Annexure-I

INT213

Scientific Calculator Using Python Programming

A Project report

Submitted in partial fulfillment of the requirements for the award of degree of

B.Tech – CSE (Hons.)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

Submitted By

Name of student: N Krishnachaithanya

Registration Number: 11916375

Signature of the student

ACKNOWLEDGMENT

First and the foremost I would like to thank to my almighty for giving me courage to a word of thanks to my teacher, friends and other sources that gave an unending support and helped me in numerous ways from the first stage of my term assignment conceived. I would also like to thank my family members for their whole hearted support and co-operation. I duly acknowledge the contribution Dr.Rahul for invaluable help. Coding scientific calculator is an uphill task and would have not been possible without proper and timely assistance of Dr.Rahul. I would also thanks to all my friends for forwarding their suggestions to make necessary modifications. Special thanks to Dr.Rahul for his able guidance and motivation where he brings lots of interests over the subject.

ABSTRACT

It is a Scientific Calculator application. It is used to calculate the math fucntions easily.

In this application two types of calculator are there

1.Standard Calculator

2.Scientific Calculator

first one is very simple to solve arithmetic operations. And also convert the result into either integer or float pointing number.

And then second one is scientific notation type math functions are there like sin.cos,tan,log etc.

it is very useful to solve the odd math calculations in less time and in simple manner and also easily to use.

Especially I used menu bar with two items one is standard and second one is scientific

after clicking the standard item it will shows the Standard Calculator

after clicking the scientific item it will shows the Scientific Calculator with standard also.

By using Tkinter in python I developed this application it is also converted into .exe file by using pyinstaller then it is now a desktop application.

Lastly it shows desktop icon in out system if u install it otherwise not show in your desktop.

INDEX

CHAPTER 1-INTRODUCTION

PYTHON

Python Language Introduction

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.

- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python graphical user interfaces (GUIs)

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython** − This is an open-source Python interface for wxWindows http://wxpython.org.
- **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

There are many other interfaces available, which you can find them on the net.

PYTHON TKINTER GUI

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

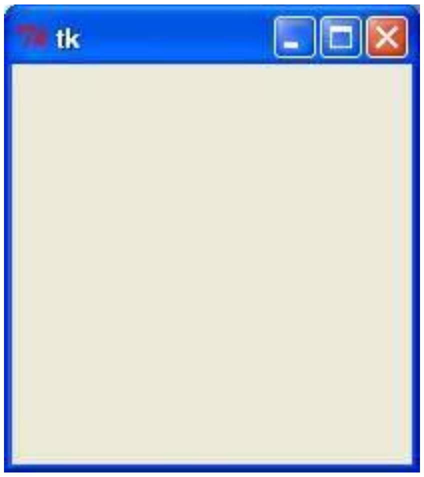
Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python


import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

This would create a following window −



Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

Sr.No.	Operator & Description
1	**Button** The Button widget is used to display buttons in your application.
2	**Canvas** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	**Checkbutton** The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	**Entry** The Entry widget is used to display a single-line text field for accepting values from a user.
5	**Frame** The Frame widget is used as a container widget to organize other widgets.
6	**Label** The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	**Listbox** The Listbox widget is used to provide a list of options to a user.
8	**Menubutton** The Menubutton widget is used to display menus in your application.

9	**Menu** The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	**Message** The Message widget is used to display multiline text fields for accepting values from a user.
11	**Radiobutton** The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	**Scale** The Scale widget is used to provide a slider widget.
13	**Scrollbar** The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	**Text** The Text widget is used to display text in multiple lines.
15	**Toplevel** The Toplevel widget is used to provide a separate window container.
16	**Spinbox** The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	**PanedWindow** A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

18	**LabelFrame** A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	**tkMessageBox** This module is used to display message boxes in your applications.

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *pack()* Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method − This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

