PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100 Feet Ring Road, BSK III Stage, Bengaluru-560 085

Department of Computer Science & Engineering

Title: Virtual Scientific Calculator

Team Member details:

PES2UG21CS486 – Satyam Kumar

PES2UG21CS548 – Sujal S

PES2UG21CS577 – Thejas P Rao

ABSTRACT

1. **Our project is mainly focused in providing the user a mathematical program which will help him to solve various mathematical operations required in his day to day life.**

2. **Our project along with scientific calculator consists of root finder, simultaneous equation solver, Complex calculator, Volume finder, Area finder, Derivative and integral calculator etc. This allows the student to solve various study related problems in an easier way.**

3. **In a world where, people are moving towards virtual reality, our virtual calculator program allows the user to calculate using hand gestures.**

4. **This project gave us an opportunity to research more on GUI applications and Open-CV module which enabled us to use the camera module and hand-detection module.**

5. **Finally, a scientific calculator is a necessity now for each and every student and hence we think this project will attract more number of students to use it.**

Table of Contents:

INTRODUCTION

A Scientific calculator is a type of electronic calculator, usually but not always handheld, designed to calculate problems in science, engineering, and mathematics.

In this project of ours, we have tried to replicate the functions performed by a scientific calculator with little additional functionality such as Derivative and integral calculator, Volume finder, Area finder, Complex Calculator and Virtual Calculator etc.

This Project helps the user to perform various mathematical functions through just one program.

CODING:

1.MODULES IMPORTED:

```
from tkinter import *

import math

import cv2

from cvzone.HandTrackingModule import HandDetector

import numpy

from Simultaneous import simultaneous_equation_solver

from Rootfinder import root_finder

from Virtual_calculator import virtual_calculator

from Derivative import calculus

from complex_calci import complex_calculator

from Volume import volume_finder

from Area import Area_finder

import cmath

import sympy
```

2.Scientific Calculator

```
def click(value):

    ex = entryField.get()

    answer = ''



    try:
```

```
if value == 'C':

    ex = ex[0:len(ex) - 1]

    entryField.delete(0, END)

    entryField.insert(0, ex)

    return


elif value == 'CE':

    entryField.delete(0, END)


elif value == '√':

    answer = round(math.sqrt(eval(ex)), 4)


elif value == 'π':

    answer = str(ex) + str(round(math.pi, 4))


elif value == 'cosθ':

    answer = round(math.cos(math.radians(eval(ex))), 4)


elif value == 'tanθ':

    answer = round(math.tan(math.radians(eval(ex))), 4)


elif value == 'sinθ':

    answer = round(math.sin(math.radians(eval(ex))), 4)
```

```
elif value == '2π':
    answer = str(ex) + str(round(2 * math.pi, 4))


elif value == 'cosh':
    answer = round(math.cosh(eval(ex)), 4)


elif value == 'tanh':
    answer = round(math.tanh(eval(ex)), 4)


elif value == 'sinh':
    answer = round(math.sinh(eval(ex)), 4)


elif value == chr(8731):
    answer = round(eval(ex) ** (1 / 3), 4)


elif value == 'x\u02b8':  # 7**2
    entryField.insert(END, '**')
    return


elif value == 'x\u00B3':
    answer = round(eval(ex) ** 3, 4)


elif value == 'x\u00B2':
    answer = round(eval(ex) ** 2, 4)
```

```
elif value == 'ln':
    answer = round(math.log(eval(ex)), 4)

elif value == 'deg':
    answer = round(math.degrees(eval(ex)), 4)

elif value == "rad":
    answer = round(math.radians(eval(ex)), 4)

elif value == 'e':
    answer = str(ex) + str(round(math.e, 4))

elif value == 'log₁₀':
    answer = round(math.log10(eval(ex)), 4)

elif value == 'x!':
    answer = math.factorial(int(ex))

elif value == chr(247):  # 7/2=3.5
    entryField.insert(END, "/")
    return

elif value == '=':
```

```
            answer = round(eval(str(ex)), 4)


        else:

            entryField.insert(END, str(value))

            return


        entryField.delete(0, END)

        entryField.insert(0, str(answer))


    except SyntaxError:

        entryField.delete(0, END)

        entryField.insert(0, "Syntax Error")

    except ZeroDivisionError:

        entryField.delete(0, END)

        entryField.insert(0, "Zero division Error")

    except ValueError:

        entryField.delete(0, END)

        entryField.insert(0, "Value Error")




root = Tk()

root.title('Scientific Calculator')

root.config(bg='SpringGreen2')

root.geometry('680x486+100+100')
```

```
entryField = Entry(root, font=('arial', 20, 'bold'), bg="SpringGreen3", fg='black', bd=10, relief=SUNKEN, width=30)

entryField.grid(row=0, column=0, columnspan=8)



button_text_list = ["C", "CE", "√", "+", "π", "cosθ", "tanθ", "sinθ",

                    "1", "2", "3", "-", "2π", "cosh", "tanh", "sinh",

                    "4", "5", "6", "*", chr(8731), "x\u02b8", "x\u00B3", "x\u00B2",

                    "7", "8", "9", chr(247), "ln", "deg", "rad", "e",

                    "0", ".", "%", "=", "log₁₀", "(", ")", "x!"]

rowvalue = 1

columnvalue = 0

for i in button_text_list:


    button = Button(root, width=5, height=2, bd=2, relief=SUNKEN, text=i, bg="SpringGreen3", fg='black',

                    font=('arial', 18, 'bold'), activebackground='dodgerblue', command=lambda button=i: click(button))

    button.grid(row=rowvalue, column=columnvalue, pady=1)

    columnvalue += 1

    if columnvalue > 7:

        rowvalue += 1

        columnvalue = 0

menubar = Menu(root)
```

```
filemenu = Menu(menubar, tearoff=0)

filemenu.add_command(label="Simultaneous equation solver", command=simultaneous_equation_solver)

filemenu.add_command(label="Rootfinder", command=root_finder)

filemenu.add_command(label="Virtual Calculator", command=virtual_calculator)

filemenu.add_command(label="Derivative", command=calculus)

filemenu.add_command(label="Complex Calculator", command=complex_calculator)

filemenu.add_command(label="Volume finder", command=volume_finder)

filemenu.add_command(label="Area finder", command=Area_finder)

filemenu.add_separator()

filemenu.add_command(label="Exit", command=root.quit)

menubar.add_cascade(label="Functions", menu=filemenu)

root.config(menu=menubar)

root.mainloop()
```

