PROJECT OF GUI CALCULATER USING PYTHON PROGRAMMINGDR.SHAKUNTALA MISRA NATIONAL REHABILITATION UNIVERSITY LUCKNOW

TECHNEX INTENSHIP IIT-BHU

May2021

TECHNEX INTENSHIP IIT-BHU

Alok kumar

PROJECT GUI CALCULATER [python programming]

Introduction

It is always worth having it at the fore-front of your mind that learning a language is not just so you can brag and tell your friends, hey I know so so and so language or to add it to the lists of things you know alongside cooking. Any programming language is just a tool for building stuff. No user ever asked or cared what language was used in the first PC or the iPhone or other valuable software, all they care about is the end result. Whether it works for them or not.

That being said, let us build a calculator *with Python*.

Need to Know

There are things you need to be comfortable with before starting this project. They include

- Python Basics *(for loops, if/else statements, strings etc..)*

- Python Classes
- Basic knowledge of the Tkinter module.

You can find the code for this project on github. But I'd prefer you following along with me from scratch.

Okay so you have read the "Need to Know" reading list or you already know about them?

What should The Calculator do/have?

We are going to list all of the features our calculator should have and use them as a guide while building.The calculator

- should have **A screen, for displaying numbers,**
- **Buttons with numbers on them.**
- Should be able to add, subtract, multiply, divide
- should support decimals.
- should have a backspace button for clearing the screen.

What Python Version Should I use?

Make use of Python 3. On Linux, if you encounter this error,

```
ImportError: No module named '_tkinter', please install the
python3-tk package
```

Then run this command

```
sudo apt-get install python3-tk
```

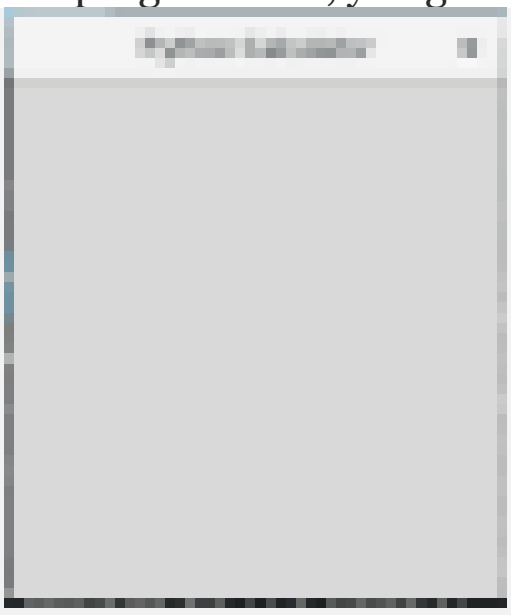
The Code

Stage One

Open an empty file and save it as **calculator.py** and type this in it.

```
from tkinter import *

class Calculator:
    def __init__(self, master):
        self.master = master
        master.title("Python Calculator")



root = Tk()
my_gui = Calculator(root)
root.mainloop()
```

What we've done here is set up our empty board or canvas if you will, where our calculator buttons and screen will live. If you run the program now, you get this



Empty Calculator

Stage Two

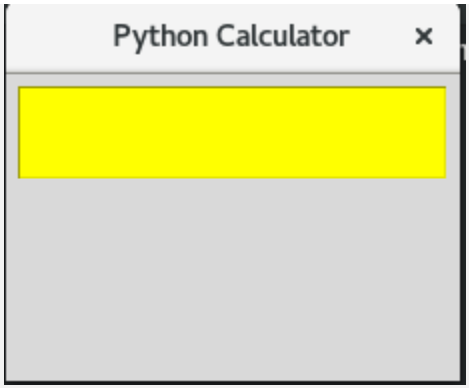
We are going to create our screen in this step.

```
from tkinter import *

class Calculator:
    def __init__(self, master):
        self.master = master
        master.title("Python Calculator")

        # create screen widget
        self.screen = Text(master, state='disabled', width=30,
height=3,background="yellow", foreground="blue")

        # position screen in window

self.screen.grid(row=0,column=0,columnspan=4,padx=5,pady=5)
        self.screen.configure(state='normal')

        # initialize screen value as empty
        self.equation = ''



root = Tk()
my_gui = Calculator(root)
root.mainloop()
```

In the above code, we use the Text class from the Tkinter module as our screen. It takes height and width and background. These properties determine the appearance of our screen. You can experiment with different dimensions and background colors to get familiar with it.

In our code we use the tkinter grid system of placing widgets on a surface. It is pretty straightforward. On line 12, we are simply asking tkinter to place our yellow screen on the zeroth row and zeroth column *(excuse me if that sounds weird, but you should know by now in programming that the first index is usually named zero)*, and that the screen should take the space of 4 columns, hence columnspan=4, also that our yellow screen should have an horizontal and vertical padding of 5 respectively *(padx and pady, just like padding in CSS if you are familiar with it)*.

