Worksheet: Building a Basic Calculator Using Tkinter

In this lab, you will create a basic calculator using 's `tkinter` GUI module. The calculator will have a grid layout for the buttons and a display area. Follow the steps below to build the calculator step by step.

Prerequisites:

- Basic knowledge of functions and variables.
- Familiarity with loops and conditional statements.
- Basic understanding of GUI programming with `tkinter` (if not, a brief introduction is included).

Step 1: Setting Up the `tkinter` Window

First, you need to set up the `tkinter` window, which will act as the main calculator interface.

1. **Create a New File:**
   - Open your editor (e.g., Spider, PyCharm, VSCode, or IDLE).
   - Create a new file and name it `calculator.py`.
2. **Import the `tkinter` Module:**
   - Add the following line to your file to import `tkinter`:

```
import tkinter as tk
```

3. **Create the Main Window:**
   - Initialize the main `tkinter` window using the code below. This window will serve as the calculator's interface.

```
root = tk.Tk()
root.title("Basic Calculator")
```

4. **Run the Application:**
   - End the script with `root.mainloop()` to keep the window open.

```
root.mainloop()
```

Checkpoint:

Run your program. A blank window with the title "Basic Calculator" should appear.

Step 2: Creating the Display Area

Next, you will create a display area where the numbers and results are shown.

1. **Create a StringVar to Hold the Input/Output:**
   - Use a `StringVar()` to manage the input and output text on the calculator screen.

```
screen_var = tk.StringVar()
screen_var.set("")
```

2. **Add an Entry Widget for the Display:**
   - Use an `Entry` widget to display the current input and results.

```
screen = tk.Entry(root, textvar=screen_var, font="Arial 20 bold",
borderwidth=4, relief="ridge")
screen.grid(row=0, column=0, columnspan=4, ipadx=10, ipady=10)
```

Checkpoint:

When you run the program, a text area should appear at the top of the window.

Step 3: Adding Calculator Buttons

Now, you will add the buttons to the calculator.

1. **Create a List of Buttons:**
   - Make a list of the button labels you need for the calculator.

```
buttons = [
    "7", "8", "9", "/",
    "4", "5", "6", "*",
```

```
    "1", "2", "3", "-",
    "C", "0", "=", "+"
]
```

2. **Create Buttons in a Grid Layout:**
    - Use a loop to create and place buttons in the window with the `grid()` layout method.

```
row_val = 1
col_val = 0

for button_text in buttons:
    button = tk.Button(root, text=button_text, font="Arial 18", padx=20,
pady=20)
    button.grid(row=row_val, column=col_val, padx=5, pady=5)
    col_val += 1
    if col_val > 3:
        col_val = 0
        row_val += 1
```

Checkpoint:

When you run the program, buttons should appear in a grid layout. The buttons will not yet function.

Step 4: Handling Button Clicks

To make the calculator work, you need to handle what happens when a button is clicked.

1. **Define a Function to Handle Click Events:**
    - Create a function called `click()` that will take care of what happens when a button is clicked.

```
def click(event):
    text = event.widget.cget("text")
    if text == "=":
        try:
            expression = screen_var.get()
            result = eval(expression)
            screen_var.set(result)
        except Exception as e:
            screen_var.set("Error")
    elif text == "C":
        screen_var.set("")
    else:
```

```
            current = screen_var.get()
            screen_var.set(current + text)
```

2. **Bind Click Events to Buttons:**
    - Now bind the click function to each button so that when a button is clicked, the `click()` function is called.

```
button.bind("<Button-1>", click)
```

Checkpoint:

Run the program. Now the buttons should respond when clicked. You can enter numbers and perform simple calculations like addition, subtraction, multiplication, and division.

Step 5: Testing and Improving the Calculator

1. **Test the Calculator:**
    - Enter numbers and perform basic operations (+, -, *, /). Ensure that the clear button (`C`) works and that the equal button (`=`) calculates the result correctly.
2. **Error Handling:**
    - If you encounter any errors (e.g., division by zero), ensure that your program displays "Error" in the text area, as implemented in the `click()` function.

Challenge: Additional Features (Optional)

If you have completed the calculator and want to further improve it, here are some additional features you can implement:

- Add a decimal point button for floating-point operations.
- Add keyboard support for entering numbers and operations using the keyboard.
- Implement a backspace button to delete the last digit entered.

Final Code

After completing all the steps, your final code should look something like this:

```
import tkinter as tk
```

```
def click(event):
    text = event.widget.cget("text")
    if text == "=":
        try:
            expression = screen_var.get()
            result = eval(expression)
            screen_var.set(result)
        except Exception as e:
            screen_var.set("Error")
    elif text == "C":
        screen_var.set("")
    else:
        current = screen_var.get()
        screen_var.set(current + text)

root = tk.Tk()
root.title("Basic Calculator")

screen_var = tk.StringVar()
screen_var.set("")

screen = tk.Entry(root, textvar=screen_var, font="Arial 20 bold",
borderwidth=4, relief="ridge")
screen.grid(row=0, column=0, columnspan=4, ipadx=10, ipady=10)

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

row_val = 1
col_val = 0

for button_text in buttons:
    button = tk.Button(root, text=button_text, font="Arial 18", padx=20,
pady=20)
    button.grid(row=row_val, column=col_val, padx=5, pady=5)
    button.bind("<Button-1>", click)
    col_val += 1
    if col_val > 3:
        col_val = 0
        row_val += 1

root.mainloop()
```

Submission:

Once you have completed the calculator, take a screenshot of your working program and submit it along with your code.