Middlesex University Mauritius

# CST 1500 Computer Systems Architecture and Operating Systems

## Coursework 3 - Python Calculator GUI

DIVYA ROSHNI PUGO
(M00796614)

# Table of Contents

Introduction ..... 2
Implementation ..... 3
Improvements..... 7
References ..... 7

# Introduction

The aim of this project is to create a user-friendly calculator which offers both basic as well as scientific functionalities, integrated in a fully functional Graphical User Interface (GUI). The programming language used throughout the program is Python 3.8. and the GUI programming toolkit used is Tkinter.

As part of a basic calculator, this program allows the user to perform the following basic operations:

- Addition
- Subtraction
- Multiplication
- Division
- Equal
- Clear

Each of the mathematical functions operate with full precision without rounding and missing decimals.

In addition to the simple calculations, scientific aspects of a calculator have been implemented. Namely:

- Square Root
- Logarithmic functions
- Trigonometric functions such as:
    - sin(x) ○ cos(x) ○ tan(x)
- Mathematical constants such as:
    - Pi
- Modulus

# Implementation

The math library was imported so that the following mathematical operations can be performed with ease:

- Square Root
- Trigonometric operations
- Modulus
- Logarithmic operations

This was done by coding the following:

```
import math
```

To gain a better understanding and grasp the concept of writing proper functions which carry out the calculations, the material taught during CST 1500 lab was used as well as personal online research was done.

In order to achieve modularity for the application and a degree of code reusing (where necessary) each operation is in its own function. Example:

```
#Declaring two global variables for proper functionality of codes
value = "0"                #Global Variable for display. Setting default display to 0.
operation = ""            #Global Variable for performing calculation

#Function taking input from buttons in calculator
def button_input(symbol, sym_calc):
    global value
    global operation
    if value == "0":            #Ensuring default display is 0
        operation = ""        #Ensuring calculation variable is empty
        value = ""              #Emptying display value variable
    value += symbol               #Storing button name as what will be displayed onto screen
    operation += str(sym_calc)  #Converting what is displayed into proper string function for calculation afterwards
    input_text.set(value)       #Displaying onto screen

#Creating function that will become the command for delete button
def delete():
    global value
    global operation
    value = ""              #Clearing the value variable
    operation = ""        #Clearing the calculation variable
    input_text.set("0")     #Setting the default display back to 0
```

Fig.1

```
#Creating function that will become the command for equal button
def equal():
    global value
    global operation
    try:
        answer = str(eval(operation)) #Evaluating mathematically stored information from calculation variable and storing it as a string
                                                                                                    # in the answer variable
        input_text.set(answer)          #Displaying the answer obtained onto calculator screen
        value = ""                      #Emptying value variable for next operations
        operation = ""                 #Emptying calculation variable for next operations
    except ZeroDivisionError:           #Ensuring that user sees error in case of division by 0
        input_text.set("ERROR. [DEL]:Cancel") #Displaying error and prompting user to press DEL button to go back to calculator


input_text = StringVar()    #Creating input_text variable which will ensure what is displayed onto screen
input_text.set("0")         #Setting default display as 0
```

Fig. 2

For the GUI, Tkinter was used for it is the standard GUI library for Python. When combined with python, Tkinter provides a fast and easy way to create GUI applications. Tkinter is not the only GUI programming toolkit available for Python but is the most commonly used one.

The calculator widget was done by creating a window first and the implementing further features such as frame, display screen and buttons. The coding for this is show by Fig. 3 and Fig. 4 below:

```
#Creating the scientific calculator widget
calculator = Tk()
calculator.title("Scientific Calculator")
calculator.geometry("330x425+500+150")
calculator.resizable(0,0)
```

Fig. 3

```
#Creating a frame for better presentation of screen in GUI
calc = Frame(calculator)
calc.grid()
#Creating the display screen of the calculator and making the numbers to be displayed justified to the right
screen = Entry(calc, font=('Serif', 20), textvariable = input_text, bg="light grey", bd=15, width=19, justify=RIGHT)
screen.grid(row=0, column=0, columnspan=4, padx=7, pady=5)
```