CHAPTER-2 IMPLEMENTATION

Technologies used - Python 3.9.0

Python Tkinter GUI

Language used - Python

CODE OF PROJECT

```
from tkinter import *
import math
import tkinter.messagebox

root = Tk()
root.title("Scientific Calculator")
root.configure(background="light green")
root.resizable(width=False, height=False)
root.geometry("480x624+20+20")
calc = Frame(root)
calc.grid()


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 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
        if self.op == "inv":
            self.total = 1 /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 tanh(self):
        self.reult = False
        self.current = math.tanh(math.radians(float(txtDisplay.get())))
        self.display(self.current)

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

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

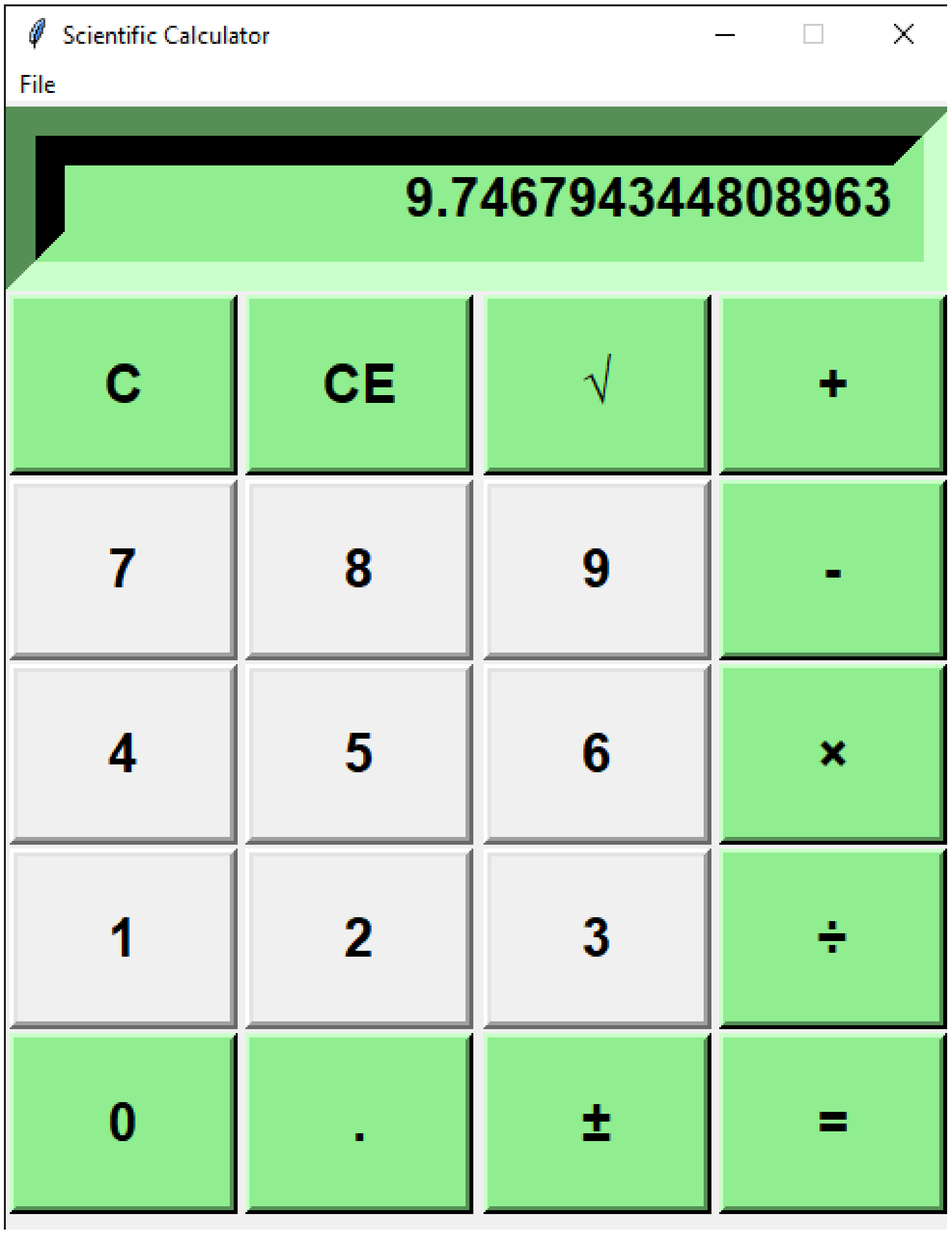
```
def sin(self):
    self.reult = False
    self.current = math.sin(math.radians(float(txtDisplay.get())))
    self.display(self.current)

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

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

def mathsPM(self):
    self.reult = False
    self.current = -(float(txtDisplay.get()))
    self.display(self.current)