3.SIMULTANEOUS EQUATION SOLVER

```
def simultaneous_equation_solver():

    sim = Tk()

    sim.title("SIMULTANEOUS EQUATION SOLVER")

    sim.config(bg="yellow")

    sim.geometry("650x525")

    L1 = Label(sim, text="Simultaneous equation solver", font=("Arial", 25), bg="yellow")

    L1.pack()


    def two_variable():
```

```
def clear1():
    E1.delete(0, END)
    E2.delete(0, END)
    E3.delete(0, END)
    E4.delete(0, END)
    E5.delete(0, END)
    E6.delete(0, END)
L2 = Label(sim, text="2 Variables: ", font=("Arial",25), bg="yellow")
L2.place(x=10,y=40)
L3 = Label(sim, text="First equation:              x +        y = ", font=("Arial",20), bg="yellow")
L3.place(x=10,y=90)
E1 = Entry(sim,width=3, font=("Arial",20))
E1.place(x=230,y=90)
E2 = Entry(sim,width=3, font=("Arial",20))
E2.place(x=330,y=90)
E3 = Entry(sim,width=3, font=("Arial",20))
E3.place(x=445,y=90)
L3 = Label(sim, text="Second equation:        x +        y = ", font=("Arial",20), bg="yellow")
L3.place(x=10,y=130)
E4 = Entry(sim,width=3, font=("Arial",20))
E4.place(x=230,y=130)
E5 = Entry(sim,width=3, font=("Arial",20))
E5.place(x=330,y=130)
E6 = Entry(sim,width=3, font=("Arial",20))
```

```
    E6.place(x=445,y=130)
    def simultaneous_eqsolver():
        a1 = int(E1.get())
        a2 = int(E4.get())
        b1 = int(E2.get())
        b2 = int(E5.get())
        c1 = int(E3.get())
        c2 = int(E6.get())
        D = (a1 * b2) - (a2 * b1)

        try:
            x = ((c1 * b2) - (c2 * b1)) / D
            y = ((a1 * c2) - (a2 * c1)) / D
            Lab = Label(sim, text=f"x = {x}, y = {y} ", font=("Arial",20), bg="yellow")
            Lab.place(x=10,y=220)
        except ZeroDivisionError:
            L3 = Label(sim, text="Error in the equation", font=("Arial",20), bg="yellow")
            L3.place(x=10,y=220)
    button1 = Button(sim, text="Submit", font=("Arial",10), padx=10, pady= 10, command=simultaneous_eqsolver)
    button1.place(x=50, y=170)
    button2 = Button(sim, text="clear", font=("Arial",10), padx=10, pady= 10, command=clear1)
    button2.place(x=150, y=170)
two_variable()
```

```
    def three_variable():
        def clear1():
            E1.delete(0, END)
            E2.delete(0, END)
            E3.delete(0, END)
            E4.delete(0, END)
            E5.delete(0, END)
            E6.delete(0, END)
            E7.delete(0, END)
            E8.delete(0, END)
            E9.delete(0, END)
            E10.delete(0, END)
            E11.delete(0, END)
            E12.delete(0, END)

        L2 = Label(sim, text="3 Variables: ", font=("Arial", 25), bg="yellow")
        L2.place(x=10, y=250)
        L3 = Label(sim, text="First equation:            x +        y +        z = ", font=("Arial",20), bg="yellow")
        L3.place(x=10, y=300)
        E1 = Entry(sim, width=3, font=("Arial",20))
        E1.place(x=230, y=300)
        E2 = Entry(sim, width=3, font=("Arial",20))
        E2.place(x=330, y=300)
```

```
E3 = Entry(sim, width=3, font=("Arial",20))
E3.place(x=430, y=300)
E4 = Entry(sim, width=3, font=("Arial",20))
E4.place(x=545, y=300)
L3 = Label(sim, text="Second equation:          x +          y +          z = ", font=("Arial",20), bg="yellow")
L3.place(x=10, y=340)
E5 = Entry(sim, width=3, font=("Arial",20))
E5.place(x=230, y=340)
E6 = Entry(sim, width=3, font=("Arial",20))
E6.place(x=330, y=340)
E7 = Entry(sim, width=3, font=("Arial",20))
E7.place(x=430, y=340)
E8 = Entry(sim, width=3, font=("Arial", 20))
E8.place(x=545, y=340)
L4 = Label(sim, text="Third equation:             x +          y +          z = ", font=("Arial", 20), bg="yellow")
L4.place(x=10, y=380)
E9 = Entry(sim, width=3, font=("Arial", 20))
E9.place(x=230, y=380)
E10 = Entry(sim, width=3, font=("Arial", 20))
E10.place(x=330, y=380)
E11= Entry(sim, width=3, font=("Arial", 20))
E11.place(x=430, y=380)
```

```
E12 = Entry(sim, width=3, font=("Arial", 20))
E12.place(x=545, y=380)
def simultaneous_eqsolver():
    a1 = int(E1.get())
    a2 = int(E5.get())
    a3 = int(E9.get())
    b1 = int(E2.get())
    b2 = int(E6.get())
    b3 = int(E10.get())
    c1 = int(E3.get())
    c2 = int(E7.get())
    c3 = int(E11.get())
    d1 = int(E4.get())
    d2 = int(E8.get())
    d3 = int(E12.get())

    D = a1 * (b2 * c3 - c2 * b3) - b1 * (a2 * c3 - c2 * a3) + c1 * (a2 * b3 - a3 * b2)


    try:
        x = round((d1 * (b2 * c3 - c2 * b3) - b1 * (d2 * c3 - c2 * d3) + c1 * (d2 * b3 - d3 * b2)) / D,3)
        y = round((a1 * (d2 * c3 - c2 * d3) - d1 * (a2 * c3 - c2 * a3) + c1 * (a2 * d3 - a3 * d2)) / D, 3)
        z = round((a1 * (b2 * d3 - d2 * b3) - b1 * (a2 * d3 - d2 * a3) + d1 * (a2 * b3 - a3 * b2)) / D, 3)
```

```
                Lab = Label(sim, text=f"x = {x}, y = {y}, z = {z}", font=("Arial",20), bg="yellow")
                Lab.place(x=10,y=470)
            except ZeroDivisionError:
                L3 = Label(sim, text="Error in the equation", font=("Arial",20), bg="yellow")
                L3.place(x=10,y=470)
        button1 = Button(sim, text="Submit", font=("Arial",10), padx=10, pady= 10, command=simultaneous_eqsolver)
        button1.place(x=50, y=420)
        button2 = Button(sim, text="clear", font=("Arial",10), padx=10, pady= 10, command=clear1)
        button2.place(x=150, y=420)
    three_variable()
    sim.mainloop()
```

4.ROOT FINDER

```
def root_finder():
    window = Tk()
    window.title("Root Finder")
    window.config(bg="dodgerblue3")
    window.geometry("800x500+100+100")
    L1 = Label(window, text="Root finder of polynomial equation", font=("Arial", 25), bg="dodgerblue3")
    L1.pack()


    def square():
```

```
L2 = Label(window, text="2nd degree:          x^2 +          x +          = 0", font=("Arial", 20), bg="dodgerblue3")

L2.place(x=10, y=50)

E1 = Entry(window, font=("Arial", 20), width=3)

E1.place(x=175, y=50)

E2 = Entry(window, font=("Arial", 20), width=3)

E2.place(x=300, y=50)

E3 = Entry(window, font=("Arial", 20), width=3)

E3.place(x=400, y=50)




def square_root():

    coeficcients = [int(E1.get()), int(E2.get()), int(E3.get())]

    roots = numpy.roots(coeficcients)

    L3 = Label(window, text=f"x = {roots[0].round(2)}, {roots[1].round(2)}", font=("Arial", 20),

               bg="dodgerblue3")

    L3.place(x=10, y=140)



    def clear_answer():

        L3.destroy()

    Button_sample = Button(window, text="Clear answer",font=("Arial", 15), command=clear_answer)

    Button_sample.place(x=400, y=140)
```

```
        def clear1():
            E1.delete(0, END)
            E2.delete(0, END)
            E3.delete(0, END)

        B1 = Button(window, text="Submit", font=("Arial", 15), command=square_root)
        B1.place(x=10, y=100)
        B2 = Button(window, text="Clear", font=("Arial", 15), command=clear1)
        B2.place(x=100, y=100)

    square()

    def cube():
        L2 = Label(window, text="3rd degree:          x^3 +         x^2 +         x +         = 0", font=("Arial", 20),
                   bg="dodgerblue3")
        L2.place(x=10, y=190)
        E1 = Entry(window, font=("Arial", 20), width=3)
        E1.place(x=175, y=190)
        E2 = Entry(window, font=("Arial", 20), width=3)
        E2.place(x=300, y=190)
        E3 = Entry(window, font=("Arial", 20), width=3)
        E3.place(x=430, y=190)
```

```
E4 = Entry(window, font=("Arial", 20), width=3)

E4.place(x=530, y=190)


def clear1():

    E1.delete(0, END)

    E2.delete(0, END)

    E3.delete(0, END)

    E4.delete(0, END)


def cubic_root():

    coeficcients = [int(E1.get()), int(E2.get()), int(E3.get()), int(E4.get())]

    roots = numpy.roots(coeficcients)

    L3 = Label(window, text=f"x = {roots[0].round(2)}, {roots[1].round(2)}, {roots[2].round(2)}",

               font=("Arial", 20), bg="dodgerblue3")

    L3.place(x=10, y=290)

    def clear_answer():

        L3.destroy()

    Button_sample =Button(window, text="Clear answer",font=("Arial", 15), command=clear_answer)

    Button_sample.place(x=550, y=290)

B1 = Button(window, text="Submit", font=("Arial", 15), command=cubic_root)

B1.place(x=10, y=240)

B2 = Button(window, text="Clear", font=("Arial", 15), command=clear1)
```

```
    B2.place(x=100, y=240)


cube()


def four_degree():
    L2 = Label(window, text="4th degree:      x^4 +      x^3 +      x^2 +      x +      = 0",
               font=("Arial", 20),
               bg="dodgerblue3")
    L2.place(x=10, y=340)
    E1 = Entry(window, font=("Arial", 20), width=3)
    E1.place(x=175, y=340)
    E2 = Entry(window, font=("Arial", 20), width=3)
    E2.place(x=300, y=340)
    E3 = Entry(window, font=("Arial", 20), width=3)
    E3.place(x=430, y=340)
    E4 = Entry(window, font=("Arial", 20), width=3)
    E4.place(x=550, y=340)
    E5 = Entry(window, font=("Arial", 20), width=3)
    E5.place(x=650, y=340)

    def clear1():
        E1.delete(0, END)
        E2.delete(0, END)
        E3.delete(0, END)
```

```
            E4.delete(0, END)

            E5.delete(0, END)



        def fourth_root():

            coeficcients = [int(E1.get()), int(E2.get()), int(E3.get()), int(E4.get()), int(E5.get())]

            roots = numpy.roots(coeficcients)

            L3 = Label(window,

                       text=f"x = {roots[0].round(2)}, {roots[1].round(2)}, {roots[2].round(2)}, {roots[3].round(2)}",

                       font=("Arial", 20), bg="dodgerblue3")

            L3.place(x=10, y=440)

            def clear_answer():

                L3.destroy()

            Button_sample =Button(window, text="Clear answer",font=("Arial", 15), command=clear_answer)

            Button_sample.place(x=700, y=440)



        B1 = Button(window, text="Submit", font=("Arial", 15), command=fourth_root)

        B1.place(x=10, y=390)

        B2 = Button(window, text="Clear", font=("Arial", 15), command=clear1)

        B2.place(x=100, y=390)

    four_degree()

    window.mainloop()
```