On line 16, self.equation, is the variable that is going to represent what is typed on the screen. For now it is empty, but don't forget about it because we are coming back to it. At this point, we have this



A Calculator with a Yellow Screen

Stage Three

Our next goal, is to add buttons to the calculator board, so that we can, you know, calculate numbers.

```
from tkinter import *

class Calculator:
    def __init__(self, master):
        self.master = master
        master.title("Python Calculator")

        # create screen widget
        self.screen = Text(master, state='disabled', width=30,
height=3,background="yellow", foreground="blue")

        # position screen in window

self.screen.grid(row=0,column=0,columnspan=4,padx=5,pady=5)
        self.screen.configure(state='normal')

        # initialize screen value as empty
        self.equation = ''

        # create buttons using method createButton
        b1 =  self.createButton(7)
        b2 = self.createButton(8)
        b3 = self.createButton(9)
        b4 = self.createButton(u"\u232B",None)
        b5 = self.createButton(4)
        b6 = self.createButton(5)
        b7 = self.createButton(6)
        b8 = self.createButton(u"\u00F7")
        b9 = self.createButton(1)
        b10 = self.createButton(2)
```

```
        b11 = self.createButton(3)
        b12 = self.createButton('*')
        b13 = self.createButton('.')
        b14 = self.createButton(0)
        b15 = self.createButton('+')
        b16 = self.createButton('-')
        b17 = self.createButton('=',None,34)

        # buttons stored in list
        buttons =
[b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16,b17]

        # intialize counter
        count = 0
        # arrange buttons with grid manager
        for row in range(1,5):
            for column in range(4):
                buttons[count].grid(row=row,column=column)
                count += 1
        # arrange last button '=' at the bottom
        buttons[16].grid(row=5,column=0,columnspan=4)

    def createButton(self,val,write=True,width=7):
        # this function creates a button, and takes one
compulsory argument, the value that should be on the button

        return Button(self.master, text=val,command = lambda:
self.click(val,write), width=width)

root = Tk()
my_gui = Calculator(root)
root.mainloop()
```

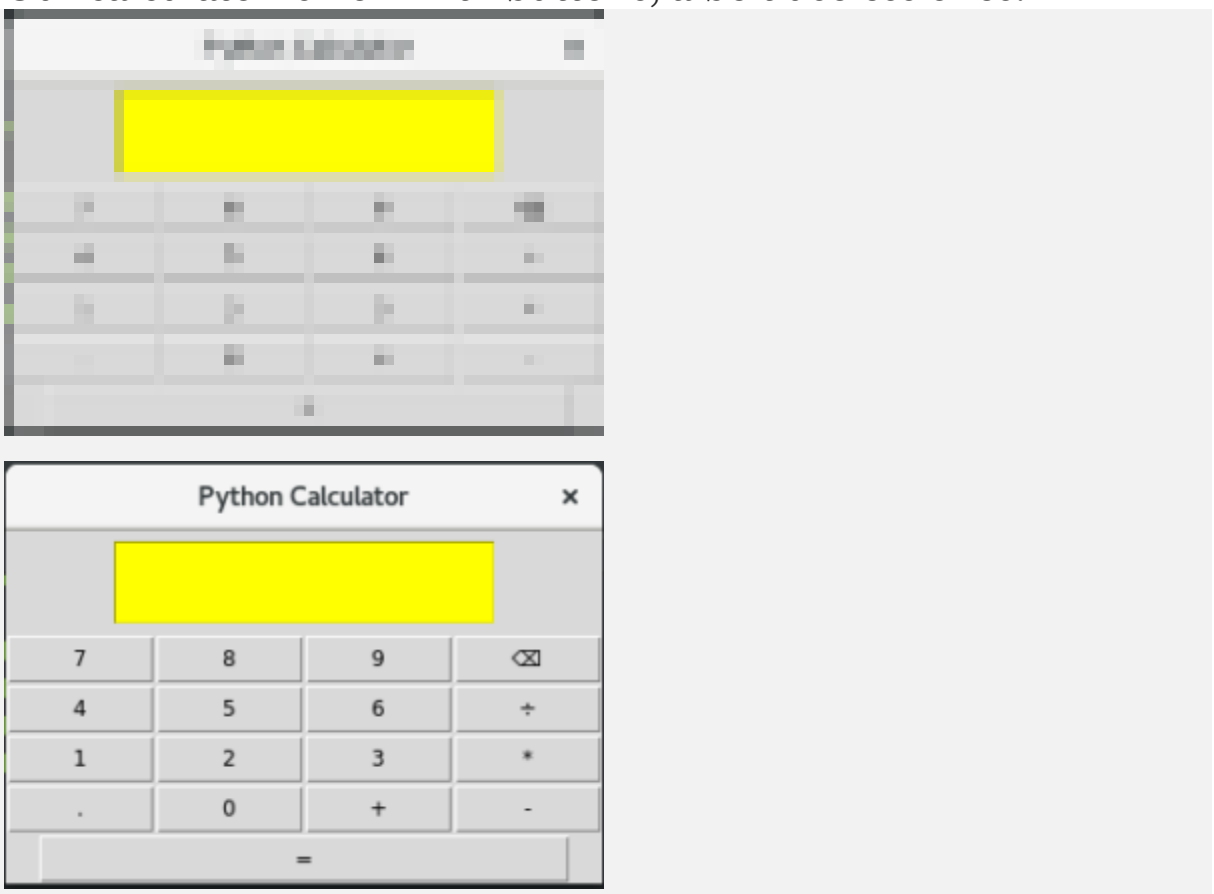
uh-oh, our code just got longer. No worries, just that we have created all our buttons (b1 — b17) using a method **createButton** on line 51. **createButton** makes use of the Tkinter Button widget. The Button widget takes the board in which it will be displayed. In our case, self.master, it takes the text to be displayed on it, and also a command attribute which takes a function. The command attribute is the most important part of the button widget, because without it the button is

