Code -

```
import tkinter as tk
from tkinter import messagebox

class TicTacToe(tk.Tk):
    def __init__(self):
        super().__init__()

        self.title("Tic Tac Toe")
        self.geometry("300x300")

        self.current_player = "X"
        self.board = [["" for a in range(3)] for a in range(3)]

        self.create_board_buttons()

    def create_board_buttons(self):
        self.buttons = [[None for b in range(3)] for b in range(3)]

        for i in range(3):
            for j in range(3):
                self.buttons[i][j] = tk.Button(self, text="", font=("Arial", 20), width=5, height=2,
                                                command=lambda row=i, col=j: self.on_button_click(row,
col))
                self.buttons[i][j].grid(row=i, column=j)

    def on_button_click(self, row, col):
        if self.board[row][col] == "":
            self.board[row][col] = self.current_player
            self.buttons[row][col].config(text=self.current_player)
            if self.check_winner():
                messagebox.showinfo("Winner!", f"Player {self.current_player} wins!")
                self.reset_game()
            elif self.check_draw():
                messagebox.showinfo("Draw!", "It's a draw!")
                self.reset_game()
            else:
                self.current_player = "O" if self.current_player == "X" else "X"

    def check_winner(self):
        for i in range(3):
            if self.board[i][0] == self.board[i][1] == self.board[i][2] != "":
                return True
            if self.board[0][i] == self.board[1][i] == self.board[2][i] != "":
                return True
        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

    def check_draw(self):
        for row in self.board:
            for cell in row:
                if cell == "":
                    return False
        return True

    def reset_game(self):
        for i in range(3):
            for j in range(3):
                self.board[i][j] = ""
                self.buttons[i][j].config(text="")
        self.current_player = "X"

app = TicTacToe()
app.mainloop()
```

Let's break down the code step by step:

1. Importing Libraries:
- `import tkinter as tk`: Imports the Tkinter library and provides an alias `tk` for easy reference.
- `from tkinter import messagebox`: Imports the `messagebox` module from Tkinter, which is used to display message boxes.

2. Creating the `TicTacToe` Class:
- `class TicTacToe(tk.Tk):`: Defines a class named `TicTacToe` that inherits from `tk.Tk`, making it a subclass of the Tkinter application window.

3. Constructor Method (`__init__`):
- `def __init__(self):`: Initializes the `TicTacToe` class.
- `super().__init__()`: Calls the constructor of the superclass (`tk.Tk`), initializing the Tkinter application window.
- `self.title("Tic Tac Toe")`: Sets the title of the window to "Tic Tac Toe".
- `self.geometry("300x300")`: Sets the size of the window to 300x300 pixels.

- `self.current_player = "X"`: Initializes the current player to "X".
- `self.board = [["" for a in range(3)] for a in range(3)]`: Initializes the game board as a 3x3 list filled with empty strings.

4. Creating Board Buttons (`create_board_buttons`):
- `def create_board_buttons(self):`: Defines a method to create buttons for the Tic Tac Toe board.
- `self.buttons = [[None for b in range(3)] for b in range(3)]`: Initializes a 2D list to hold references to the buttons.
- Nested loops iterate through each cell of the 3x3 grid.
- For each cell, a Tkinter `Button` widget is created with appropriate properties (text, font, width, height, and command). The `lambda` function is used to pass the row and column indices to the `on_button_click` method when the button is clicked.
- The button is then placed in the grid layout using the `grid` method.

5. Handling Button Clicks (`on_button_click`):
- `def on_button_click(self, row, col):`: Defines a method to handle button clicks.
- Checks if the clicked cell on the board is empty.
- If the cell is empty, updates the board with the current player's symbol ("X" or "O") and updates the button text accordingly.
- Checks for a winner or a draw using the `check_winner` and `check_draw` methods.
- If there is a winner, displays a message box indicating the winning player and resets the game. If it's a draw, displays a draw message and resets the game. Otherwise, switches to the next player.

6. Checking for a Winner (`check_winner`):
- `def check_winner(self):`: Defines a method to check if there's a winner.
- Iterates through rows, columns, and diagonals to check if any player has three symbols in a row.
- Returns `True` if there's a winner, otherwise `False`.

7. Checking for a Draw (`check_draw`):
- `def check_draw(self):`: Defines a method to check if the game is a draw.
- Checks if there are any empty cells left on the board.
- Returns `True` if it's a draw (no empty cells), otherwise `False`.

8. Resetting the Game (`reset_game`):
- `def reset_game(self):`: Defines a method to reset the game.
- Resets the game board and button texts to empty strings.
- Resets the current player to "X".

9. Creating an Instance of `TicTacToe` and Running the Main Loop:
   - `app = TicTacToe()`: Creates an instance of the `TicTacToe` class.
   - `app.mainloop()`: Starts the Tkinter event loop, allowing the application to handle user interactions and events.

This code creates a simple Tic Tac Toe game with a graphical interface using Tkinter, where players can take turns clicking on the board to make their moves. The game will determine the winner or declare a draw when appropriate.