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

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

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

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

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

def tau(self):
    self.reult = False
    self.current = math.tau
    self.display(self.current)

def e(self):
    self.reult = False
    self.current = math.e
    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()
txtDisplay = Entry(calc, relief=SUNKEN, font=('grotesque', 20, 'bold'), bg="light green", 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, font=('grotesque', 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
btnClear = Button(calc, text=chr(67), width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
                  command=added_value.Clear_Entry).grid(row=1, column=0, pady=1)
btnAllClear = Button(calc, text=chr(67) + chr(69), width=6, height=2, font=('grotesque', 20, 'bold'), bd=4,
                    bg="light green", command=added_value.all_Clear_Entry).grid(row=1, column=1, pady=1)

btnSq = Button(calc, text="√", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
              command=added_value.squared).grid(row=1, column=2, pady=1)
btnAdd = Button(calc, text="+", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
               command=lambda: added_value.operation("add")).grid(row=1, column=3, pady=1)

btnSub = Button(calc, text="-", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
               command=lambda: added_value.operation("sub")).grid(row=2, column=3, pady=1)
btnMult = Button(calc, text="×", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
                command=lambda: added_value.operation("multi")).grid(row=3, column=3, pady=1)

btnDiv = Button(calc, text=chr(247), width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
               command=lambda: added_value.operation("divide")).grid(row=4, column=3, pady=1)
btnZero = Button(calc, text="0", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
                command=lambda: added_value.numberEnter(0)).grid(row=5, column=0, pady=1)

btnDot = Button(calc, text=".", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
               command=lambda: added_value.numberEnter(".")).grid(row=5, column=1, pady=1)
btnPM = Button(calc, text=chr(177), width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
              command=added_value.mathsPM).grid(row=5, column=2, pady=1)

btnEquals = Button(calc, text="=", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="light green",
                  command=added_value.sum_of_total).grid(row=5, column=3, pady=1)

btnPi = Button(calc, text='π', width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
              command=added_value.pi).grid(row=1, column=4, pady=1)
btnCos = Button(calc, text="cos", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
               command=added_value.cos).grid(row=1, column=5, pady=1)

btnTan = Button(calc, text="tan", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
               command=added_value.tan).grid(row=1, column=6, pady=1)
btnSin = Button(calc, text="sin", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
               command=added_value.sin).grid(row=1, column=7, pady=1)

btn2Pi = Button(calc, text='2π', width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
               command=added_value.tau).grid(row=2, column=4, pady=1)
btnCosh = Button(calc, text="cosh", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
                command=added_value.cosh).grid(row=2, column=5, pady=1)

btnTanh = Button(calc, text="tanh", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
                command=added_value.tanh).grid(row=2, column=6, pady=1)
btnSinh = Button(calc, text="sinh", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
                command=added_value.sinh).grid(row=2, column=7, pady=1)

btnLog = Button(calc, text='log', width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
               command=added_value.log).grid(row=3, column=4, pady=1)
btninv = Button(calc, text="Inv", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
               command=lambda: added_value.operation("inv")).grid(row=3, column=5, pady=1)

btnMod = Button(calc, text="Mod", width=6, height=2, font=('grotesque', 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, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
             command=added_value.e).grid(row=3, column=7, pady=1)

btnLog2 = Button(calc, text='log2', width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
               command=added_value.log2).grid(row=4, column=4, pady=1)
btnDeg = Button(calc, text="deg", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
              command=added_value.degrees).grid(row=4, column=5, pady=1)

btnAcosh = Button(calc, text="acosh", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
                command=added_value.acosh).grid(row=4, column=6, pady=1)
btnAsinh = Button(calc, text="asinh", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="gray",
                command=added_value.asinh).grid(row=4, column=7, pady=1)

btnLog10 = Button(calc, text='log10', width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
                command=added_value.log10).grid(row=5, column=4, pady=1)
btnLog1p = Button(calc, text="log1p", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
                command=added_value.log1p).grid(row=5, column=5, pady=1)

btnExpm1 = Button(calc, text="expm1", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
                command=added_value.expm1).grid(row=5, column=6, pady=1)
btnLgamma = Button(calc, text="lgamma", width=6, height=2, font=('grotesque', 20, 'bold'), bd=4, bg="Green",
                 command=added_value.lgamma).grid(row=5, column=7, pady=1)

lblDisplay = Label(calc, text="Scientific Calculator", font=('grotesque', 30, 'bold'), justify=CENTER)
lblDisplay.grid(row=0, column=4, columnspan=4)

lblDisplay = Label(calc, text="NkC Calculator", font=('grotesque', 30, 'bold'), justify=CENTER)
lblDisplay.grid(row=6, column=0, columnspan=4)


def iExit():
    iExit = tkinter.messagebox.askyesno("Scientific Calculator - NkC Configured", "Confirm if you want to exit")
    if iExit > 0:
        root.destroy()
        return