5.VIRTUAL CALCULATOR

```
def virtual_calculator():
    class Button:
        def __init__(self, pos, width, height, value):
            self.pos = pos
            self.width = width
            self.height = height
            self.value = value

        def draw(self, img):
            cv2.rectangle(img, self.pos, (self.pos[0] + self.width, self.pos[1] + self.height),
                          (140, 0, 0))
            cv2.rectangle(img, self.pos, (self.pos[0] + self.width, self.pos[1] + self.height),
                          (0, 0, 0), 4)
            cv2.putText(img, self.value, (self.pos[0] + 20, self.pos[1] + 30),
cv2.FONT_HERSHEY_PLAIN,
                        2, (255, 0, 0), 2)

        def checkclick(self, x, y):
            if self.pos[0] < x < self.pos[0] + self.width and \
                    self.pos[1] < y < self.pos[1] + self.height:
                cv2.rectangle(img, self.pos, (self.pos[0] + self.width, self.pos[1] + self.height),
                              (250, 250, 250), cv2.FILLED)
                cv2.rectangle(img, self.pos, (self.pos[0] + self.width, self.pos[1] + self.height),
                        (0, 0, 0), 4)

            cv2.putText(img, self.value, (self.pos[0] + 20, self.pos[1] + 30),
cv2.FONT_HERSHEY_PLAIN,
                        2, (0, 0, 0), 2)

            return True

        else:

            return False



    # Webcam

    cap = cv2.VideoCapture(0)

    cap.set(3, 1350)

    cap.set(4, 1000)

    detector = HandDetector(detectionCon=0.8, maxHands=1)



    # creating button

    buttonList1 = [['C', 'CE', "sqrt", "+", "cos", "tan", "sin"],

                   ['7', "8", "9", "-", "acos", "asin", "atan"],

                   ["4", "5", "6", "*", 'cosh', 'tanh', "sinh"],

                   ["1", "2", "3", "/", "sec", "cosec", "cot"],

                   ["0", ".", "2^x", "=", "x^y", "x^2", "x^3"],

                   ["(", ")", "pi", "2pi", "|x|", "e^x", "1/x"],

                   ["e", "log10", "ln", "rad", "deg", "10^x", "x!"]]

    buttonList = []

    for x in range(7):
```

```
    for y in range(7):
        xpos = x * 120 + 300
        ypos = y * 50 + 200
        buttonList.append(Button((xpos, ypos), 120, 50, buttonList1[y][x]))


myEquation = ""
delayCounter = 0
while True:
    success, img = cap.read()
    img = cv2.flip(img, 1)


    # detection of hand
    hands, img = detector.findHands(img, flipType=False)


    # draw all buttons
    cv2.rectangle(img, (300, 120), (500 + 640, 120 + 80), (0, 0, 0))
    cv2.rectangle(img, (300, 120), (500 + 640, 120 + 80), (0, 0, 0), 4)
    for button in buttonList:
        button.draw(img)


    # Check for Hand
    if hands:
        lmList = hands[0]["lmList"]
        length, _, img = detector.findDistance(lmList[8], lmList[12], img)
```

```
x, y = lmList[8]
if length < 70:
    for i, button in enumerate(buttonList):
        if button.checkclick(x, y) and delayCounter == 0:
            my_value = (buttonList1[int(i % 7)][int(i / 7)])
            try:
                if my_value == 'C':
                    myEquation = myEquation[0:len(myEquation) - 1]
                elif my_value == "CE":
                    myEquation = ""
                elif my_value == "sqrt":
                    myEquation = round(math.sqrt(eval(str(myEquation))), 4)
                elif my_value == 'pi':
                    myEquation = myEquation + str(round(math.pi, 4))
                elif my_value == 'cos':
                    myEquation = round(math.cos(math.radians(eval(str(myEquation)))), 4)
                elif my_value == 'tan':
                    myEquation = round(math.tan(math.radians(eval(str(myEquation)))), 4)
                elif my_value == 'sin':
                    myEquation = round(math.sin(math.radians(eval(str(myEquation)))), 4)
                elif my_value == 'cosh':
                    myEquation = round(math.cosh(eval(str(myEquation))), 4)
                elif my_value == 'tanh':
                    myEquation = round(math.tanh(eval(str(myEquation))), 4)
```

```
elif my_value == 'sinh':
    myEquation = round(math.sinh(eval(str(myEquation))), 4)
elif my_value == "x^y":
    myEquation = str(myEquation) + "**"
elif my_value == "x!":
    myEquation = math.factorial(int(eval(str(myEquation))))
elif my_value == 'log10':
    myEquation = round(math.log10(int(eval(str(myEquation)))), 4)
elif my_value == "ln":
    myEquation = round(math.log(int(eval(str(myEquation)))), 4)
elif my_value == "=":
    myEquation = round(float(eval(str(myEquation))), 4)
elif my_value == "e":
    myEquation = myEquation + str(round(math.e, 4))
elif my_value == "2pi":
    myEquation = str(myEquation) + str(round(2 * math.pi, 4))
elif my_value == "acos":
    myEquation = round(math.acos((eval(str(myEquation)))), 4)
elif my_value == "asin":
    myEquation = round(math.asin((eval(str(myEquation)))), 4)
elif my_value == "atan":
    myEquation = round(math.atan((eval(str(myEquation)))), 4)
elif my_value == "sec":
    myEquation = round(1 / math.cos(math.radians(eval(str(myEquation)))), 4)
```

```
elif my_value == "cosec":
    myEquation = round(1 / math.sin(math.radians(eval(str(myEquation)))), 4)
elif my_value == "cot":
    myEquation = round(1 / math.tan(math.radians(eval(str(myEquation)))), 4)
elif my_value == "e^x":
    myEquation = str(myEquation) + str(round(math.e, 4)) + "**"
elif my_value == "2^x":
    myEquation = str(myEquation) + "2**"
elif my_value == "x^2":
    myEquation = round((float(eval(str(myEquation))) ** 2), 4)
elif my_value == "x^3":
    myEquation = round((float(eval(str(myEquation))) ** 3), 4)
elif my_value == "|x|":
    myEquation = round(math.fabs(float(eval(str(myEquation)))), 4)
elif my_value == "1/x":
    myEquation = round(1 / float(eval(str(myEquation))), 4)
elif my_value == "10^x":
    myEquation = str(myEquation) + "10**"
elif my_value == "rad":
    myEquation = round(math.radians(float(eval(str(myEquation)))),4)
elif my_value == "deg":
    myEquation = round(math.degrees(float(eval(str(myEquation)))), 4)
else:
    myEquation = str(myEquation) + my_value
```

```
                except ZeroDivisionError:
                    myEquation = "Division by 0 invalid"
                except SyntaxError:
                    myEquation = "Syntax error"
                except TypeError:
                    myEquation = "Type Error"
                except ValueError:
                    myEquation = "Value Error "
                delayCounter = 1

# Avoid repitions:
if delayCounter != 0:
    delayCounter += 1
    if delayCounter > 10:
        delayCounter = 0
# Display equation
cv2.putText(img, str(myEquation), (300 + 20, 120 + 50), cv2.FONT_HERSHEY_PLAIN,
            2.5, (0, 0, 235), 2)

# Display image
cv2.imshow("image", img)


key = cv2.waitKey(1)
if key == ord("C"):
```

```
            cv2.destroyAllWindows()
            break
```

6.DERIVATIVE AND INTEGRAL

```
def calculus():
    window = Tk()
    window.title("DERIVATIVE AND INTEGRAL CALCULATOR")
    window.config(bg="yellow")
    window.geometry("700x600")
    L1 = Label(window,text="Derivative and integral value calculator", font=("Arial",25), bg="yellow")
    L1.pack()
    L2 = Label(window,text='1.Derivative: ', font=("Arial", 20), bg="yellow")
    L2.place(x=10,y=50)
    L3 = Label(window,text='Enter the expression in terms of x: ', font=("Arial",17), bg="yellow")
    L3.place(x=10,y=100)
    E1 = Entry(window, font=("Arial", 17),width=40)
    E1.place(x=10, y=130)

    def derivative():
        entry1 = E1.get()
        x = symbols('x')
        y = diff(entry1)
        y1 = str(y).replace("**", "^")
        Label1 = Label(window, text=f"Derivative = {y1}", font=("Arial", 17), bg="yellow")
    Label1.place(x=200, y=230)
B1 = Button(window,text="Submit",command=derivative, font=("Arial",15))
B1.place(x=10, y=180)
def clear1():
    global E2
    E1.delete(0,END)

B2 = Button(window, text="Clear", command=clear1, font=("Arial",15))
B2.place(x=100, y=180)
L4 = Label(window,text='2. Indefinite integral: ', font=("Arial",20), bg="yellow")
L4.place(x=10,y=280)
L5 = Label(window,text='Enter the expression in terms of x: ', font = ("Arial",17), bg="yellow")
L5.place(x=10,y=330)
E4 = Entry(window, font=("Arial", 17),width=40)
E4.place(x=10, y=380)

def integration():
    entry1 = E4.get()
    x = symbols('x')
    y = integrate(entry1)
    y1 = str(y).replace("**", "^")
    Label2 = Label(window, text=f"Integral = {y1}", font=("Arial", 17), width=40, bg="yellow")
    Label2.place(x=10,y=480)
def clear2():
        global E3
        E4.delete(0,END)
    B3 = Button(window,text="Submit",command=integration, font=("Arial",15))
    B3.place(x=10, y=430)
    B4 = Button(window, text="Clear", command=clear2, font=("Arial",15))
    B4.place(x=100,y=430)
    window.mainloop()
```

