19ECS392
Comprehensive Skill Development V

CODING

PYTHON MINI PROJECT

SCIENTIFIC CALCULATOR USING PYTHON

PROJECT TITLE:

DESIGN A SCIENTIFIC CALCULATOR WITH PROPER GUI USING PYTHON.

PROJECT MEMBERS:

- 122010302036 - B BALAKRISHNA
- 122010302038 - CH SAI GANGA SUDHEER
- 122010302042 - A NAGA VARUN

INTRODUCTION

Introduction of the report gives us a brief description and overview of the project. It gives a scope of every procedure and operations done in the project to make a Scientific Calculator using GUI in Python.

In this project, we have created a Scientific Calculator in Python using Tkinter module that is a fully functional GUI system that contains everything .The Scientific Calculator has the same symbols, operators and numbers as a regular calculator . However, the user cannot enter a input using the keyboard. Mathematical and scientific calculations can be done using Python, a computer language.

By using the Tkinter GUI module in python programming language, we are building a scientific calculator that can be used by the user in a simple way. This calculator like all other have functions like add, subtract, divide, multiply, roots, cos and others. Tkinter creates an interface that can be operated with buttons. Any mathematical problem can be solved by ALU that has all mathematical functions.

Problem Statement

To create a scientific calculator application using Python's Tkinter library. The calculator should have a graphical user interface that supports basic arithmetic operations such as addition, subtraction, multiplication, and division.

In addition to these basic arithmetic operations, the calculator should support advanced mathematical functions such as square root, logarithm, trigonometric functions, and exponential functions. The calculator should also have the ability to convert degrees to radians and vice versa.

The application should have a clear and concise user interface that allows users to enter numeric values and perform calculations with ease. The user should be able to see the current value and result of the calculation displayed on the screen.

The calculator should be able to handle input errors such as division by zero and display appropriate error messages to the user. The application should have a clear button that clears the current value and a reset button that clears the entire calculator history.

ABBREVIATIONS

GUI : Graphical User Interface

ALU : Arithmetic Logic Unit

TKINTER : It is standard GUI library for Python

Design and Methodology

OBJECTIVES

The main objective of the project is to-

• To build Scientific Calculator that can be used to generally solve root, fractional and scientific calculations. When there are several stages to be taken and mathematical functions to be done this kind of calculator is useful.

• This calculator can be used to solve problems that are involving algebraic expressions, polynomial functions and operations with fractions, square roots and other scientific calculations.

• Trigonometric equations, exponential functions and other problems involving logarithms can be solved using scientific calculators.

Features of Scientific Calculator

- All basic operators like **addition, multiplication, subtraction,** and **division**.
- Dedicated buttons for **sin, cos, tan,** and **cot**.
- Functions for **log**
- Functions for **ratio** and **exponents**.
- Other buttons like **Pi, mod,** and **x!**

Algorithm

Here's the step-by-step algorithm:

1. Import the necessary modules: a. tkinter b. math

2. Create a tkinter instance and customize its properties.

3. Create a frame for the calculator and place it in the tkinter instance.

4. Create a class for the calculator.
   a. Initialize the class variables.
   b. Define the methods for performing various operations such as number entry, sum, valid function, operations, clearing entries, and trigonometric functions.

