A

PROJECT

ON

"Math Solver"

Submitted to

Shiksha Mandal's

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR

(AUTONOMOUS)

In the Partial Fulfillment of

B.Com. (Computer Application) Final Year

Submitted by

Prathmesh P. Gulhane

Under the Guidance of

Pravin J. Yadao

Shiksha Mandal's

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR

(AUTONOMOUS)

2022-2023

Shiksha Mandal's

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR (AUTONOMOUS)

CERTIFICATE

(2022 - 2023)

This is to certify that **Mr. Prathmesh P. Gulhane** has completed their project on the topic of **Math Solver** prescribed by G. S. College of Commerce & Economics, Nagpur (Autonomous) for B.Com. (Computer Application) – Semester-VI.

Date:

Place: Nagpur

Pravin J. Yadao

Project Guide

External Examiner **Internal Examiner**

ACKNOWLEDGEMENT

We take this opportunity to express our deep gratitude and whole hearted thanks to project guide Prof. Pravin Yadao, Coordinator for his guidance throughout this work. We are very much thankful to him for his constant encouragement, support and kindness.

We are also grateful to our teachers Prof. Rahul Tiwari, Prof. Sushma Gawande, Prof. Preeti Rangari, Prof. Prajakta Deshpande and Prof. Haresh Naringe for their encouragement, help and support from time to time.

We also wish to express our sincere thanks to Principal Dr. Swati S.Kathaley for providing us wide range of opportunities, facilities and inspiration to gather professional knowledge and material without which this project could not have been complete

Prathmesh P. Gulhane

Date:

Place: Nagpur

DECLARATION

We **Mr. Prathmesh P. Gulhane** hereby honestly declare that the work entitled **"Math Solver"** submitted by us at G. S. College of Commerce & Economics, Nagpur (Autonomous) in partial fulfillment of requirement for the award of B.Com. (Computer Application) degree by Rashtrasant Tukadoji Maharaj, Nagpur University, Nagpur has not been submitted elsewhere for the award of any degree, during the academic session 2022-2023.

The project has been developed and completed by us independently under the supervision of the subject teacher and project guide.

Prathmesh P. Gulhane

Date:

Place: Nagpur

Index

INTRODUCTION

1) INTRODUCTION:

Mathematics is a subject that plays an essential role in our everyday lives. From simple arithmetic to advanced calculus, it is used in a variety of fields, including science, engineering, finance, and technology. However, solving mathematical problems can be a challenging task, and it often requires a great deal of time and effort. To simplify the process of mathematical calculations, I have developed a Python GUI math solver that can solve a wide range of mathematical problems with ease.

Our Python GUI math solver is designed to assist students and professionals in performing mathematical calculations in a user-friendly and efficient manner. The program allows users to input mathematical problems through a graphical user interface and provides solutions for each problem. It includes a variety of features such as Pythagoras theorem, mean, mode, median, and other basic statistical calculations and mathematical calculations, making it a versatile tool for solving mathematical problems of different complexities.

The program is written in Python, a high-level programming language that is widely used in data science, artificial intelligence, and machine learning. Python is known for its ease of use and readability, making it an ideal choice for developing a math solver program that can be easily understood by users with varying levels of mathematical proficiency. The program utilizes various Python libraries such as Tkinter to perform complex calculations and generate graphical representations of the data.

The math solver program is designed to simplify the process of mathematical calculations for students and professionals alike. With the ability to solve a wide range of mathematical problems, the program can be used by individuals in different fields,

including education, research, and finance. By for each problem, the program not only makes mathematical calculations faster.

My Python GUI math solver is a powerful tool that can simplify the process of solving mathematical problems for students and professionals. With a user-friendly interface, the ability to solve a wide range of mathematical problems, and the use of Python libraries to perform complex calculations, the program is an excellent resource for those who wish to improve their mathematical skills and understanding. Whether you are a student, researcher, or professional in a mathematical field, our math solver program can assist you in performing your calculations with ease and accuracy.

Furthermore, our Python GUI math solver is not only a time-saving tool but also a highly accurate one. As the program follows a systematic approach in solving mathematical problems, it reduces the likelihood of human error and ensures accurate results. Additionally, the program can handle complex calculations with ease, which may otherwise require considerable time and effort when performed manually.

Overall, my Python GUI math solver is a versatile tool that can benefit a wide range of users. With its user-friendly interface, accurate results, and ability to handle complex calculations with ease, it is an essential resource for anyone who needs to perform mathematical calculations. By simplifying the process of mathematical calculations, our program not only saves time and effort but also helps users to improve their mathematical concepts.

OBJECTIVE

2) OBJECTIVES:

1. **Develop a user-friendly Python GUI.**

This objective involves creating a program that is easy to use and understand for users with varying levels of mathematical proficiency. The program should have a user-friendly interface that makes it easy for users to input mathematical problems and get accurate solutions.