7.COMPLEX CALCULATOR

```
def complex_calculator():
    def click(value):
        ex = entryField.get()
        answer = ''

        try:

            if value == 'C':
                ex = ex[0:len(ex) - 1]  # 78
                entryField.delete(0, END)
                entryField.insert(0, ex)
                return
            elif value == 'CE':
                entryField.delete(0, END)
            elif value == 'π':
```

```
    answer = ex + str(round(math.pi, 4))
elif value == 'cos':
    answer1 = cmath.cos(complex(ex))
    answer = round(answer1.real, 4) + round(answer1.imag, 4) * 1j
elif value == 'tan':
    answer1 = cmath.tan(complex(ex))
    answer = round(answer1.real, 2) + round(answer1.imag, 2) * 1j
elif value == 'sin':
    answer1 = cmath.sin(complex(ex))
    answer = round(answer1.real, 2) + round(answer1.imag, 2) * 1j
elif value == 'acos':
    answer1 = cmath.acos(complex(ex))
    answer = round(answer1.real, 2) + round(answer1.imag, 2) * 1j
elif value == 'atan':
    answer1 = cmath.atan(complex(ex))
    answer = round(answer1.real, 2) + round(answer1.imag, 2) * 1j
elif value == 'asin':
    answer1 = cmath.asin(complex(ex))
    answer = round(answer1.real, 2) + round(answer1.imag, 2) * 1j
elif value == 'x\u00B2':
    answer1 = complex(ex) ** 2
    answer = round(answer1.real, 4) + round(answer1.imag, 4) * 1j
elif value == 'ln':
    answer1 = cmath.log(complex(ex))
    answer = round(answer1.real, 4) + round(answer1.imag, 4) * 1j
elif value == 'log₁₀':
    answer1 = cmath.log(complex(ex))
    answer = round(answer1.real, 4) + round(answer1.imag, 4) * 1j


elif value == '=':
    if "j" in ex:


        answer = ex
        for i in range(0, len(ex)):
            if ex[i] == "-" or ex[i] == "*" or ex[i] == "/" or ex[i] == "+" or ex[i] == "^":
                if ex[i - 1] == ")":
                    l, r = complex(ex[:i]), complex(ex[i + 1:])
                    if ex[i] == "+":
                        answer = l + r
                    elif ex[i] == "-":
                        answer = l - r
                    elif ex[i] == "*":
                        answer = l * r
                    elif ex[i] == "^":
                        answer = l ** r
                    else:
                        answer = l / r
                    answer = round(answer.real, 4) + round(answer.imag, 4) * 1j
    else:
        answer = eval(str(ex))
elif value == "real":
    answer = round(complex(ex).real, 4)
elif value == "imag":
    answer = round(complex(ex).imag, 4)
elif value == "phase":
    answer = round(cmath.phase(complex(ex)), 4)
elif value == "rect":
    list1 = ex.split(",")
    a = list1[0].strip("(")
    b = list1[1].strip(")")
    c = b.strip(" ")
    tuple1 = (a, c)
    answer1 = cmath.rect(float(tuple1[0]), float(tuple1[1]))
    answer = round(answer1.real, 4) + round(answer1.imag, 4) * 1j
elif value == "mod":
    answer1 = cmath.polar(complex(ex))
    answer = round(answer1[0], 4)
elif value == chr(247):
    answer = ex + "/"
elif value == "exp":
    answer1 = cmath.exp(complex(ex))
    answer = round(answer1.real, 4) + round(answer1.imag, 4) * 1j
```

```
        elif value == "√":
            answer1 = cmath.sqrt(complex(ex))
            answer = round(answer1.real, 4) + round(answer1.imag, 4) * 1j
        else:
            entryField.insert(END, str(value))
            return
        entryField.delete(0, END)
        entryField.insert(0, str(answer))

    except SyntaxError:
        entryField.delete(0, END)
        entryField.insert(0, "Syntax error")
    except ValueError:
        entryField.delete(0, END)
        entryField.insert(0, "Value Error")
    except ZeroDivisionError:
        entryField.delete(0, END)
        entryField.insert(0, "Zero division Error")

window = Tk()
window.title('Complex Calculator')
window.config(bg='light sea green')
window.geometry('680x486+100+100')
```

```
    entryField = Entry(window, font=('arial', 20, 'bold'), bg="turquoise", fg='black', bd=10,
relief=SUNKEN, width=30)
    entryField.grid(row=0, column=0, columnspan=8)


    button_list = ["C", "CE", "j", "+", "(", "real", "imag", "phase",
                   "1", "2", "3", "-", ")", "mod", "rect", ",",
                   "4", "5", "6", "*", "√", "π", "x\u00B2", "exp",
                   "7", "8", "9", chr(247), "ln", "sin", "cos", "tan",
                   "0", ".", "^", "=", "log₁₀", "asin", "acos", "atan"]
    rowvalue = 1
    columnvalue = 0
    for i in button_list:


        button = Button(window, width=5, height=2, bd=2, relief=SUNKEN, text=i, bg='turquoise',
fg='black',
                        font=('arial', 18, "bold"), activebackground='dodgerblue', command=lambda
button=i:
            click(button))
        button.grid(row=rowvalue, column=columnvalue, pady=1)
        columnvalue += 1
        if columnvalue > 7:
            rowvalue += 1
            columnvalue = 0

    window.mainloop()
```