5. Create an object for the calculator class.

6. Create an entry widget to display the result.

7. Create a number pad for the calculator using a for loop.

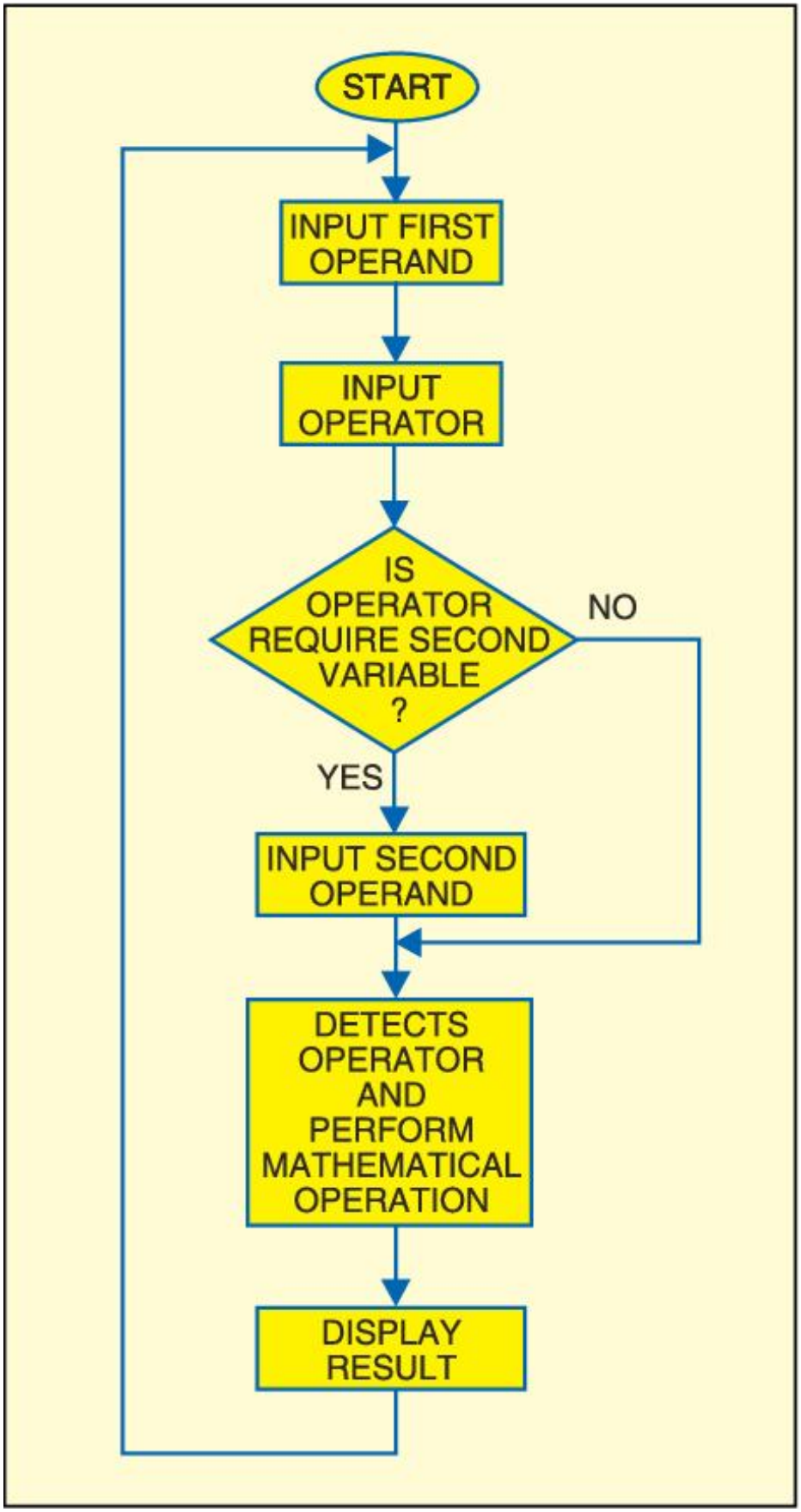
8. Define the functions for each button in the number pad.

