USING **TKINTER** TO BUILD A TIC TAC TOE GAME WITH PYTHON

1. The Initial setup:

1. We basically use two classed to **handle the board** and to **handle the game logic** - **TicTacToeBoard** and **TicTacToeLogic** respectively.
2. We use **tkinter.label** to display the texts and images as well as to store and display the previous moves and all the available spaces
3. We use **tkinter.Button** to display the grid of the cells. when clicked by the user the game logic runs and checks for the winning condition
4. We extend the **tictactoeBoard** class with **Tkclass of tkinter** to create a window and insert the grid of cells with the label
5. **Title** --- displays the title of the game
6. We will use **\_cells**, initially an empty directory , to mark the positions of the buttons.
7. We use a frame object to hold the display of the game. Tk.Frame(master=self) This make the game window as the frames parent window.
8. .pack() function is used to place the frame on the main windows top border and its fill is set to tk.X to ensure the frame is filled even when the user resizes the window.
9. With the tk.label object the master is set to the current frame so that the label stays inside the frame. Then the text is shown indicating the game is ready to be played.
10. create\_grid(self) --- to create the grid of cells

--- creating a grid\_frame to hold all the cells

--- iterating the for loop thrice as we need to create a 3X3 grid consisting of buttons

--- updating the directory using buttons as keys and the row, col co-ordinates as the values and applying that to the entire grid.

2. SETTING UP THE LOGIC OF THE GAME :

* 1. we create two classes Player and Move. And we use NamedTuples imported from typing module
  2. Player class contains label--- X and O and color--- the target color of the element
  3. Move class contains row and col to position the targeted spot and label str defaulted to “” representing the move had been made yet.

---- the BOARD\_SIZE and DEFAULT\_PLAYERS(tuple containing the label and color of the players) are defined

* 1. In the TicTacToeLogic function we defined an initialization class that takes default players and board size as input.

----- it has the following variables :

1. Players--- set to a cyclic iteration of the default players
2. Boardsize
3. Current player – defined the current turn of the players
4. Winner combo- combination of the cells in the game that declared the winner
5. Winning\_combo --- combinations that define the winner in the game
6. Current\_moves --- players moves history
7. Has\_winner --- Boolean value that will be true when a winner is identified
   1. \_get\_winning\_combos function:

---- thi is to get all the possible winning combinations

--- there are 8 winning combination in TicTacToe

--- All the rows

--- all the columns

--- 2 diagnols

3.PROCESSING THE PLAYERS MOVES AND THE LOGIC OF THE GAME :

1.In this we –

* Write a function to toggle between the users
* Validate a player’s move
* Check for a winner
* Check for a tie

1. \_toggle\_user(): just self.\_current\_player=next(self.\_players)
2. Is\_valid\_move(self,move):

The move is valid based if: 1)game has no winner 2) the move is already empty i.e, the button is nor clicked yet.

1. Move\_process(): we pass the last move to this function to check if that move resulted a winner
   * + We iterated over the winning combinations that we previously set and create a set to store all the labels at the winning combos
     + If the result is of len 1 that means only one player had filled all the spaces in that combination
     + The winner is declared if the len(res) is one and there are no spaces in the set for that combination
2. Is\_tied(): defining the tie situation
   * + The game is tied when there is no winner and all the possible places are filled up
     + We used a generator expression to store all the labels in the buttons and use all() to check the sequence for gaps

4.PROCESSING THE PLAYERS MOVES ON THE GAME BOARD :