def Scientific():
    root.resizable(width=False, height=False)
    root.geometry("944x624+20+20")


def Standard():
    root.resizable(width=False, height=False)
    root.geometry("480x624+20+20")


menubar = Menu(calc)

filemenu = Menu(menubar, tearoff=0)
menubar.add_cascade(label="File", menu=filemenu)
filemenu.add_command(label="Standadrd", command=Standard)
filemenu.add_command(label="Scientific", command=Scientific)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=iExit)
root.config(menu=menubar)
root.mainloop()
```

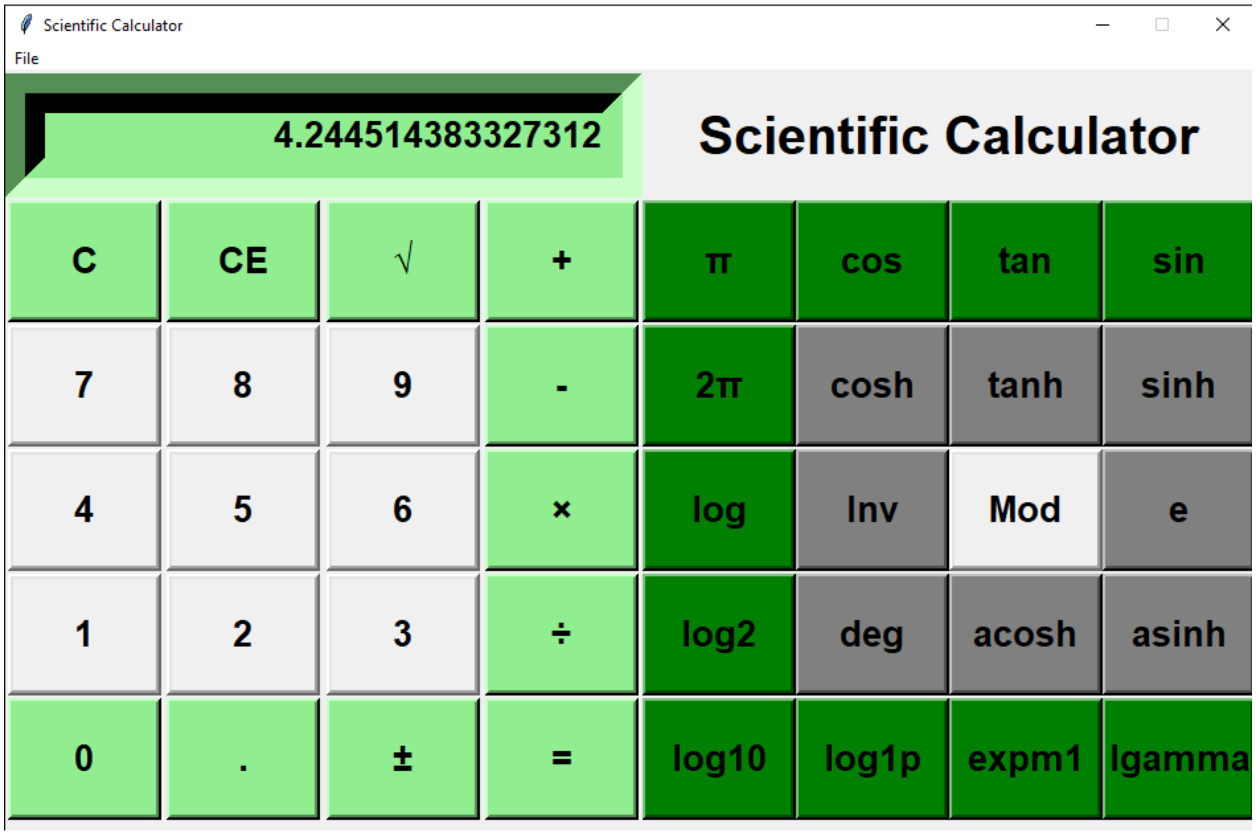
CHAPTER-3 SCREENSHOTS

GUI – Main display window with name of the Scientific Calculator

1.Standard Calculator



2.Scientific Calculator



CHAPTER-4 CONCLUSION

This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language. As we have used python Tkinter as a GUI it provides various controls, such as buttons, labels and text boxes to build a user friendly application.

The fast growing use of internet confirms the good future and scope of the proposed project.

Finally it has taught us a valuable lifelong lesson about the improvements and work Individually

THE END