8.VOLUME FINDER

```
def volume_finder():

    Vol = Tk()

    Vol.title("VOLUME FINDER")

    Vol.config(bg="cyan")

    Vol.geometry("680x506+100+100")

    L1 = Label(Vol, text="Volume Finder", font=("Arial", 30), bg="cyan")

    L1.pack()

    def Cuboid():

        def clear():

            E1.delete(0, END)

            E2.delete(0, END)



        L2 = Label(Vol, text="Length=", font=("Arial", 20), bg="cyan")

        L2.place(x=10, y=150)

        L3 = Label(Vol, text="Breadth=", font=("Arial", 20), bg="cyan")

        L3.place(x=225, y=150)

        L4 = Label(Vol, text="Height=", font=("Arial", 20), bg="cyan")

        L4.place(x=450, y=150)

        E1 = Entry(Vol, width=6, font=("Arial", 20))

        E1.place(x=120, y=150)

        E2 = Entry(Vol, width=6, font=("Arial", 20))

        E2.place(x=350, y=150)
```

```
E3 = Entry(Vol, width=6, font=("Arial", 20))
E3.place(x=555, y=150)

def calculate():
    a1 = int(E1.get())
    a2 = int(E2.get())
    a3 = int(E3.get())
    A = a1 * a2 * a3
    def clear_everything():
        E1.destroy()
        E2.destroy()
        E3.destroy()
        Lab.destroy()
        L2.destroy()
        L3.destroy()
        L4.destroy()
        button1.destroy()
        button2.destroy()
    Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg = "cyan")
    Lab.place(x=10, y=250)
    Button_sample = Button(Vol, text="Clear", command=clear_everything)
    Button_sample.place(x=300, y=80)
button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
```

```
    button1.place(x=50, y=200)

    button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

    button2.place(x=150, y=200)


def Cube():

    def clear():

        E1.delete(0, END)


    L2 = Label(Vol, text="Length of Side=", font=("Arial", 20), bg="cyan")

    L2.place(x=10, y=150)

    E1 = Entry(Vol, width=6, font=("Arial", 20))

    E1.place(x=250, y=150)


    def calculate():

        a1 = int(E1.get())

        A = a1 ** 3

        def clear_everything():

            L2.destroy()

            E1.destroy()

            Lab.destroy()

            button1.destroy()

            button2.destroy()

        Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")

        Lab.place(x=10, y=250)
```

```
            Button_sample = Button(Vol, text="Clear", command=clear_everything)
            Button_sample.place(x=300, y=80)

        button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
        button1.place(x=50, y=200)
        button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
        button2.place(x=150, y=200)

    def Cone():
        def clear():
            E1.delete(0, END)
            E2.delete(0, END)

        L2 = Label(Vol, text="Base Radius=", font=("Arial", 20), bg="cyan")
        L2.place(x=10, y=150)
        L3 = Label(Vol, text="Height=", font=("Arial", 20), bg="cyan")
        L3.place(x=340, y=150)
        E1 = Entry(Vol, width=6, font=("Arial", 20))
        E1.place(x=220, y=150)
        E2 = Entry(Vol, width=6, font=("Arial", 20))
        E2.place(x=450, y=150)

        def calculate():
            a1 = int(E1.get())

            a2 = int(E2.get())

            A = round((pi*a1*a1*a2)/3, 4)

            Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")

            Lab.place(x=10, y=250)

            def clear_everything():

                L2.destroy()

                L3.destroy()

                Lab.destroy()

                E1.destroy()

                E2.destroy()

                button1.destroy()

                button2.destroy()

            Button_sample = Button(Vol, text="Clear", command=clear_everything)

            Button_sample.place(x=300, y=80)

        button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)

        button1.place(x=50, y=200)

        button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

        button2.place(x=150, y=200)


    def Cylinder():

        def clear():

            E1.delete(0, END)
            E2.delete(0, END)


        L2 = Label(Vol, text="Radius=", font=("Arial", 20), bg="cyan")

        L2.place(x=10, y=150)

        L3 = Label(Vol, text="Height=", font=("Arial", 20), bg="cyan")

        L3.place(x=230, y=150)

        E1 = Entry(Vol, width=6, font=("Arial", 20))

        E1.place(x=120, y=150)

        E2 = Entry(Vol, width=6, font=("Arial", 20))

        E2.place(x=330, y=150)


        def calculate():

            a1 = int(E1.get())

            a2 = int(E2.get())

            A = round((pi*a1*a1*a2), 4)

            def clear_everything():

                L2.destroy()

                L3.destroy()

                Lab.destroy()

                E1.destroy()

                E2.destroy()

                button1.destroy()

                button2.destroy()
```

```
            Button_sample = Button(Vol, text="Clear", command=clear_everything)

            Button_sample.place(x=300, y=80)

            Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")

            Lab.place(x=10, y=250)


        button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)

        button1.place(x=50, y=200)

        button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

        button2.place(x=150, y=200)


    def Square_Pyramid():

        def clear():

            E1.delete(0, END)

            E2.delete(0, END)


        L2 = Label(Vol, text="Base Side=", font=("Arial", 20), bg="cyan")

        L2.place(x=10, y=150)

        L3 = Label(Vol, text="Height=", font=("Arial", 20), bg="cyan")

        L3.place(x=300, y=150)

        E1 = Entry(Vol, width=6, font=("Arial", 20))

        E1.place(x=180, y=150)

        E2 = Entry(Vol, width=6, font=("Arial", 20))

        E2.place(x=450, y=150)
```

```
def calculate():
    a1 = int(E1.get())
    a2 = int(E2.get())
    A = round((a1**2*a2)/3, 4)
    Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")
    Lab.place(x=10, y=250)
    def clear_everything():
        L2.destroy()
        L3.destroy()
        Lab.destroy()
        E1.destroy()
        E2.destroy()
        button1.destroy()
        button2.destroy()

    Button_sample = Button(Vol, text="Clear", command=clear_everything)
    Button_sample.place(x=300, y=80)

button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
button1.place(x=50, y=200)
button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
button2.place(x=150, y=200)
```

```
def Torus():
    def clear():
        E1.delete(0, END)
        E2.delete(0, END)

    L2 = Label(Vol, text="Major Radius=", font=("Arial", 20), bg="cyan")
    L2.place(x=10, y=150)
    L3 = Label(Vol, text="Minor Radius=", font=("Arial", 20), bg="cyan")
    L3.place(x=340, y=150)
    E1 = Entry(Vol, width=6, font=("Arial", 20))
    E1.place(x=215, y=150)
    E2 = Entry(Vol, width=6, font=("Arial", 20))
    E2.place(x=550, y=150)

    def calculate():
        a1 = int(E1.get())
        a2 = int(E2.get())
        A = round((2*pi*a1)*(pi*a2**2),4)
        Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")
        Lab.place(x=10, y=250)
        def clear_everything():
            L2.destroy()
            L3.destroy()
```

```
                Lab.destroy()

                E1.destroy()

                E2.destroy()

                button1.destroy()

                button2.destroy()


            Button_sample = Button(Vol, text="Clear", command=clear_everything)

            Button_sample.place(x=300, y=80)


        button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)

        button1.place(x=50, y=200)

        button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

        button2.place(x=150, y=200)


    def Sphere():

        def clear():

            E1.delete(0, END)


        L2 = Label(Vol, text="Radius=", font=("Arial", 20), bg="cyan")

        L2.place(x=10, y=150)

        E1 = Entry(Vol, width=6, font=("Arial", 20))

        E1.place(x=150, y=150)
        def calculate():
            a1 = int(E1.get())
            A = round((4*pi*a1**3)/3, 4)
            def clear_everything():
                L2.destroy()
                E1.destroy()
                Lab.destroy()
                button1.destroy()
                button2.destroy()
            Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")
            Lab.place(x=10, y=250)
            Button_sample = Button(Vol, text="Clear", command=clear_everything)
            Button_sample.place(x=300, y=80)

        button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
        button1.place(x=50, y=200)
        button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
        button2.place(x=150, y=200)

    def Hemisphere():
        def clear():
            E1.delete(0, END)
```

```
L2 = Label(Vol, text="Radius=", font=("Arial", 20), bg="cyan")
L2.place(x=10, y=150)
E1 = Entry(Vol, width=6, font=("Arial", 20))
E1.place(x=150, y=150)


def calculate():
    a1 = int(E1.get())
    A = round((2*pi*a1**3)/3, 4)
    def clear_everything():
        L2.destroy()
        E1.destroy()
        Lab.destroy()
        button1.destroy()
        button2.destroy()
    Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")
    Lab.place(x=10, y=250)
    Button_sample = Button(Vol, text="Clear", command=clear_everything)
    Button_sample.place(x=300, y=80)


button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
button1.place(x=50, y=200)
button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
button2.place(x=150, y=200)
```

```
def Tetrahedron():

    def clear():

        E1.delete(0, END)



    L2 = Label(Vol, text="Side Length=", font=("Arial", 20), bg="cyan")

    L2.place(x=10, y=150)

    E1 = Entry(Vol, width=6, font=("Arial", 20))

    E1.place(x=200, y=150)



    def calculate():

        a1 = int(E1.get())

        A = round(((a1 ** 3) / 6) * sqrt(2), 4)

        def clear_everything():

            L2.destroy()

            E1.destroy()

            Lab.destroy()

            button1.destroy()

            button2.destroy()

        Lab = Label(Vol, text=f"Volume={A}", font=("Arial", 20), bg="cyan")

        Lab.place(x=10, y=250)

        Button_sample = Button(Vol, text="Clear", command=clear_everything)

        Button_sample.place(x=300, y=80)
    button1 = Button(Vol, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)

    button1.place(x=50, y=200)

    button2 = Button(Vol, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

    button2.place(x=150, y=200)


def show():


    opt = clicked.get()

    if opt == "1.Cuboid":

        Cuboid()

    if opt == "2.Cube":

        Cube()

    if opt == "3.Cone":

        Cone()

    if opt == "4.Cylinder":

        Cylinder()

    if opt == "5.Square_Pyramid":

        Square_Pyramid()

    if opt == "6.Torus":

        Torus()

    if opt == "7.Sphere":

        Sphere()

    if opt == "8.Hemisphere":
        Hemisphere()
    if opt == "9.Tetrahedron":
        Tetrahedron()

option = ["1.Cuboid", "2.Cube", "3.Cone", "4.Cylinder", "5.Square_Pyramid", "6.Torus", "7.Sphere",
          "8.Hemisphere", "9.Tetrahedron"]
clicked = StringVar(Vol)
clicked.set("Select an option")
drop = OptionMenu(Vol, clicked, *option)
drop.place(x=10, y=80)
button = Button(Vol, text="Submit", command=show)
button.place(x=200, y=80)
button_clear = Button(Vol, text="Clear", command=NONE)
button_clear.place(x=300, y=80)

Vol.mainloop()
```

9.Area Finder

```
def Area_finder():
    Are = Tk()
    Are.title("AREA FINDER")
    Are.config(bg="orange")
    Are.geometry("750x506+100+100")
    L1 = Label(Are, text="Area Finder", font=("Arial", 30), bg="orange")
```

```
L1.pack()



def Rectangle():
    def clear():
        E1.delete(0, END)
        E2.delete(0, END)

    L2 = Label(Are, text="Length=", font=("Arial", 20), bg="orange")
    L2.place(x=10, y=150)
    L3 = Label(Are, text="Breadth=", font=("Arial", 20), bg="orange")
    L3.place(x=300, y=150)
    E1 = Entry(Are, width=6, font=("Arial", 20))
    E1.place(x=150, y=150)
    E2 = Entry(Are, width=6, font=("Arial", 20))
    E2.place(x=450, y=150)

    def calculate():
        a1 = int(E1.get())
        a2 = int(E2.get())
        A = a1 * a2
        Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")
        Lab.place(x=10, y=250)
        def clear_everything():
            L2.destroy()
            L3.destroy()
            E1.destroy()
            E2.destroy()
            Lab.destroy()
            button1.destroy()
            button2.destroy()

        Button_sample = Button(Are, text="Clear", command=clear_everything)
        Button_sample.place(x=340, y=80)
    button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
    button1.place(x=50, y=200)
    button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
    button2.place(x=150, y=200)

def Square():
    def clear():
        E1.delete(0, END)

    L2 = Label(Are, text="Length of Side=", font=("Arial", 20), bg="orange")
    L2.place(x=10, y=150)
    E1 = Entry(Are, width=6, font=("Arial", 20))
```

```
        E1.place(x=220, y=150)

        def calculate():
            a1 = int(E1.get())
            A = a1 ** 2
            Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")
            Lab.place(x=10, y=250)
            def clear_everything():
                L2.destroy()
                E1.destroy()
                Lab.destroy()
                button1.destroy()
                button2.destroy()
            Button_sample = Button(Are, text="Clear", command=clear_everything)
            Button_sample.place(x=340, y=80)
        button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
        button1.place(x=50, y=200)
        button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
        button2.place(x=150, y=200)

    def Triangle1():
        def clear():
            E1.delete(0, END)
            E2.delete(0, END)


        L2 = Label(Are, text="Base=", font=("Arial", 20), bg="orange")

        L2.place(x=10, y=150)

        L3 = Label(Are, text="Height=", font=("Arial", 20), bg="orange")

        L3.place(x=220, y=150)

        E1 = Entry(Are, width=6, font=("Arial", 20))

        E1.place(x=100, y=150)

        E2 = Entry(Are, width=6, font=("Arial", 20))

        E2.place(x=330, y=150)


        def calculate():

            a1 = int(E1.get())

            a2 = int(E2.get())

            A =  (a1 * a2)/2

            def clear_everything():

                L2.destroy()

                L3.destroy()

                E1.destroy()

                E2.destroy()

                Lab.destroy()

                button1.destroy()

                button2.destroy()
```

```
            Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")

            Lab.place(x=10, y=250)

            Button_sample = Button(Are, text="Clear", command=clear_everything)

            Button_sample.place(x=340, y=80)

        button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)

        button1.place(x=50, y=200)

        button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

        button2.place(x=150, y=200)


    def Triangle2():

        def clear():

            E1.delete(0, END)

            E2.delete(0, END)

            E3.delete(0, END)


        L2 = Label(Are, text="Side-1=", font=("Arial", 20), bg="orange")

        L2.place(x=10, y=150)

        L3 = Label(Are, text="Side-2=", font=("Arial", 20), bg="orange")

        L3.place(x=220, y=150)

        L4 = Label(Are, text="Side-3=", font=("Arial", 20), bg="orange")

        L4.place(x=430, y=150)

        E1 = Entry(Are, width=6, font=("Arial", 20))

        E1.place(x=110, y=150)
```

```
E2 = Entry(Are, width=6, font=("Arial", 20))

E2.place(x=320, y=150)

E3 = Entry(Are, width=6, font=("Arial", 20))

E3.place(x=530, y=150)


def calculate():

    a1 = int(E1.get())

    a2 = int(E2.get())

    a3 = int(E3.get())

    s = (a1 + a2 + a3)/2

    A = round(sqrt(abs(s * (s - a1) * (s - a2) * (s - a3))), 4)

    Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")

    Lab.place(x=10, y=250)


    def clear_everything():

        E1.destroy()

        E2.destroy()

        E3.destroy()

        Lab.destroy()

        L2.destroy()

        L3.destroy()

        L4.destroy()

        button1.destroy()

        button2.destroy()
```

```
            Button_sample = Button(Are, text="Clear", command=clear_everything)
            Button_sample.place(x=340, y=80)
        button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
        button1.place(x=50, y=200)
        button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
        button2.place(x=150, y=200)


    def Trapezium():
        def clear():
            E1.delete(0, END)
            E2.delete(0, END)
            E3.delete(0, END)


        L2 = Label(Are, text="Base Length=", font=("Arial", 20), bg="orange")
        L2.place(x=10, y=150)
        L3 = Label(Are, text="Height=", font=("Arial", 20), bg="orange")
        L3.place(x=285, y=150)
        L4 = Label(Are, text="Top Length=", font=("Arial", 20), bg="orange")
        L4.place(x=490, y=150)
        E1 = Entry(Are, width=6, font=("Arial", 20))
        E1.place(x=190, y=150)
        E2 = Entry(Are, width=6, font=("Arial", 20))
```

```
E2.place(x=390, y=150)

E3 = Entry(Are, width=6, font=("Arial", 20))

E3.place(x=650, y=150)


def calculate():

    a1 = int(E1.get())

    a2 = int(E2.get())

    a3 = int(E3.get())

    A = round(((a1 + a3)/2) * a2, 4)

    Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")

    Lab.place(x=10, y=250)

    def clear_everything():

        E1.destroy()

        E2.destroy()

        E3.destroy()

        Lab.destroy()

        L2.destroy()

        L3.destroy()

        L4.destroy()

        button1.destroy()

        button2.destroy()


    Button_sample = Button(Are, text="Clear", command=clear_everything)

    Button_sample.place(x=340, y=80)
```