useless. We pass a method self.click to it, which we have not defined yet.

In __init__, after creating all our buttons using **createButton**, we store it in a list so we can iterate through it. We now need to place our buttons on our board using the grid system which is all about placing items on available rows and columns on a board.

We iterate through range 1,5 for rows because we are trying to keep only 4 buttons on a row, apart from the = button.

Our calculator is now with buttons, albeit useless ones.



A Calculator with Useless Buttons

Stage Four

We are about to make our buttons less useless. Remember the value self.click that we passed to the command attribute in method **createButton**? Go ahead, look up and search for it. I am waiting... Yup, that one, well let's define it and some other helper methods too.

```
from tkinter import *

class Calculator:
    def __init__(self, master):
        self.master = master
        master.title("Python Calculator")

        # create screen widget
        self.screen = Text(master, state='disabled', width=30,
height=3,background="yellow", foreground="blue")

        # position screen in window

self.screen.grid(row=0,column=0,columnspan=4,padx=5,pady=5)
        self.screen.configure(state='normal')

        # initialize screen value as empty
        self.equation = ''

        # create buttons using method createButton
        b1 =  self.createButton(7)
        b2 = self.createButton(8)
        b3 = self.createButton(9)
        b4 = self.createButton(u"\u232B",None)
        b5 = self.createButton(4)
        b6 = self.createButton(5)
        b7 = self.createButton(6)
        b8 = self.createButton(u"\u00F7")
        b9 = self.createButton(1)
        b10 = self.createButton(2)
        b11 = self.createButton(3)
        b12 = self.createButton('*')
        b13 = self.createButton('.')
        b14 = self.createButton(0)
        b15 = self.createButton('+')
        b16 = self.createButton('-')
        b17 = self.createButton('=',None,34)
```

```
        # buttons stored in list
        buttons =
[b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16,b17]

        # intialize counter
        count = 0
        # arrange buttons with grid manager
        for row in range(1,5):
            for column in range(4):
                buttons[count].grid(row=row,column=column)
                count += 1
        # arrange last button '=' at the bottom
        buttons[16].grid(row=5,column=0,columnspan=4)

    def createButton(self,val,write=True,width=7):
        # this function creates a button, and takes one
compulsory argument, the value that should be on the button

        return Button(self.master, text=val,command = lambda:
self.click(val,write), width=width)

    def click(self,text,write):
        # this function handles what happens when you click a
button
        # 'write' argument if True means the value 'val' should
be written on screen, if None, should not be written on screen
        if write == None:

            #only evaluate code when there is an equation to be
evaluated
            if text == '=' and self.equation:
                # replace the unicode value of division ./.with
python division symbol / using regex
                self.equation= re.sub(u"\u00F7", '/',
self.equation)
                print(self.equation)
                answer = str(eval(self.equation))
                self.clear_screen()
                self.insert_screen(answer,newline=True)
            elif text == u"\u232B":
                self.clear_screen()


        else:
            # add text to screen
            self.insert_screen(text)


    def clear_screen(self):
        #to clear screen
        #set equation to empty before deleting screen
```

```
        self.equation = ''
        self.screen.configure(state='normal')
        self.screen.delete('1.0', END)

    def insert_screen(self, value,newline=False):
        self.screen.configure(state='normal')
        self.screen.insert(END,value)
        # record every value inserted in screen
        self.equation += str(value)
        self.screen.configure(state ='disabled')

root = Tk()
my_gui = Calculator(root)
root.mainloop()
```

The click method defined on line 74, determines what happens when we click our buttons. It takes the value of the button as first argument, while the second argument ***write*** if not defined, means the value of the button should be written to the yellow screen. When ***write*** is set to None, it means no write to screen. This is especially useful when creating the buttons = and backspace. When the = button is clicked the built in python function ***eval*** does all the heavy lifting of evaluating our expression.

