```
# Copy the text in this file and paste it into a tictactoe.py file to run it
################################################################################
#    TIC-TAC-TOE
#    In Python
#    Using the Tkinter Library for the GUI
#    By Krish Kochar
#
# Run this program with: $ python3 tictactoe.py
# Must have tkinter library on your system
################################################################################

import tkinter as tk
from tkinter import messagebox
from tkinter import font


class TicTacToe:
   # Constructor for game
   def __init__(self):
       # Initializes the window and all the required fields for the game logic
       self.root = tk.Tk()
       self.root.title("Tic Tac Toe")
       self.current_player = 'X'
       self.board = [[' ' for _ in range(3)] for _ in range(3)]
       self.buttons = [[None for _ in range(3)] for _ in range(3)]
       self.create_board()

   # create_board() initializes the window by creating the required buttons on the
board
   def create_board(self):
       button_font = font.Font(family='Helvetica', size=12, weight='bold')
       for i in range(3):
           for j in range(3):
               # lambda function used to ensure the button calls handle_click
               self.buttons[i][j] = tk.Button(self.root, text='', width=20, height=10,
command=lambda i=i, j=j: self.handle_click(i, j), font=button_font)
               self.buttons[i][j].grid(row=i, column=j)

   # handle_click(row, col) is called whenever a button is pressed on the GUI
   def handle_click(self, row, col):
       if self.board[row][col] == ' ':
```

```python
            self.board[row][col] = self.current_player
            self.buttons[row][col].config(text=self.current_player)
            if self.check_winner():
                messagebox.showinfo("You win!", f"Player {self.current_player} wins!")
                self.reset_game()
            elif self.is_board_full():
                messagebox.showinfo("It's a tie!", "It's a tie!")
                self.reset_game()
            else:
                self.current_player = 'O' if self.current_player == 'X' else 'X'

    # check_winner() checks the win conditions after every turn played
    def check_winner(self):
        # Check rows for win conditions
        for i in range(3):
            if self.board[i][0] == self.board[i][1] == self.board[i][2] != ' ':
                return True
        # Check columns for win conditions
        for i in range(3):
            if self.board[0][i] == self.board[1][i] == self.board[2][i] != ' ':
                return True
        # Check diagonals for win conditions
        if self.board[0][0] == self.board[1][1] == self.board[2][2] != ' ':
            return True
        if self.board[0][2] == self.board[1][1] == self.board[2][0] != ' ':
            return True
        return False

    # is_board_full checks if the board is full for a tie
    def is_board_full(self):
        for row in self.board:
            for cell in row:
                if cell == ' ':
                    return False
        return True

    # reset_game() resets the board to empty
    def reset_game(self):
        self.current_player = 'X'
        self.board = [[' ' for _ in range(3)] for _ in range(3)]
        for i in range(3):
            for j in range(3):
                self.buttons[i][j].config(text='')

    # run() is called by main and runs the tkinter window
    def run(self):
        self.root.mainloop()


if __name__ == "__main__":
    game = TicTacToe()
    game.run()
```