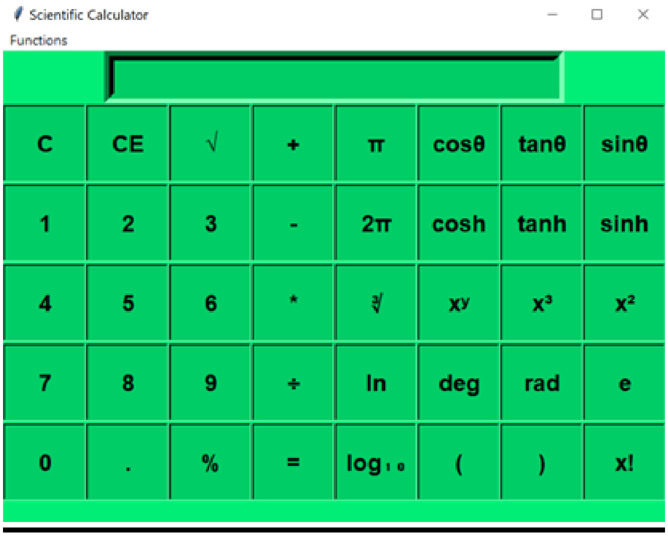
```
        button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)

        button1.place(x=50, y=200)

        button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

        button2.place(x=150, y=200)


    def Parallelogram():

        def clear():

            E1.delete(0, END)

            E2.delete(0, END)


        L2 = Label(Are, text="Length=", font=("Arial", 20), bg="orange")

        L2.place(x=10, y=150)

        L3 = Label(Are, text="Breadth=", font=("Arial", 20), bg="orange")

        L3.place(x=300, y=150)

        E1 = Entry(Are, width=6, font=("Arial", 20))

        E1.place(x=150, y=150)

        E2 = Entry(Are, width=6, font=("Arial", 20))

        E2.place(x=450, y=150)


        def calculate():

            a1 = int(E1.get())

            a2 = int(E2.get())
            A = a1 * a2
            Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")
            Lab.place(x=10, y=250)
            def clear_everything():
                L2.destroy()
                L3.destroy()
                E1.destroy()
                E2.destroy()
                Lab.destroy()
                button1.destroy()
                button2.destroy()

            Button_sample = Button(Are, text="Clear", command=clear_everything)
            Button_sample.place(x=340, y=80)

        button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
        button1.place(x=50, y=200)
        button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
        button2.place(x=150, y=200)

    def Circle():
        def clear():
            E1.delete(0, END)
```

```
L2 = Label(Are, text="Radius=", font=("Arial", 20), bg="orange")
L2.place(x=10, y=150)
E1 = Entry(Are, width=6, font=("Arial", 20))
E1.place(x=150, y=150)


def calculate():
    a1 = int(E1.get())
    A = round((pi*a1**2), 4)
    Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")
    Lab.place(x=10, y=250)
    def clear_everything():
        L2.destroy()
        E1.destroy()
        Lab.destroy()
        button1.destroy()
        button2.destroy()
    Button_sample = Button(Are, text="Clear", command=clear_everything)
    Button_sample.place(x=340, y=80)


button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)
button1.place(x=50, y=200)
button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)
```

```
    button2.place(x=150, y=200)


def SemiCircle():
    def clear():
        E1.delete(0, END)


    L2 = Label(Are, text="Radius=", font=("Arial", 20), bg="orange")
    L2.place(x=10, y=150)
    E1 = Entry(Are, width=6, font=("Arial", 20))
    E1.place(x=150, y=150)


    def calculate():
        a1 = int(E1.get())
        A = round((pi*a1**2)/2, 4)
        Lab = Label(Are, text=f"Area = {A}", font=("Arial", 20), bg="orange")
        Lab.place(x=10, y=250)
        def clear_everything():
            L2.destroy()
            E1.destroy()
            Lab.destroy()
            button1.destroy()
            button2.destroy()
        Button_sample = Button(Are, text="Clear", command=clear_everything)
        Button_sample.place(x=340, y=80)
        button1 = Button(Are, text="Submit", font=("Arial", 10), padx=10, pady=7, command=calculate)

        button1.place(x=50, y=200)

        button2 = Button(Are, text="clear", font=("Arial", 10), padx=10, pady=7, command=clear)

        button2.place(x=150, y=200)


    def show():

        opt = clicked.get()

        if opt == "1.Rectangle":

            Rectangle()

        if opt == "2.Square":

            Square()

        if opt == "3.Triangle(2-sides known)":

            Triangle1()

        if opt == "4.triangle(3-sides known)":

            Triangle2()

        if opt == "5.Trapezium":

            Trapezium()

        if opt == "6.Parallelogram":

            Parallelogram()

        if opt == "7.Circle":

            Circle()

        if opt == "8.Semi-Circle":

            SemiCircle()
```

```
option = ["1.Rectangle", "2.Square", "3.Triangle(2-sides known)", "4.triangle(3-sides known)",
          "5.Trapezium", "6.Parallelogram", "7.Circle", "8.Semi-Circle"]
clicked = StringVar(Are)
clicked.set("Select an option")
drop = OptionMenu(Are, clicked, *option)
drop.place(x=10, y=80)
button = Button(Are, text="Submit", padx=10, command=show)
button.place(x=240, y=80)
button_clear = Button(Are, text="Clear", command=NONE)
button_clear.place(x=340,y=80)
Are.mainloop()
```

RESULT AND ANALYSIS

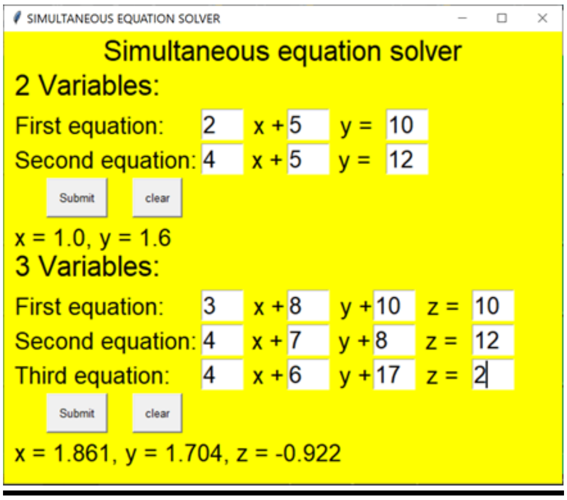


SCIENTIFIC CALCULATOR[2]

- This interface works as a basic scientific calculator which contains Trigonometric functions, hyperbolic functions, logarithmic, exponential and many more functions alongside basic arithmetic functions.
- Modules used: math, tkinter

SIMULTANEOUS EQUATION SOLVER

- This interface allows the user to solve simultaneous equations having 2 and 3 variables respectively.
- Modules used: tkinter



SIMULTANEOUS EQUATION SOLVER

Simultaneous equation solver

2 Variables:

First equation: 2 x + 5 y = 10

Second equation: 4 x + 5 y = 12

x = 1.0, y = 1.6

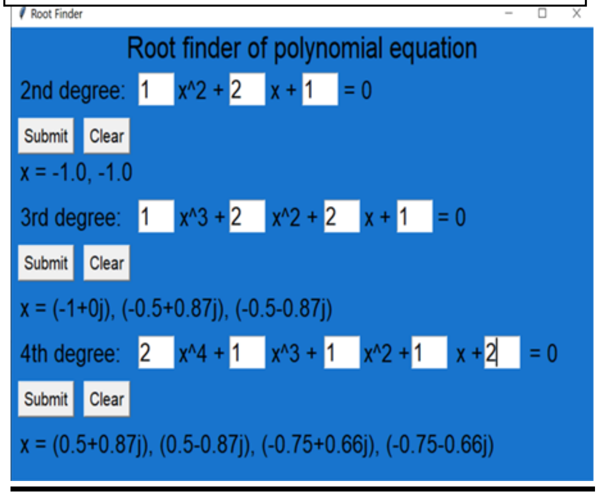
3 Variables:

First equation: 3 x + 8 y + 10 z = 10

Second equation: 4 x + 7 y + 8 z = 12

Third equation: 4 x + 6 y + 17 z = 2

x = 1.861, y = 1.704, z = -0.922



Root Finder

Root finder of polynomial equation

2nd degree: 1 x^2 + 2 x + 1 = 0

x = -1.0, -1.0

3rd degree: 1 x^3 + 2 x^2 + 2 x + 1 = 0

x = (-1+0j), (-0.5+0.87j), (-0.5-0.87j)

4th degree: 2 x^4 + 1 x^3 + 1 x^2 +1 x +2 = 0

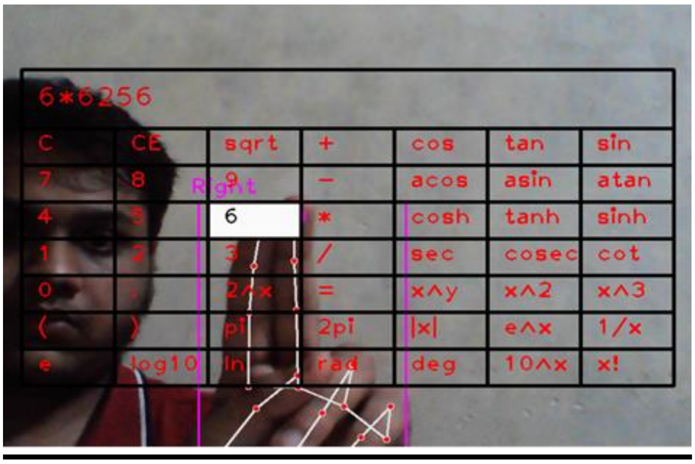
x = (0.5+0.87j), (0.5-0.87j), (-0.75+0.66j), (-0.75-0.66j)

ROOT FINDER[4]

- This interface allows the user to find the roots of a given quadratic, cubic or a quartic (degree 4) polynomial.
- Modules Used: tkinter, numpy

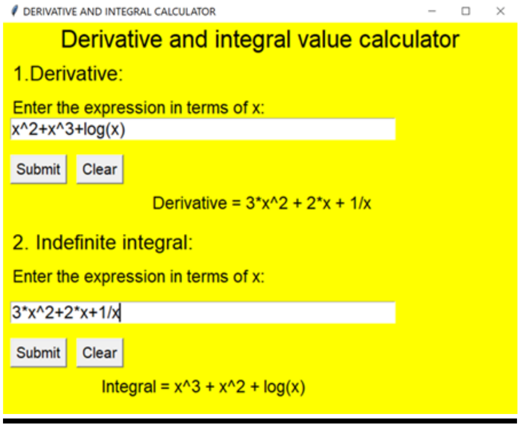
VIRTUAL CALCULATOR[1]

- This interface allows the user to use the scientific calculator using hand-gestures.
- Modules used: cv2, cvzone.HandTrackingModule, math



6*6256

C	CE	sqrt	+	cos	tan	sin
7	8	9	–	acos	asin	atan
4	5	6	*	cosh	tanh	sinh
1	2	3	/	sec	cosec	cot
0	.	2^x	=	x^y	x^2	x^3
(	)	pi	2pi	|x|	e^x	1/x
e	log10	ln	rad	deg	10^x	x!


DERIVATIVE AND INTEGRAL CALCULATOR

Derivative and integral value calculator

1.Derivative:

Enter the expression in terms of x:

x^2+x^3+log(x)

Derivative = 3*x^2 + 2*x + 1/x

2. Indefinite integral:

Enter the expression in terms of x:

3*x^2+2*x+1/x

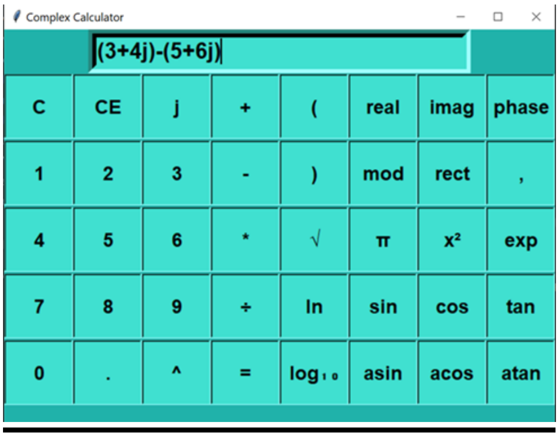
Integral = x^3 + x^2 + log(x)

DERIVATIVE AND INTEGRAL[6]

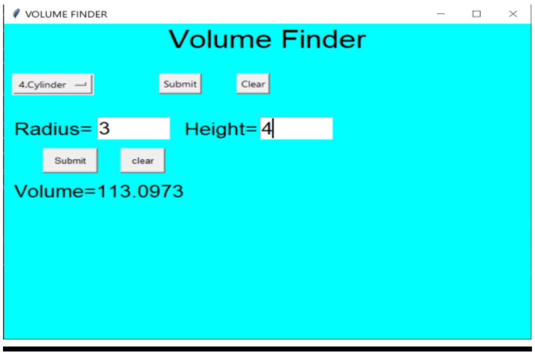
- This interface allows the user to find the derivative and integral of a given equations which are in terms of ‘x’.
- Modules used: tkinter, sympy

COMPLEX CALCULATOR[3]

- This interface allows the user to perform trigonometric, inverse trigonometric, logarithmic, complex and basic arithmetic operations on complex numbers.
- Modules used: tkinter, cmath, math


Complex Calculator

(3+4j)-(5+6j)


VOLUME FINDER

Volume Finder

4.Cylinder

Radius= 3 Height= 4

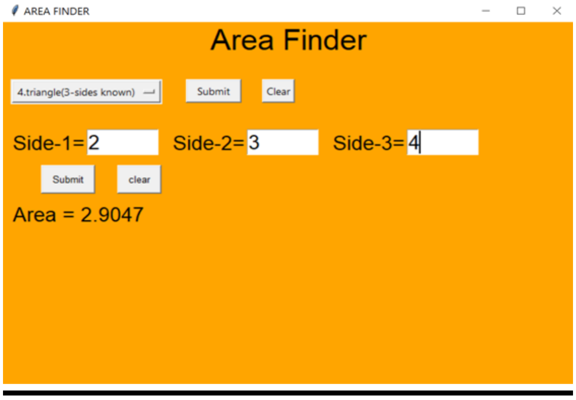
Volume=113.0973

VOLUME FINDER[5]

- This interface allows the user to find Area of different 2-D figures such as square, rectangle, triangle, circle etc.
- Modules used: tkinter, math

AREA FINDER[5]

- This interface allows the user to find Volume of different 3-D figures such as cuboid, cube, sphere, cylinder etc.
- Modules used: tkinter, math


AREA FINDER

Area Finder

4.triangle(3-sides known)

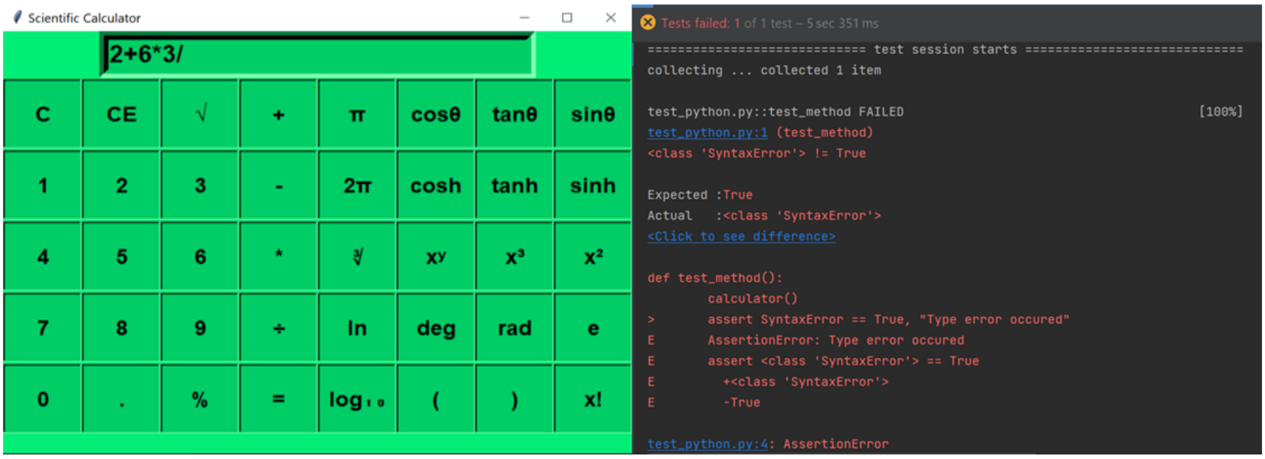
Side-1= 2 Side-2= 3 Side-3= 4

Area = 2.9047

TESTING

Testing -1: Scientific Calculator

- Usually the expression is expected to end with a proper integer or a floating value.
- So, here the testing case is when the expression ends with an operator i.e a SYNTAX ERROR.
- The issue is sorted by using try and except method.