9. Create the number pad buttons using a for loop.

10. Place the number pad buttons in the calculator frame.

11. Start the mainloop of the tkinter instance.

FLOW CHART


START
INPUT FIRST OPERAND
INPUT OPERATOR
IS OPERATOR REQUIRE SECOND VARIABLE ?
NO
YES
INPUT SECOND OPERAND
DETECTS OPERATOR AND PERFORM MATHEMATICAL OPERATION
DISPLAY RESULT

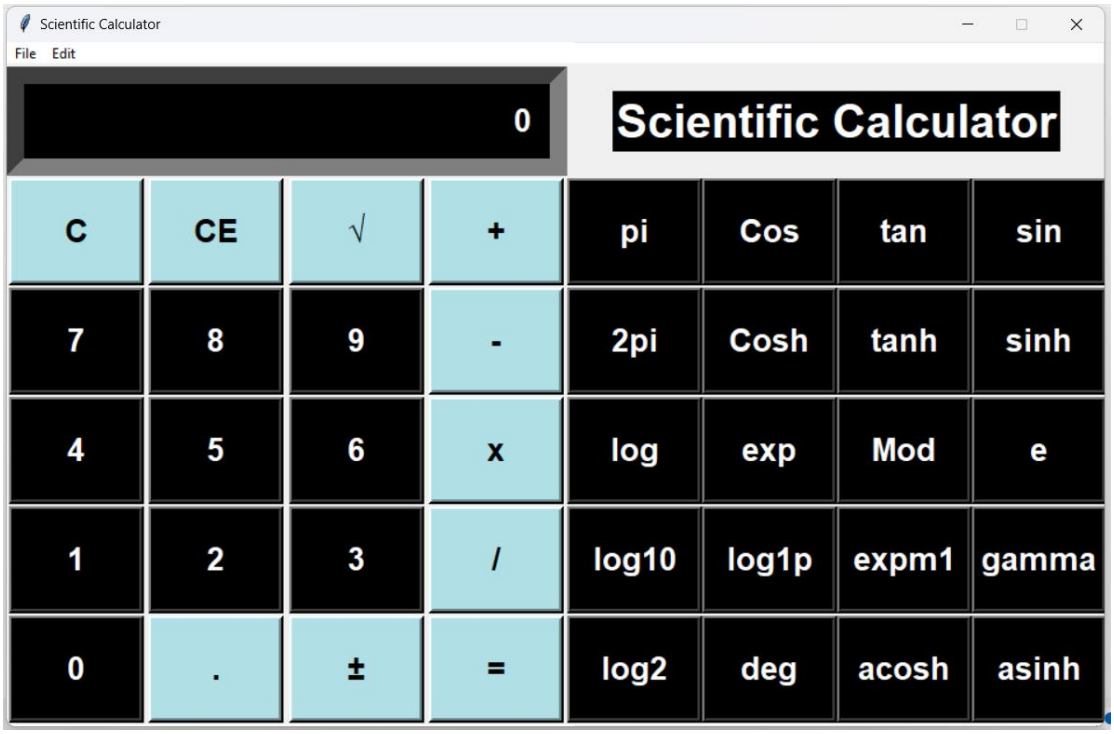
Implementation

The code creates a window and sets the background color to white. The calculator's layout is created using a grid of buttons and an Entry widget that serves as the display. The calculator has basic arithmetic functions like addition, subtraction, multiplication, and division, and also provides other advanced functions like square root, trigonometric functions, logarithms, etc.

The Calc class is used to implement the calculator's functionalities. The class contains methods for handling user input, performing calculations, and displaying the results on the calculator's screen. The numberEnter method is called when a number is pressed, and it appends the digits to the display. The operation method is called when an operator is pressed, and it stores the operator and the current value in class variables. The sum_of_total method is called when the "=" button is pressed, and it performs the appropriate calculation based on the operator and the values stored in class variables. Other methods like Clear_Entry, All_Clear_Entry, pi, cos, sin, log, etc., are used to handle various other functionalities provided by the calculator.

The code creates a GUI window with the calculator's layout, and it uses the bind method to bind the calculator's buttons to their respective functions. The mainloop method is called to run the GUI window and to listen for user inputs.


