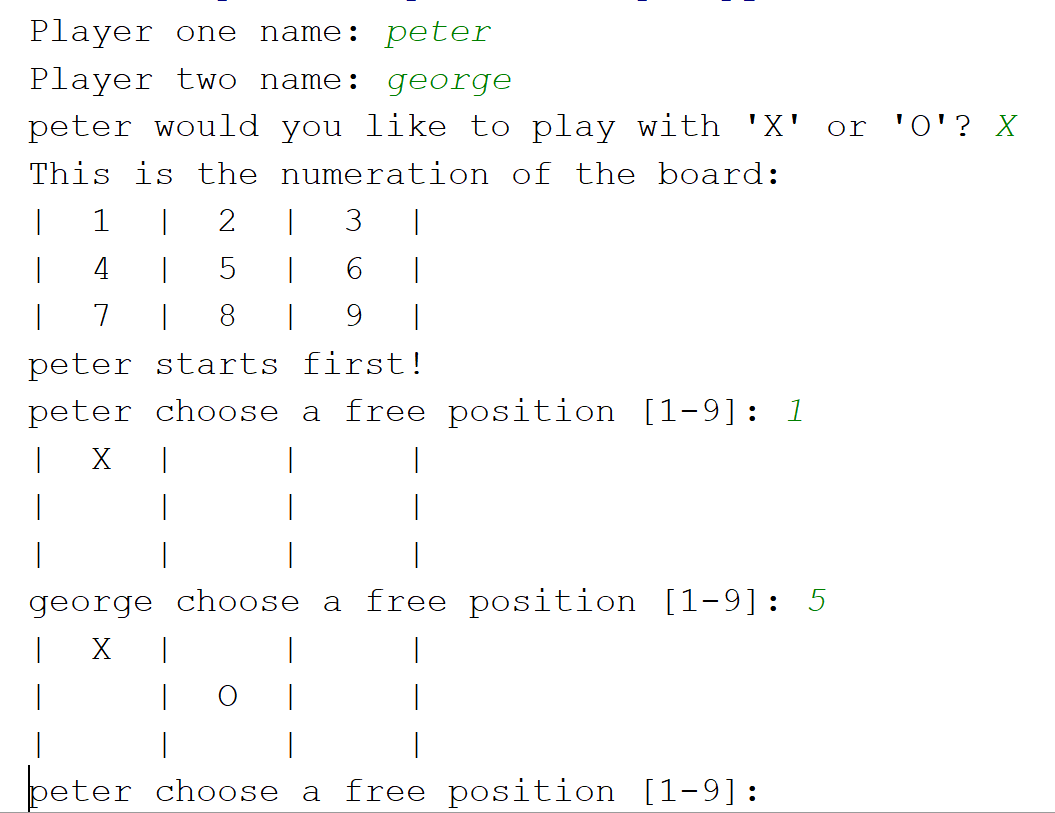
# Workshop: Console Tic-Tac-Toe

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:



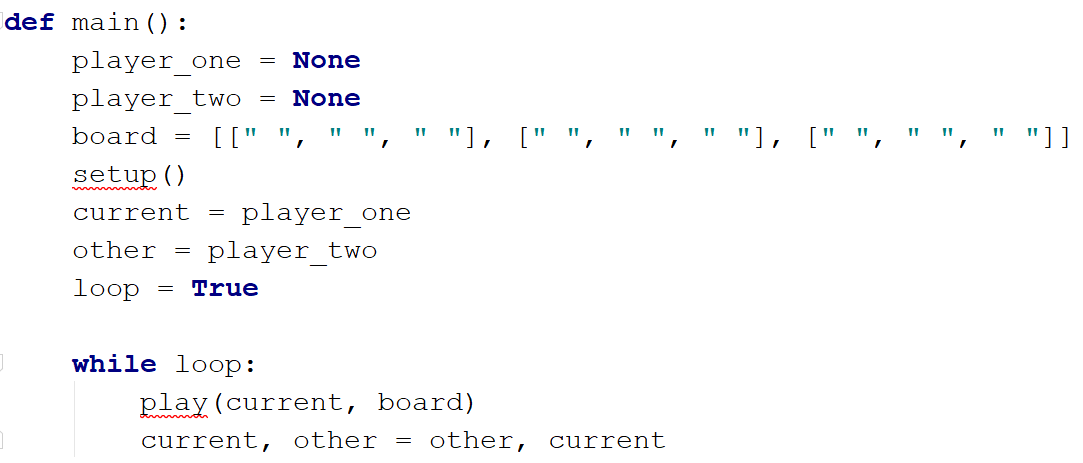
## The Main Logic

### 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)

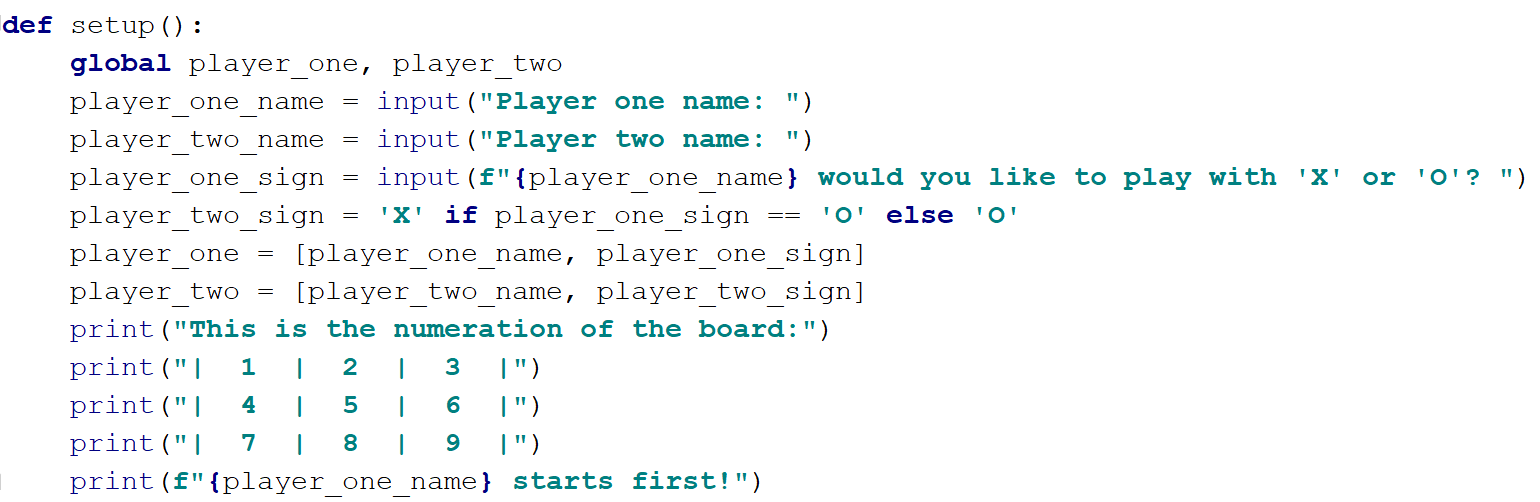
### Implementation

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



* 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 win
* In there, we call a function called **play()** which will take the **choice** of the current player and **apply** his/her choice **to the board**
* Finally, we **switch the players**, so in the next iteration, the other player should choose

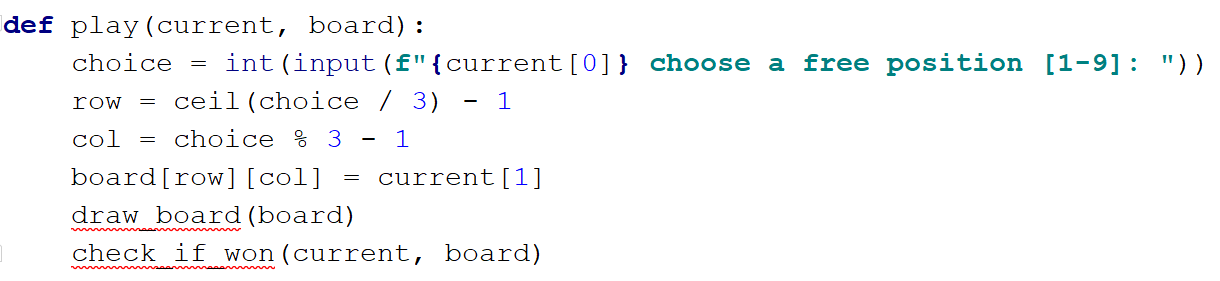
### Creating the Setup() Function



* 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** about the game **rules** and start with player one

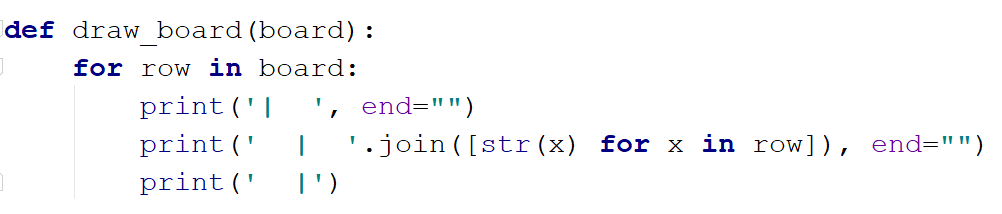
### Creating the Play() Function

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



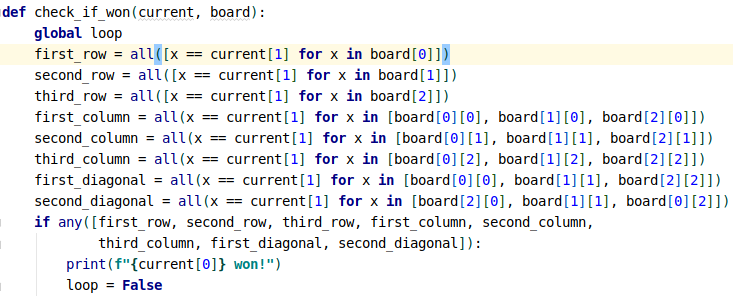
* Here we ask the player to choose a **free space** to apply his/her sign
* We create the variables **row** and **col,** which calculate the **row** and **col** of the selected **label** (don't forget to **import ceil** from the math library)
* Then we **apply** the sign of the current player on the 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



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

### Creating the Check\_if\_won() Function



* In this function, we first 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**)

## 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