Fig. 4

With the use of a for loop, the buttons for the mathematical operands were created. A *lambda function* was inserted in the command function of the button to ensure that the two lists are mapped against each other accordingly.

```
#Creating a list of the buttons' names and the signs shown on the button
signs = ["√(", "ln(", "log(", "sin(",
         "cos(", "tan(", "π","Abs(",
         "(",")",  ".", "*",
         "/", "+", "-", "^2"]
#Creating a list of the command for the button's name and sign respectively
sign_function = ["math.sqrt(", "math.log(", "math.log10(", "math.sin(",
                 "math.cos(", "math.tan(", "math.pi", "abs(",
                 "(",")",  ".", "*",
                 "/", "+", "-", "**2"]

#Using a for loop to create the button in the row of range 1-5 and column 0-4
#Using the lambda function inside the command function to assign the functionality of the buttons
i = 0
for row in range(1,5):
    for column in range(0,4):
        Button(calc, text=signs[i], width=8, height=2, font=('Serif', 10), bd=4, bg="SkyBlue4", command=lambda name=signs[i], syntax=sign_function[i]: butto
        i += 1
```

For the numbers on the number pad, a variable of type *string* was created before appending each number into a list via a for loop again. Following a similar function as the one written for the mathematical operands, the number buttons became fully functional as well. This is shown by Fig. 5 below:

```
#Creating a string variable for the numbers of the number pad and appending them in a list
#Using a for loop to display the button and a lambda function to assign the command function of each button
number = "123456789"
x = 0
buttons = []
for i in range(5, 8):
    for j in range(3):
        buttons.append(Button(calc, width=8, height=2, font=('Serif', 10), bd=4, bg="LightSteelBlue3", text=number[x], command=lambda symbol=number[x], sym_
        buttons[x].grid(row=i, column=j, pady=1)

        x += 1
#Creating the remaining buttons of the calculator assigning each button to their respective function
zero_button = Button(calc, text='0', width=8, height=5, font=('Serif', 10), bd=4, bg="LightSteelBlue3", command=lambda symbol="0", sym_calc="0": button_inpu
delete_button = Button(calc, text='DEL', width=8, height=2, font=('Serif', 10), bd=4, bg="brown2", command= lambda:delete()).grid(row=4, column=3, padx=1, p
equal_button = Button(calc, text='=', width=8, height=2, font=('Serif', 10), bd=4, bg="SkyBlue4", command= lambda:equal()).grid(row=5, column=3, padx=1, pad

#Calling the tkinter mainloop() function to ensure everything runs smoothly
```

Fig.5

In the end, with the use of the tkinter GUI programming toolkit and other python functions and libraries, the end-product of the calculator is as follows:


Fig. 6

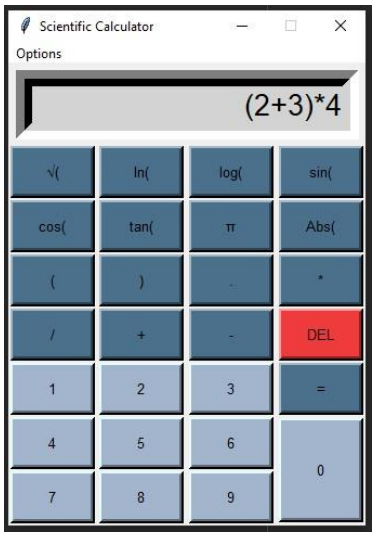


Fig. 7

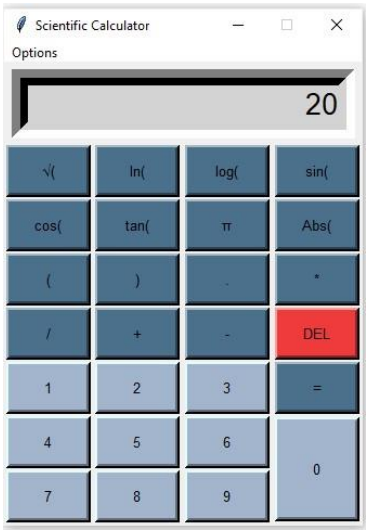


Fig.8

Fig.6 up to Fig. 8 showcase the calculator at use performing simple calculations. Whereas figure Fig. 9 and Fig. 10 showcase the operation of the scientific aspect of the calculator.

