Code -

```
import tkinter as tk

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

        self.title("Simple Calculator")
        self.geometry("250x300")

        self.expression = ""
        self.create_widgets()

    def create_widgets(self):
        self.result_var = tk.StringVar()

        entry = tk.Entry(self, textvariable=self.result_var, font=("Arial", 16), justify="right")
        entry.grid(row=0, column=0, columnspan=4, padx=10, pady=10)

        buttons = [
            ("7", 1, 0), ("8", 1, 1), ("9", 1, 2), ("/", 1, 3),
            ("4", 2, 0), ("5", 2, 1), ("6", 2, 2), ("*", 2, 3),
            ("1", 3, 0), ("2", 3, 1), ("3", 3, 2), ("-", 3, 3),
            ("0", 4, 0), (".", 4, 1), ("=", 4, 2), ("+", 4, 3)
        ]

        for (text, row, column) in buttons:
            button = tk.Button(self, text=text, font=("Arial", 16), width=4, height=2,
                               command=lambda t=text: self.on_button_click(t))
            button.grid(row=row, column=column, padx=5, pady=5)

    def on_button_click(self, value):
        if value == "=":
            try:
                self.expression = str(eval(self.expression))
            except Exception as e:
                self.expression = "Error"
        else:
            self.expression += value

        self.result_var.set(self.expression)
app = Calculator()
app.mainloop()
```

Let's go through the code step by step:

1. Importing Libraries:
- `import tkinter as tk`: Imports the Tkinter library and provides an alias `tk` for easy reference.

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

3. Constructor Method (`__init__`):
- `def __init__(self):`: Initializes the `Calculator` class.
- `super().__init__()`: Calls the constructor of the superclass (`tk.Tk`), initializing the Tkinter application window.
- `self.title("Simple Calculator")`: Sets the title of the window to "Simple Calculator".
- `self.geometry("250x300")`: Sets the size of the window to 250x300 pixels.
- `self.expression = ""`: Initializes the expression to an empty string.

4. Creating Widgets (`create_widgets`):
- `def create_widgets(self):`: Defines a method to create widgets (UI elements) for the calculator.
- `self.result_var = tk.StringVar()`: Initializes a `StringVar` object to hold the result of the calculation.
- Creates an `Entry` widget (`entry`) for displaying the expression and result. The `textvariable` option is set to `self.result_var`, and the font and alignment are configured.
- Creates buttons for digits (0-9), decimal point (.), mathematical operations (+, -, *, /), and equals (=) button.
- Buttons are organized in a grid layout using the `grid` method.

5. Handling Button Clicks (`on_button_click`):
- `def on_button_click(self, value):`: Defines a method to handle button clicks.
- If the clicked button is "=", the expression is evaluated using the `eval` function and the result is stored in `self.expression`. Any errors encountered during evaluation are caught and "Error" is assigned to `self.expression`.
- If the clicked button is not "=", the corresponding value (digit or operation) is appended to the expression (`self.expression`).
- The result (expression) is displayed in the entry widget using `self.result_var.set()`.

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

This code creates a simple calculator with a graphical interface using Tkinter. Users can click on buttons to input digits and mathematical operations, and the calculator will evaluate the expression when the "=" button is clicked. The result is displayed in a text entry widget.