Scientific Calculator
File Edit
0
C
CE
√
+
7
8
9
-
4
5
6
x
1
2
3
/
0
.
±
=


Scientific Calculator
File Edit
0
Scientific Calculator
C
CE
√
+
pi
Cos
tan
sin
7
8
9
-
2pi
Cosh
tanh
sinh
4
5
6
x
log
exp
Mod
e
1
2
3
/
log10
log1p
expm1
gamma
0
.
±
=
log2
deg
acosh
asinh

CODEING

```
#Step 1: Import module
from tkinter import *
import math
import tkinter.messagebox

#Step 2:  Here we will create geometry or a so-called layout
for the GUI of the calculator by using Tkinter.
root = Tk()
root.title("Scientific Calculator")
root.configure(background = 'white')
root.resizable(width=False, height=False)
root.geometry("480x568+450+90") # sets the geometry
calc = Frame(root)
calc.grid()

#Step 3: Now we will create a class in which we will create
all the functions of the scientific calculator so that they
can be called and perform easily.
class Calc():
    def __init__(self):
        self.total=0
        self.current=''
        self.input_value=True
        self.check_sum=False
        self.op=''
        self.result=False

    def numberEnter(self, num):
        self.result=False
        firstnum=txtDisplay.get()
        secondnum=str(num)
        if self.input_value:
            self.current = secondnum
            self.input_value=False
        else:
            if secondnum == '.':
                if secondnum in firstnum:
                    return
            self.current = firstnum+secondnum
        self.display(self.current)

    def sum_of_total(self):
        self.result=True
        self.current=float(self.current)
        if self.check_sum==True:
            self.valid_function()
```

```
        else:
            self.total=float(txtDisplay.get())

    def display(self, value):
        txtDisplay.delete(0, END)
        txtDisplay.insert(0, value)

    def valid_function(self):
        if self.op == "add":
            self.total += self.current
        if self.op == "sub":
            self.total -= self.current
        if self.op == "multi":
            self.total *= self.current
        if self.op == "divide":
            self.total /= self.current
        if self.op == "mod":
            self.total %= self.current
        self.input_value=True
        self.check_sum=False
        self.display(self.total)

    def operation(self, op):
        self.current = float(self.current)
        if self.check_sum:
            self.valid_function()
        elif not self.result:
            self.total=self.current
            self.input_value=True
        self.check_sum=True
        self.op=op
        self.result=False

    def Clear_Entry(self):
        self.result = False
        self.current = "0"
        self.display(0)
        self.input_value=True

    def All_Clear_Entry(self):
        self.Clear_Entry()
        self.total=0

    def pi(self):
        self.result = False
        self.current = math.pi
        self.display(self.current)

    def tau(self):
```

```
        self.result = False
        self.current = math.tau
        self.display(self.current)

    def e(self):
        self.result = False
        self.current = math.e
        self.display(self.current)

    def mathPM(self):
        self.result = False
        self.current = -(float(txtDisplay.get()))
        self.display(self.current)

    def squared(self):
        self.result = False
        self.current = math.sqrt(float(txtDisplay.get()))
        self.display(self.current)

    def cos(self):
        self.result = False
        self.current =
math.cos(math.radians(float(txtDisplay.get())))