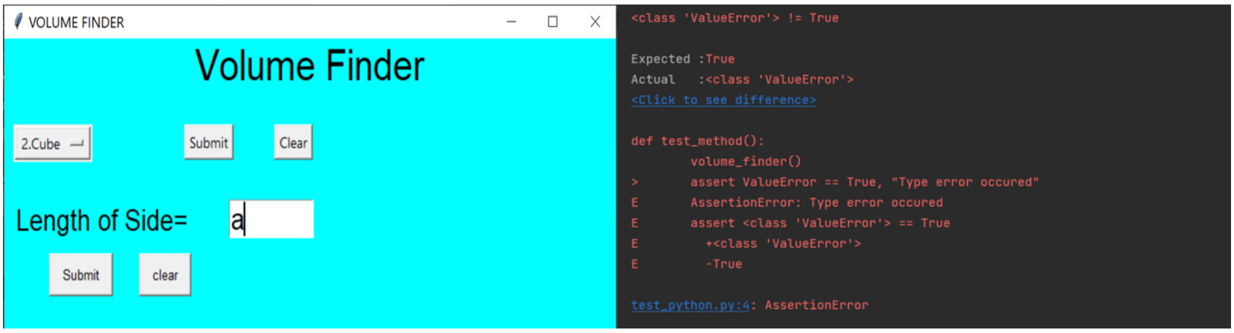


2+6*3/

```
Tests failed: 1 of 1 test – 5 sec 351 ms
============================ test session starts =============================
collecting ... collected 1 item

test_python.py::test_method FAILED                                      [100%]
test_python.py:1 (test_method)
<class 'SyntaxError'> != True

Expected :True
Actual   :<class 'SyntaxError'>
<Click to see difference>

def test_method():
        calculator()
>       assert SyntaxError == True, "Type error occured"
E       AssertionError: Type error occured
E       assert <class 'SyntaxError'> == True
E         +<class 'SyntaxError'>
E         -True

test_python.py:4: AssertionError
```

2. Testing-2: Volume Finder

- Here this module is being tested for cases when a non-numeric value is entered, an assertion error is raised which says "Type error encountered".
- This makes it compulsory for the user to input integer values in Entry widgets.
- This error can be rectified by using "try and except" method.

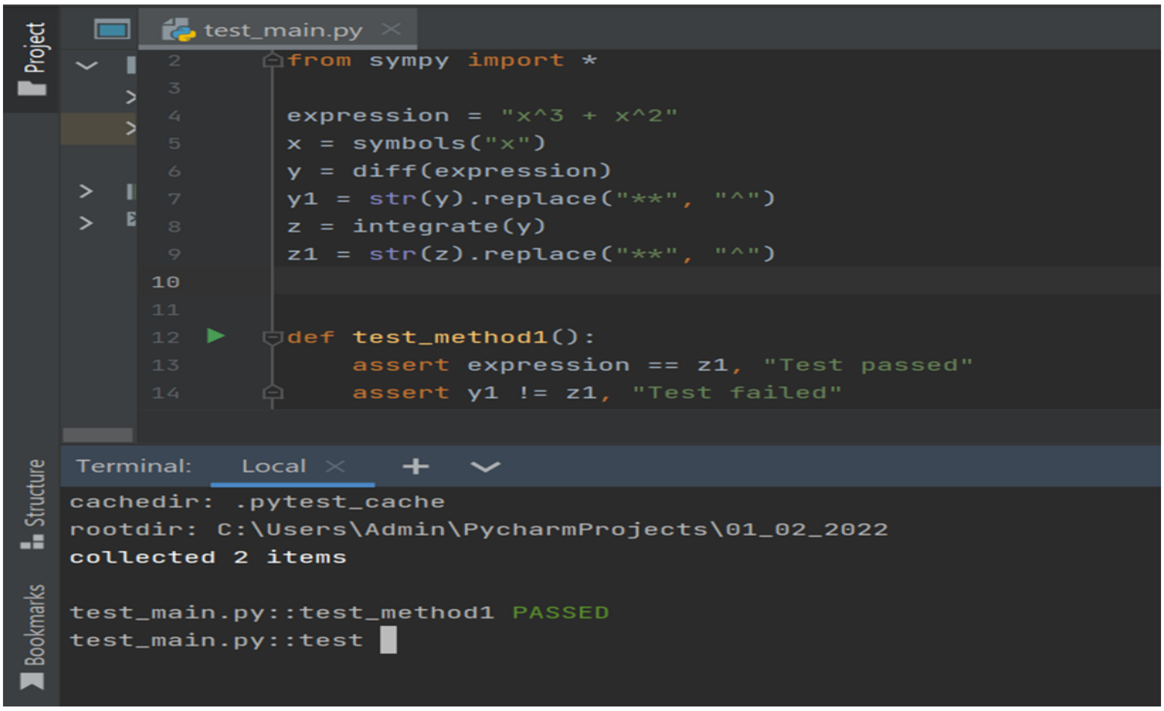


Length of Side= a

```
<class 'ValueError'> != True

Expected :True
Actual   :<class 'ValueError'>
<Click to see difference>

def test_method():
        volume_finder()
>       assert ValueError == True, "Type error occured"
E       AssertionError: Type error occured
E       assert <class 'ValueError'> == True
E         +<class 'ValueError'>
E         -True

test_python.py:4: AssertionError
```

Testing 3-Derivative

- Here the testing case is when an expression is entered whose derivative is to be found, the derivative is said to be correct only if the integral of derivative gives us the same expression back.
- In this case of our testing,



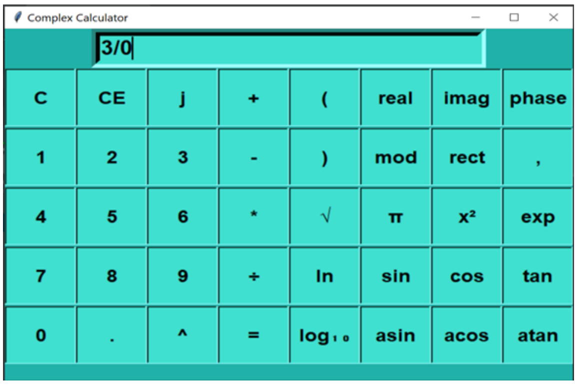
```python
from sympy import *

expression = "x^3 + x^2"
x = symbols("x")
y = diff(expression)
y1 = str(y).replace("**", "^")
z = integrate(y)
z1 = str(z).replace("**", "^")


def test_method1():
    assert expression == z1, "Test passed"
    assert y1 != z1, "Test failed"
```

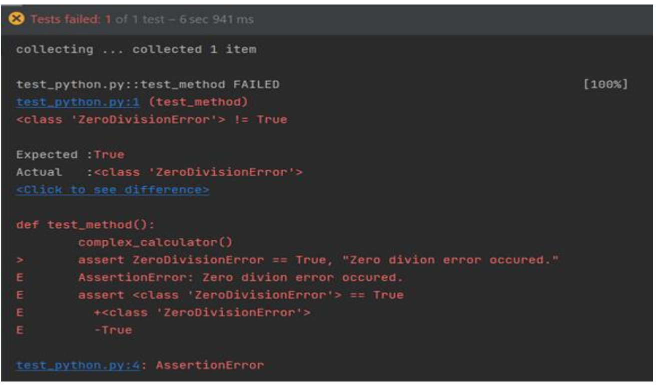
```
cachedir: .pytest_cache
rootdir: C:\Users\Admin\PycharmProjects\01_02_2022
collected 2 items

test_main.py::test_method1 PASSED
test_main.py::test
```

Testing-4:Complex Calculator:

- We know that whenever an integer is divided by “0” the answer is not defined.
- Hence the testing case here is division by zero.
- When an integer is divided by 0, a “ZeroDivisionError” is encountered.

This is solved using try and except method





Tests failed: 1 of 1 test – 6 sec 941 ms

```
collecting ... collected 1 item

test_python.py::test_method FAILED                                       [100%]
test_python.py:1 (test_method)
<class 'ZeroDivisionError'> != True

Expected :True
Actual   :<class 'ZeroDivisionError'>
<Click to see difference>

def test_method():
        complex_calculator()
>       assert ZeroDivisionError == True, "Zero divion error occured."
E       AssertionError: Zero divion error occured.
E       assert <class 'ZeroDivisionError'> == True
E         +<class 'ZeroDivisionError'>
E         -True

test_python.py:4: AssertionError
```

CONCLUSION:

- When a student sits to solve a problem the basic problems he faces are:
- How to calculate complex arithmetical and trigonometric operations?
- How to find integral and derivative of complex equations?
- How to find roots of 3rd or 4th degree polynomial?
- How to find Volumes or areas of different figures? Etc.
- So our project is focused in providing solutions to these problems by integrating all these functionalities in a single program. This will allow the student to focus on solving the problem and not on performing complex calculations. This will also help the student to save his time and use it judicially.

Future Enhancements:

- Certain functionalities such as displaying the graph of expressions, Unit converter, base converter can be added to make it more user friendly.

References:

[1] https://youtu.be/DZMJ77akgec

[2] geeksforgeeks.org/scientific-gui-calculator-using-tkinter-in-python/

[3] https://www.geeksforgeeks.org/complex-numbers-in-python-set-1-introduction/

[4] https://docs.scipy.org/doc/scipy/reference/optimize.html

[5] https://www.geeksforgeeks.org/tkinter-optionmenu-widget/

[6] https://www.codegrepper.com/code-examples/python/derivative+calculator+python