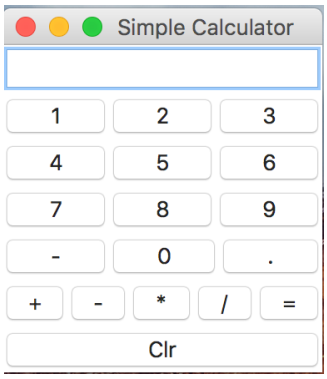
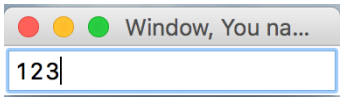
CS8 Calculator Lab

The objective of this lab is to build a working calculator. In doing so, I hope to introduce some of the basics about ***tkinter*****,** Python's de-facto standard GUI (Graphical User Interface) package. I suggest that you read chapter 13 in the text to get a fuller picture of what can be done with tkinter.

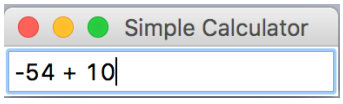


A. Open **gui1.py** in Wing101. This small program illustrates how to create and display a **Window** and assign a title that is placed in the window's title-bar (lines 6-9). In addition, an **Entry** widget is placed inside the window and a variable, ***self.display,*** is defined to hold the contents (lines 11-14). **Widgets** are things that can be placed inside a window or into containers in general. An **Entry** widget permits a user to enter data in it for processing. Finally, the main() function, uses a constructor to create a new **GUI1** object (lines 20-21).

   As illustrated below, the number 123 is entered into the Entry widget called ***entrybox*** and stored in the variable **self.display** after gui1.py is executed.

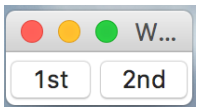


   Create a new program, in Wing101, called **simplecalculator.py**. Copy the code from **gui1.py** and make the following adjustments, make the title-bar contain the string "Simple Calculator", name the class Calculator, change the main() to create a new Calculator object.

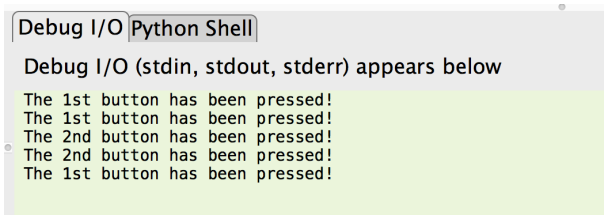


B. Open **gui2.py** in Wing101. This small program illustrates how to group together widgets in a container called a **Frame** widget. By grouping widgets together in a Frame, we simplify the task of programming a GUI by limiting the number of widgets we deal with at any one time. Most often, widgets are grouped by their related functionality to one another. But, in our case, we

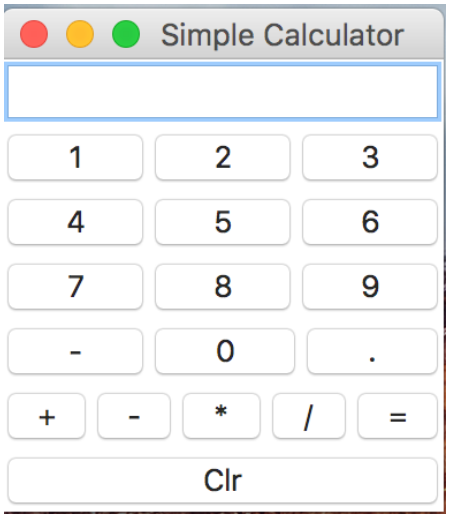
group buttons only for aesthetics (for the calculator, if we don't use Frames all buttons would appear in the window in a single row from left to right). Our **Frame**, called ***frame***, is defined in lines 11-13. Note, ***frame*** is placed inside **self.window**. Next, create two **Button** widgets that are each placed inside ***frame*** in lines 16-18 and in lines 20-23. The first button is labeled "1st" and the other is labeled "2nd". **Button** widgets are used to carry out an action by calling a specific method. In addition, the method ***func1()***, associated with the first button, is called when this **Button** is clicked see lines 28-29. Similarly, the method ***func2()***, associated with the second button, is defined in lines 32-33. When **gui2.py** is executed, the following is displayed



Whenever either button is clicked, a appropriate message is printed in the Debug/IO window as illustrated below:



C. Now modify **simplecalcuator.py** by using what you have learned in part B above. Define six **Frame**s named ***frameA***, ***frameB***, ***frameC***, ***frameD***, ***frameE***, and ***frameF***, each containing the **Button**s shown to their left. For now, assign method **func1()** to each of these buttons. Note: there are 18 buttons in all.