        self.display(self.current)

    def cosh(self):
        self.result = False
        self.current =
math.cosh(math.radians(float(txtDisplay.get())))
        self.display(self.current)

    def tan(self):
        self.result = False
        self.current =
math.tan(math.radians(float(txtDisplay.get())))
        self.display(self.current)

    def tanh(self):
        self.result = False
        self.current =
math.tanh(math.radians(float(txtDisplay.get())))
        self.display(self.current)

    def sin(self):
        self.result = False
        self.current =
math.sin(math.radians(float(txtDisplay.get())))
        self.display(self.current)
```

```
    def sinh(self):
        self.result = False
        self.current =
math.sinh(math.radians(float(txtDisplay.get())))
        self.display(self.current)

    def log(self):
        self.result = False
        self.current = math.log(float(txtDisplay.get()))
        self.display(self.current)

    def exp(self):
        self.result = False
        self.current = math.exp(float(txtDisplay.get()))
        self.display(self.current)

    def acosh(self):
        self.result = False
        self.current = math.acosh(float(txtDisplay.get()))
        self.display(self.current)

    def asinh(self):
        self.result = False
        self.current = math.asinh(float(txtDisplay.get()))
        self.display(self.current)

    def expm1(self):
        self.result = False
        self.current = math.expm1(float(txtDisplay.get()))
        self.display(self.current)

    def lgamma(self):
        self.result = False
        self.current = math.lgamma(float(txtDisplay.get()))
        self.display(self.current)

    def degrees(self):
        self.result = False
        self.current = math.degrees(float(txtDisplay.get()))
        self.display(self.current)

    def log2(self):
        self.result = False
        self.current = math.log2(float(txtDisplay.get()))
        self.display(self.current)

    def log10(self):
        self.result = False
        self.current = math.log10(float(txtDisplay.get()))
```

```
        self.display(self.current)

    def log1p(self):
        self.result = False
        self.current = math.log1p(float(txtDisplay.get()))
        self.display(self.current)

added_value = Calc()

#Step 4: The below code will create a display in the GUI of
the calculator by passing the font style, font size,
background color, foreground color as an argument inside the
entry function.
txtDisplay = Entry(calc, font=('Helvetica',20,'bold'),
                bg='black',fg='white',
                bd=30,width=28,justify=RIGHT)
txtDisplay.grid(row=0,column=0, columnspan=4, pady=1)
txtDisplay.insert(0,"0")

numberpad = "789456123"
i=0
btn = []
for j in range(2,5):
    for k in range(3):
        btn.append(Button(calc, width=6, height=2,
                        bg='black',fg='white',
                        font=('Helvetica',20,'bold'),
                        bd=4,text=numberpad[i]))
        btn[i].grid(row=j, column= k, pady = 1)
        btn[i]["command"]=lambda
x=numberpad[i]:added_value.numberEnter(x)
        i+=1

#Step 6:  Now we will place all the buttons/operators in their
respective position in the grid. This is up to you to set them
as per your choice by changing their row and column value. In
this, each button function is just taking the name of the
operator, width, height, background, foreground, font, and
respective column & row position of the button as an argument.
btnClear = Button(calc, text=chr(67),width=6,