2. **Provide a comprehensive solution for a wide range of mathematical problems.**

This objective involves creating a program that can solve a variety of mathematical problems, including basic arithmetic operations and more advanced statistical calculations. The program should be able to provide solutions for problems such as finding the mean, median, and mode of a set of data, calculating the Pythagorean theorem, and solving equations.

3. **Provide accurate and fast solutions to complex mathematical problems.**

This objective involves ensuring that the program is accurate and efficient in solving mathematical problems. The program should be able to provide solutions quickly and accurately, reducing the likelihood of human error and saving time and effort for the user.

4. **Generate graphical representations.**

This objective involves creating a program that can generate graphical representations of data, making it easier for users to understand the underlying concepts and visualize the results. The program should utilize Python libraries such as Tkinter to create attractive and interesting that can help users better understand the data.

5. Make the program accessible to users with varying levels of mathematical proficiency.

This objective involves creating a program that is accessible to users with varying levels of mathematical proficiency. By utilizing Python's ease of use and readability, the program should be able to provide solutions that are understandable for users with different levels of mathematical knowledge.

6. Create a versatile tool.

This objective involves creating a program that can be used in various fields such as education, research, finance, and technology. The program should be versatile enough to solve a variety of mathematical problems in different fields.

7.Provide a cost-effective solution for mathematical calculations.

This objective involves creating a program that is cost-effective for users. As the program is available for free and can be downloaded and used by anyone with access to a computer, it provides a cost-effective solution for mathematical calculations.

8.Contribute to the advancement.

This objective involves contributing to the advancement of the field of mathematics by creating a reliable and efficient tool for solving mathematical problems. The program can be used by professionals and researchers to solve complex mathematical problems, further advancing the field of mathematics.

PRELIMINARY SYSTEM ANALYSIS

3) Preliminary System Analysis:

Preliminary System analysis refers to the finding or arranging the various resources which will be needed at the time of developing a system, Information , and basic analysis of system. It is first and basic step for developing every system . It is the phase where system is investigated. The objective of this phase is to conduct an initial analysis and findings of the system.

3.1) Preliminary Investigation:

The project aims to develop a Python GUI math solver that simplifies the Process of solving mathematical problems for students and professionals.The preliminary investigation involved identifying the need for a tool that can perform mathematical calculations efficiently and effectively, without requiring manual efforts.

3.2) Present System in Use:

Currently, individuals rely on manual calculations or the use of calculators to perform mathematical calculations. While calculators are efficient, they do not always provide all the solutions at a single place and also less attractive interface, making it challenging for individuals to use.

3.3) Flaws in present System:

The flaws in the present system include the lack of accuracy and attractive Interface solutions , which hinder the understanding of mathematical concepts and makes it boring. Additionally , manual calculations are time-consuming and prone to errors, resulting in inaccurate results.

3.4) Need of New System:

The need for a new system arise from the flaw in the present system. A new system that can provide all the basic math solutions at one place and perform calculations efficiently and accurately would be an ideal solution for students and professionals alike.

3.5) Feasibility Study:

A feasibility study was conducted to assess the viability of the project. The study evaluated the technical, operational, economic, and legal feasibility of developing a Python GUI math solver. It was determined that the project is technically feasible, as the necessary skills and resources are available to develop the program. Additionally, the program is operationally feasible, as it can be easily integrated into the user's workflow. From an economic standpoint, the program is cost-effective and offers significant benefits to its users. Finally, the program is legally feasible, as there are no legal barriers to developing and distributing the program.

- **Technical Feasibility –**

Technical feasibility, we considered various technical aspects considering the software and hardware requirements and the other technical requirements will be needed to develop a website and analysed that they will be feasible to build to website or not. Searched for the best Integrated Development Environment(IDE) and ensured that the software will be able to work with multiple Programming languages without giving any system error, also whether the various software we are installing for the development are compatible with the hardware of device and the versions are supported by them or not is fully investigated.

- **Operational Feasibility:**

To determine operational Feasibility of the system we analysed that the software is reliable to handle every type of human error. Also, checked whether it is supporting various file extensions and images extensions or not. What will be the sustainability and productivity after the execution and implementation of the website is analysed, as well as the information, maps and images we are providing through our website is updated or not is also ensured. Apart form these, the adoptability of the system whether after the implementation how efficiently it allowing owner to make changes or updates in the system? And how much the system will remain dynamic with different size of the devices? Studied.