← frameA (3 buttons)

← frameB (3 buttons)

← frameC (3 buttons)

← frameD (3 buttons)

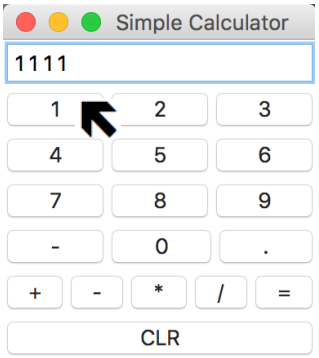
← frameE (5 buttons)

← frameF (1 button)

D. Since there are 18 different buttons, 18 different methods are required to make this GUI function as a working calculator. Remember, ***self.display*** is the variable that holds the contents of the calculator's display. There are two methods that help manipulate ***self.display***. The first is ***get()***, which is used to access the contents of the calculator's display (a string) and the other is ***set()*** which is used to change the calculator's display. Here is the method associated with ***self.button1***. Replace ***func1()*** with this one.

```
def func1(self):
    self.display.set(self.display.get() + '1')
```

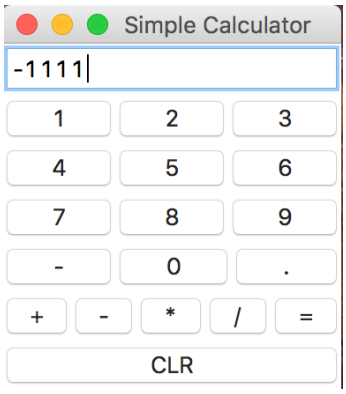
Each time this button is clicked another 1 is entered into ***self.display***.



Fifteen of the remaining methods are programmed in a similar way. Your task is to create these functions and assign them to each of their respective **Button**s. The (=) and negation (-) buttons are different.

E. Suppose the calculator looks like the image above. When the negation (-) button is pressed the calculator now looks like

negation →



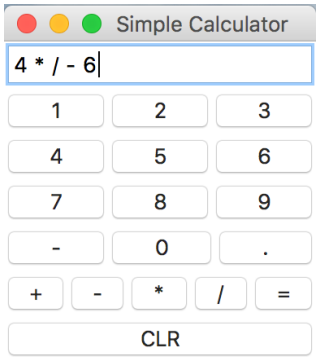
Note: a minus sign is appended in front of the contents of the variable ***self.display***. Now write the method associated with the negation (-) button.

F. The last method to implement is associated with the (=) button. Hint: use the ***eval()*** function. Here are a couple of examples illustrating how it is used. Note: ***eval()*** takes a string representing an arithmetic expression and evaluates its value.

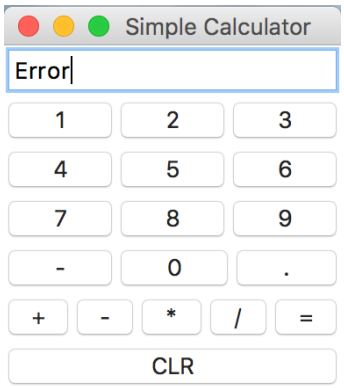
```
>>> eval('56 + 12')
    68
>>> eval('25*36 - 99')
    801
>>>
```

Remember, ***self.display.get()*** returns a string representing the expression appearing in the calculator's display. By applying ***eval()*** to this expression computes the value of this expression. Next, place that value back into ***self.display***.

Finally, use a try-except clause (use **Exception** as the error to catch any that occur) to display an error in ***self.display()*** when an illegal operation is evaluated as illustrated below.



After clicking (=) button we get



Disclaimer:
There are more economical ways to construct a calculator, with far fewer lines of code than was presented here. The beauty of this lab is, a student without much knowledge about **tkinte**r and a few simple examples about **Window**s, **Frame**s, **Button**s an **Entry**s, can get their calculator running close to perfect during the time allotted for this lab.