                height=2,bg='powder blue',
                font=('Helvetica',20,'bold')
                ,bd=4, command=added_value.Clear_Entry
                ).grid(row=1, column= 0, pady = 1)

btnAllClear = Button(calc, text=chr(67)+chr(69),
                    width=6, height=2,
                    bg='powder blue',
                    font=('Helvetica',20,'bold'),
```

```
                    bd=4,
                    command=added_value.All_Clear_Entry
                    ).grid(row=1, column= 1, pady = 1)

btnsq = Button(calc, text="\u221A",width=6, height=2,
            bg='powder blue', font=('Helvetica',
                                    20,'bold'),
            bd=4,command=added_value.squared
            ).grid(row=1, column= 2, pady = 1)

btnAdd = Button(calc, text="+",width=6, height=2,
                bg='powder blue',
                font=('Helvetica',20,'bold'),
                bd=4,command=lambda:added_value.operation("add
")
                ).grid(row=1, column= 3, pady = 1)

btnSub = Button(calc, text="-",width=6,
                height=2,bg='powder blue',
                font=('Helvetica',20,'bold'),
                bd=4,command=lambda:added_value.operation("sub
")
                ).grid(row=2, column= 3, pady = 1)

btnMul = Button(calc, text="x",width=6,
                height=2,bg='powder blue',
                font=('Helvetica',20,'bold'),
                bd=4,command=lambda:added_value.operation("mul
ti")
                ).grid(row=3, column= 3, pady = 1)

btnDiv = Button(calc, text="/",width=6,
                height=2,bg='powder blue',
                font=('Helvetica',20,'bold'),
                bd=4,command=lambda:added_value.operation("div
ide")
                ).grid(row=4, column= 3, pady = 1)

btnZero = Button(calc, text="0",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=lambda:added_value.numberEnter(0)
                ).grid(row=5, column= 0, pady = 1)

btnDot = Button(calc, text=".",width=6,
                height=2,bg='powder blue',
                font=('Helvetica',20,'bold'),
                bd=4,command=lambda:added_value.numberEnter(".
")
```

```
                ).grid(row=5, column= 1, pady = 1)
btnPM = Button(calc, text=chr(177),width=6,
            height=2,bg='powder blue',
font=('Helvetica',20,'bold'),
            bd=4,command=added_value.mathPM
            ).grid(row=5, column= 2, pady = 1)

btnEquals = Button(calc, text="=",width=6,
                height=2,bg='powder blue',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.sum_of_total
                ).grid(row=5, column= 3, pady = 1)
# ROW 1 :
btnPi = Button(calc, text="pi",width=6,
            height=2,bg='black',fg='white',
            font=('Helvetica',20,'bold'),
            bd=4,command=added_value.pi
            ).grid(row=1, column= 4, pady = 1)

btnCos = Button(calc, text="Cos",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.cos
            ).grid(row=1, column= 5, pady = 1)

btntan = Button(calc, text="tan",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.tan
            ).grid(row=1, column= 6, pady = 1)

btnsin = Button(calc, text="sin",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.sin
            ).grid(row=1, column= 7, pady = 1)

# ROW 2 :
btn2Pi = Button(calc, text="2pi",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.tau
            ).grid(row=2, column= 4, pady = 1)

btnCosh = Button(calc, text="Cosh",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.cosh
                ).grid(row=2, column= 5, pady = 1)
```