The method **insert_screen**, handles writing to the screen anytime the button is clicked, while the **clear_screen** method handles deleting the values on the screen once the backspace button *(the one with the unicode character of u"\u232B")* has been clicked.

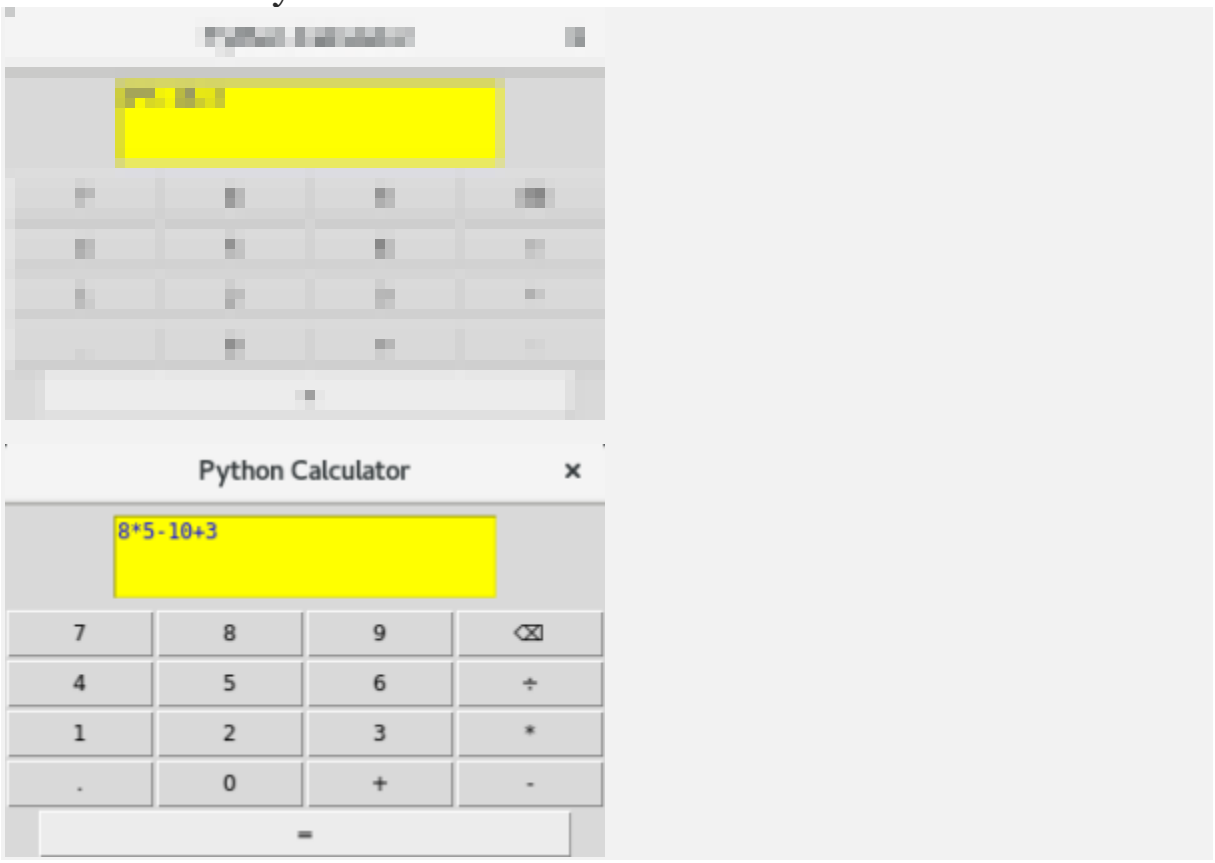
Finally we run

```
python calculator.py
```

or

```
python3 calculator.py
```

as the case maybe and we have a calculator that does stuff.



Functional Calculator

By the way, this calculator is nowhere near perfect. It is what it is, a functional calculator. Feel free to make improvements on your own. For instance, the backspace button currently clears the whole screen. You could improve yours to delete values one after the other. I hope you have learnt something. Let me know what you think in the comments.

Cheers!