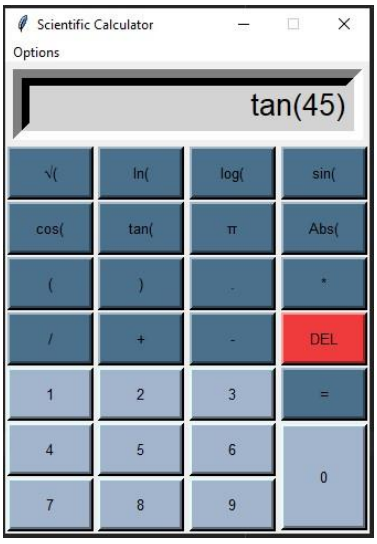


Fig. 9

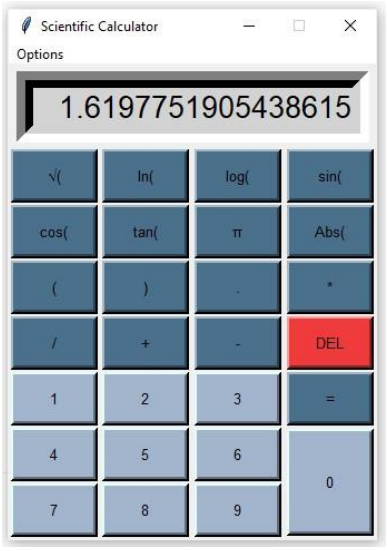


Fig. 10

Fig. 11 and Fig. 12, on the other hand, showcase an error which the calculator displays in case of a ZeroDivisionError and prompts the user on what to do to reset the calculator.

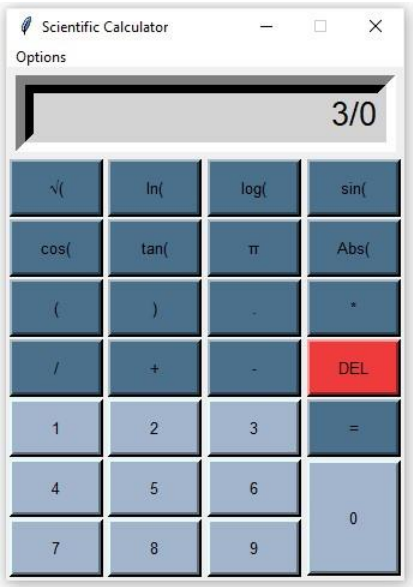


Fig. 11

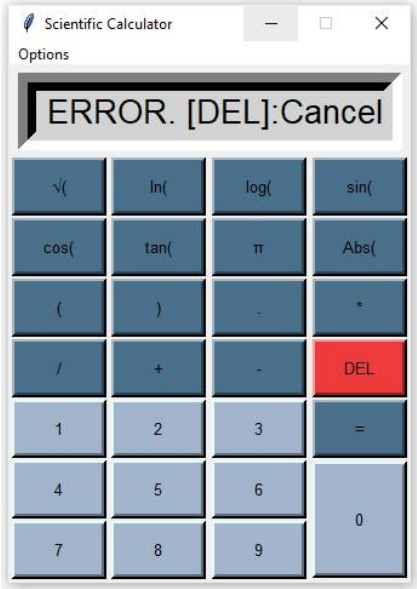


Fig. 12

# Improvements

- Adopting an Object-Oriented approach would make the code more maintainable.
- The display could be a multi-line one.
- More methods to solve equations (simultaneous, quadratic, cubic, quartic…), sketch graphs or solve matrices could be implemented.

# References

1. Python Documentation 3.9.2 (2021) Available at: https://docs.python.org/3/ (Accessed: March 2021).
2. *Python Tkinter Tutorial - GeeksforGeeks* (2020) Available at: https://www.geeksforgeeks.org/python-tkinter-tutorial/ (Accessed: March 2021).
3. Python 3 Tutorial - Tutorialspoint (Not available) Available at: https://www.tutorialspoint.com/python3/index.htm (Accessed: March 2021).