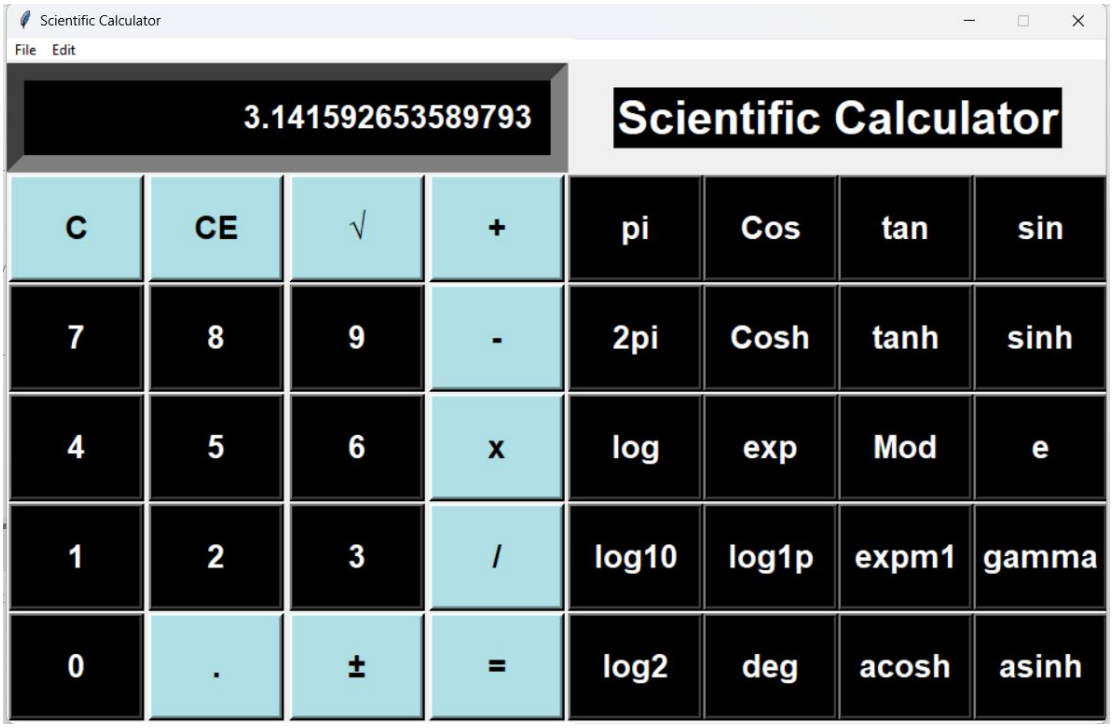
```
btntanh = Button(calc, text="tanh",width=6,
                 height=2,bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
                 bd=4,command=added_value.tanh
                 ).grid(row=2, column= 6, pady = 1)

btnsinh = Button(calc, text="sinh",width=6,
                 height=2,bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
                 bd=4,command=added_value.sinh
                 ).grid(row=2, column= 7, pady = 1)

# ROW 3 :
btnlog = Button(calc, text="log",width=6,
                 height=2,bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
                 bd=4,command=added_value.log
             ).grid(row=3, column= 4, pady = 1)

btnExp = Button(calc, text="exp",width=6, height=2,
                 bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
                 bd=4,command=added_value.exp
             ).grid(row=3, column= 5, pady = 1)

btnMod = Button(calc, text="Mod",width=6,
                 height=2,bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
                 bd=4,command=lambda:added_value.operation("mod
")
                 ).grid(row=3, column= 6, pady = 1)

btnE = Button(calc, text="e",width=6,
                 height=2,bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
                 bd=4,command=added_value.e
             ).grid(row=3, column= 7, pady = 1)

# ROW 4 :
btnlog10 = Button(calc, text="log10",width=6,
                 height=2,bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
                 bd=4,command=added_value.log10
                 ).grid(row=4, column= 4, pady = 1)

btncos = Button(calc, text="log1p",width=6,
                 height=2,bg='black',fg='white',
                 font=('Helvetica',20,'bold'),
```

```
                bd=4,command=added_value.log1p
                ).grid(row=4, column= 5, pady = 1)

btnexpm1 = Button(calc, text="expm1",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd = 4,command=added_value.expm1
                ).grid(row=4, column= 6, pady = 1)

btngamma = Button(calc, text="gamma",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.lgamma
                ).grid(row=4, column= 7, pady = 1)
# ROW 5 :
btnlog2 = Button(calc, text="log2",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.log2
                ).grid(row=5, column= 4, pady = 1)

btndeg = Button(calc, text="deg",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.degrees
            ).grid(row=5, column= 5, pady = 1)

btnacosh = Button(calc, text="acosh",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.acosh
                ).grid(row=5, column= 6, pady = 1)

btnasinh = Button(calc, text="asinh",width=6,
                height=2,bg='black',fg='white',
                font=('Helvetica',20,'bold'),
                bd=4,command=added_value.asinh
                ).grid(row=5, column= 7, pady = 1)

lblDisplay = Label(calc, text = "Scientific Calculator",
                font=('Helvetica',30,'bold'),
                bg='black',fg='white',justify=CENTER)

lblDisplay.grid(row=0, column= 4,columnspan=4)

#Step 7: Now at last we will create a menubar of the
calculator GUI.
def iExit():
```

