Calculator GUI build using Python Tkinter

Requirements:

User must have python install in their computer to successfully run the calculator app. The calculator app consists of 4 most basic functions: add, subtract, multiply, and divide. In the folder, the user with see the manual guide, Calculator Icon and the classCalculator.py file. Run the classCalculator.py file to use the app.

Steps to run: Please Extract the files before. Click on the classCalculator.py file to run.

Alternative: In case the file does not work. Right click on the classCalculator.py file. Choose Edit with IDLE. On the menu bar, click on Run, and then Run Module as shown in Figure 1. Once you see the app shown in Figure 2, you have successfully open the app.

Graphical user interface, text, application

Description automatically generated

**Figure 1: Running classCalculator.py in IDLE.**

A picture containing text, light, furniture

Description automatically generated

**Figure 2: CalculatorApp**

1. **Overall Description of the Project**

The project is a calculator app that can do 4 most basic operations. The project is built using OOP and the main class is calculator. The most basic calculator is the main class and people can use inheritance to develop more operations to make more advance calculators like scientific calculator or graphic calculator.

It can be use in the same way as normal calculators’ work. The design and functionality are very similar to window’s calculator app except that this calculator can only do 4 most basic operations but can be further developed to create very advance calculators.

I have used several numbers and calculations to test the calculator app. I have also put Zero Division Error check in the division operator. As a result, calculator will show an error message if you divide by Zero.

My first test case is 1/0. The error message shown below.

A picture containing text, furniture, white, light

Description automatically generated

**Figure 3: Division by Zero Error**

1. **Components**

The working status of the calculator is fully functional. The calculator is also coded using OOP and have class Calculator. As a result, the code is flexible and anybody who wants to further develop can also use Calculator class and its methods to create more advance Calculator class.

In the application, the operations are:

+ – adding two numbers

- – subtracting two numbers

x – multiplying two numbers

/ – Dividing two numbers

= - Calculate and show result

Clear – clear

The application also has number buttons from 0-9 that represents the value of the two numbers.

To simply run choose a first number using the number buttons and choose one of the 4 operations buttons. Then choose another number using number buttons. If you wish to calculate more than 2 numbers, you can keep choosing another operation and continue the steps. To display the value calculated click on =.

For example: You wish to calculate 4+2+6

Click on 4, click on +, click on 2, click on +, click on 6, click on =.

1. **Modularity**

Some of the variables and functions of Calculator class are explain below:

**Creating new object =** takes in one argument string, that will be the name of the calculator.

**Self.window** = Tkinter’s window

**Self.button.**(from 0-9, add, subtract, multiply, divide, equal, clear) = fixed button size by the developer.

**Self.place(button, row number, column number, columnspan) =** Place buttons in the app

**Button\_click(number) =** Collects user input and add the numbers into a string.

**Button\_add(), Button\_multiply(), Button\_divide(), Button\_subtract()** – Save the first number input to the variable fnum. Then set math into operation accordingly.

**Button\_equal()** – Make a decision base on the math. Takes a second number and do calculation base on the math sign.

**Button\_clear()** – Clear

**Self.e** = Entry box, Also the result showing bar.

**run()** – run Tk module from the class.

**Changes in the future**

More operations can be add such as square root and more variables can be added such as pi. The four basic operations are not going to change.

1. **Extensibility**

More buttons can be created by the user. The place() function takes in button, row, column, and column span. The calculator right now has a column span of 3. More buttons can be manually code similar to the structure in the code.

**self.button\_1 = Button(self.window, text="1", padx = 40, pady = 20, command = lambda: self.button\_click(1))**

To create a button the user must create a function accordingly. The function can be place after the lambda function where **self.button\_click(1)** is. Then use the built in function place() to place in the grid.

Example – self.place(ButtonExample, row, column, columnspan)

The currently operations and the number pad are all statically bound since they should not be edited. For example, all calculators should have basic operations and number from 0-9.

The user can dynamically bound more variables and functions such as pi, sin, cos, tan since they are not required for all basic calculators.

The place() is dynamically bound since user can use to create number buttons and use this method to place it in the app.

The Entry Box/Display is statically bound since calculators also need to show result.

1. **Reusability**

The calculator is built using OOP. There are methods such as place() that allows users to not have to code to place the buttons but just type in row, column and columnspan to place their new buttons.

All number/operators can be reuse so new user only have to develop more buttons and not have to work on the whole basic calculator.

The class also comes with an argument of title so user can type in whatever name they want to give such as Scientific Calculator.

Since this is the most basic calculator, in the future user may want to develop scientific/graphic calculator. All the basic number pads and operations are implanted as well as the clear and equal sign. User may only need to develop new operations and user the basic functions they already have.

**Challenges/Course**

**I did have a challenging week in during the course where I did not turn in Memory Management assignment. If I earn extra credit, I would greatly appreciate it if it would cover the credit for my missing assignment week.**