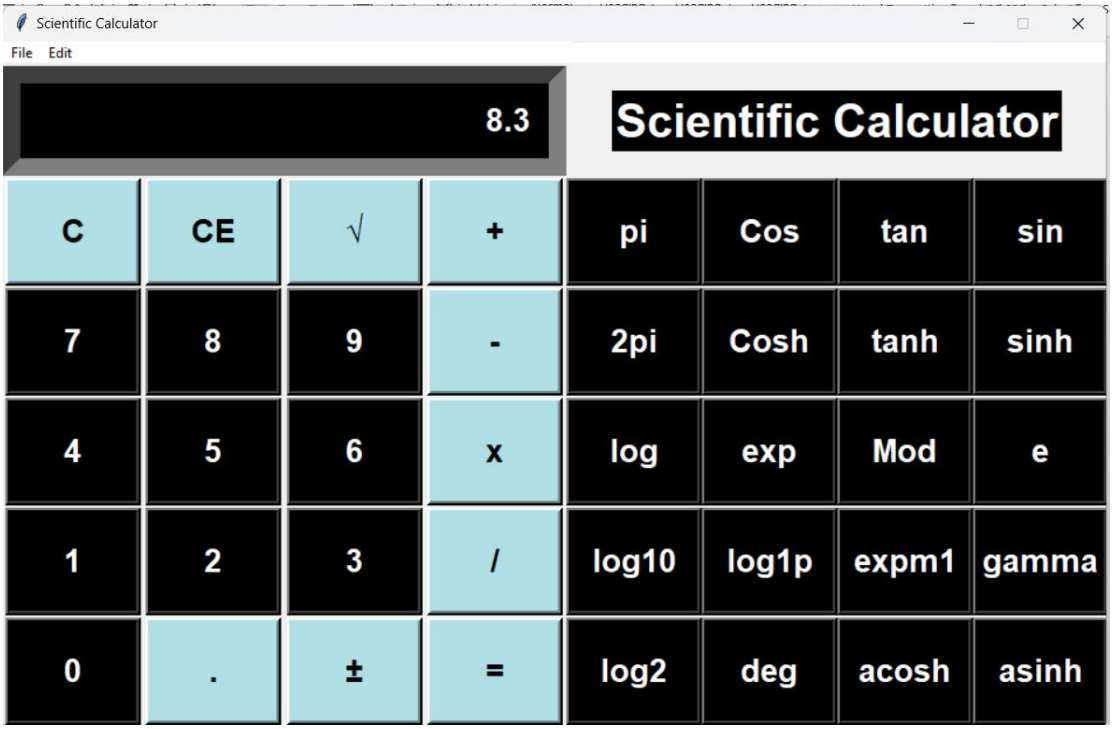
```
    iExit = tkinter.messagebox.askyesno("Scientific
Calculator",
                                        "Do you want to
exit ?")
    if iExit>0:
        root.destroy()
        return

def Scientific():
    root.resizable(width=False, height=False)
    root.geometry("944x568+0+0")


def Standard():
    root.resizable(width=False, height=False)
    root.geometry("480x568+0+0")

menubar = Menu(calc)

# ManuBar 1 :
filemenu = Menu(menubar, tearoff = 0)
menubar.add_cascade(label = 'File', menu = filemenu)
filemenu.add_command(label = "Standard", command = Standard)
filemenu.add_command(label = "Scientific", command =
Scientific)
filemenu.add_separator()
filemenu.add_command(label = "Exit", command = iExit)

# ManuBar 2 :
editmenu = Menu(menubar, tearoff = 0)
menubar.add_cascade(label = 'Edit', menu = editmenu)
editmenu.add_command(label = "Cut")
editmenu.add_command(label = "Copy")
editmenu.add_separator()
editmenu.add_command(label = "Paste")

root.config(menu=menubar)

root.mainloop()
```

OUTPUTS

Pi =



2.3 + 6 =



CONCLUSION

The calculator is one application that we use in our day to day lives. Nowadays calculating large numbers with **complex mathematical functions is difficult and time consuming**. So, we have built a simple scientific calculator using GUI in python which allows us to perform both simple and complex calculations. We used Tkinter module of python to implement GUI.

We have used T**kinter module, one of the fastest and easiest way to build GUI application** which is most commonly used. Python offers many various utilities to design the graphical user interface. In this we have developed a scientific calculator using Tkinter and defined what happens when user interacts with user interface.

In order to start building the GUI calculator, it becomes necessary to understand the structure. There are some pre-requisites for creating a calculator program in python that are **Functions in Python, Function Arguments, User-defined Functions in Python** Thus using Tkinter frames, labels and functionalities, Tkinter windows and widgets we can successfully build up a working scientific calculator We conclude our project in we learned mathematical basic functions ALU and Tkinter library in GUI.