- **Economic Feasibility:**

The total estimated cost of the project has been projected. What will be the cost needed to develop a project right form collecting the resources to the implementation has been discussed. The financial aspects of the projects and the current budget we have in our hand to develop the project is suitable for the execution or not is studied. After analysing and proper distribution of the budget between various sections the Project is completely financially feasible. The cost which will be needed to buy all the requirements of the project is already finalised properly.

- **Legal Feasibility:**

In legal feasibility, ensured that the website we are developing is legal in the laws and permitted by the law. Any information which will be provided will not hurt the cultural and religious sentiments of the people and the information we are providing on the site from the sources are valid or not is ensured.

- **Time Feasibility:**

In time feasibility study, taken into account the period in which the project is going to take up to its completion. We made sure that the project will not take too long for completion.

3.6) Project Category:

The project falls under the category of educational software, as it aims to assist students and professionals in performing mathematical calculations more efficiently and accurately. This project logic is developed in python using the tkinter module.

Language Used:

Python:

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

Database:

The Database is used to take the data from the user and save it on the system, which only will be visible to the system admin. The feedback form is created to take user data from the user and share their personal views about the website and the difficulties they faced. So, that we can improve the website.

Many databases available like MySQL, Sybase, Oracle, MongoDB, PostgreSQL, SQL Server, etc. In this section, we are going to focus on MySQL mainly.

MySQL:

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched. The process of MySQL environment is the same as the client-server model.

MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc. MySQL works very quickly.

Software and Hardware Requirements

Hardware and Software Requirements:

Software and Hardware Requirements Specifications:

Every application needs the software in which it has to be executed and a hardware theapplication is going to perform its function. Some application cannot run on every platform and some applications needs some specific requirement in the software or in hardware to get operated. Let's take an example of the applications which cannot be runon every platform like windows, android, Linux, etc. Applications made in visual basic is only supported for the windows, one cannot access these applications from the mobile phones, etc. So, here are some hardware and software specifications which are mandatory for the application to get operated.

Software Requirements:

Operating System: The software can run on any platform, including Windows, Linux, and macOS.
Python: The software requires Python 3.x installed on the system.
Python libraries: The following Python libraries need to be installed for the software to function correctly:

* tkinter: To create the graphical user interface.
PIL import ImageTk,Image: To import the images.
Pymysql: To connect the mysql database.
Statistics: For the statistical predefined functions.
Math: To use the predefined math functions.

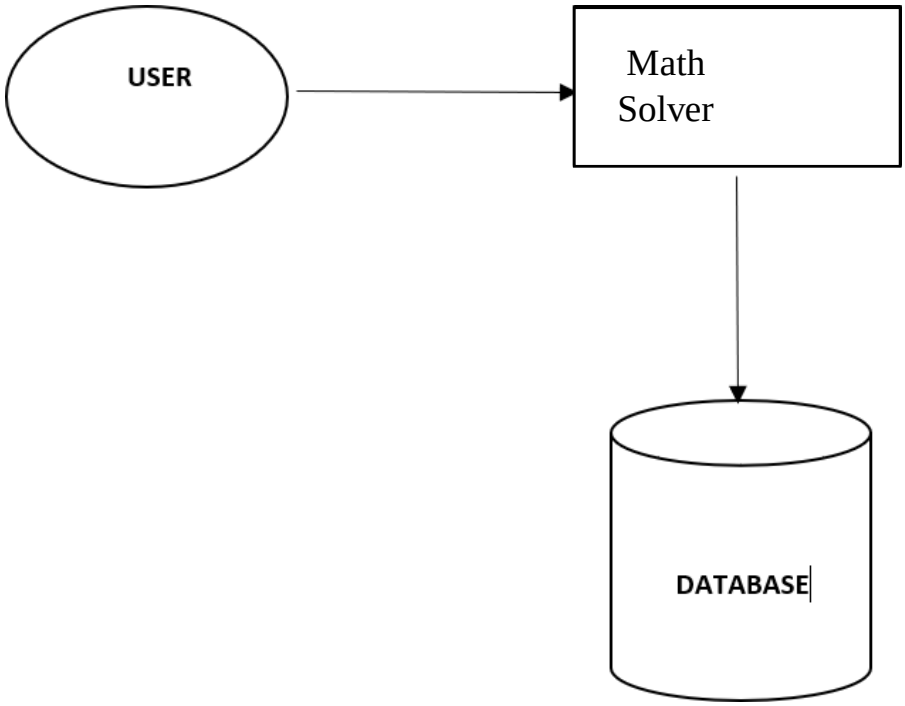
Hardware Requirements:

Processor: The software can run on any processor with a speed of 1 GHz or higher.
RAM: The minimum required RAM for the software is 2 GB, although 4 GB or more is recommended for optimal performance.
Storage: The software requires approximately 100 MB of free storage space on the hard drive.

Detailed System Analysis

5) Detailed System Analysis

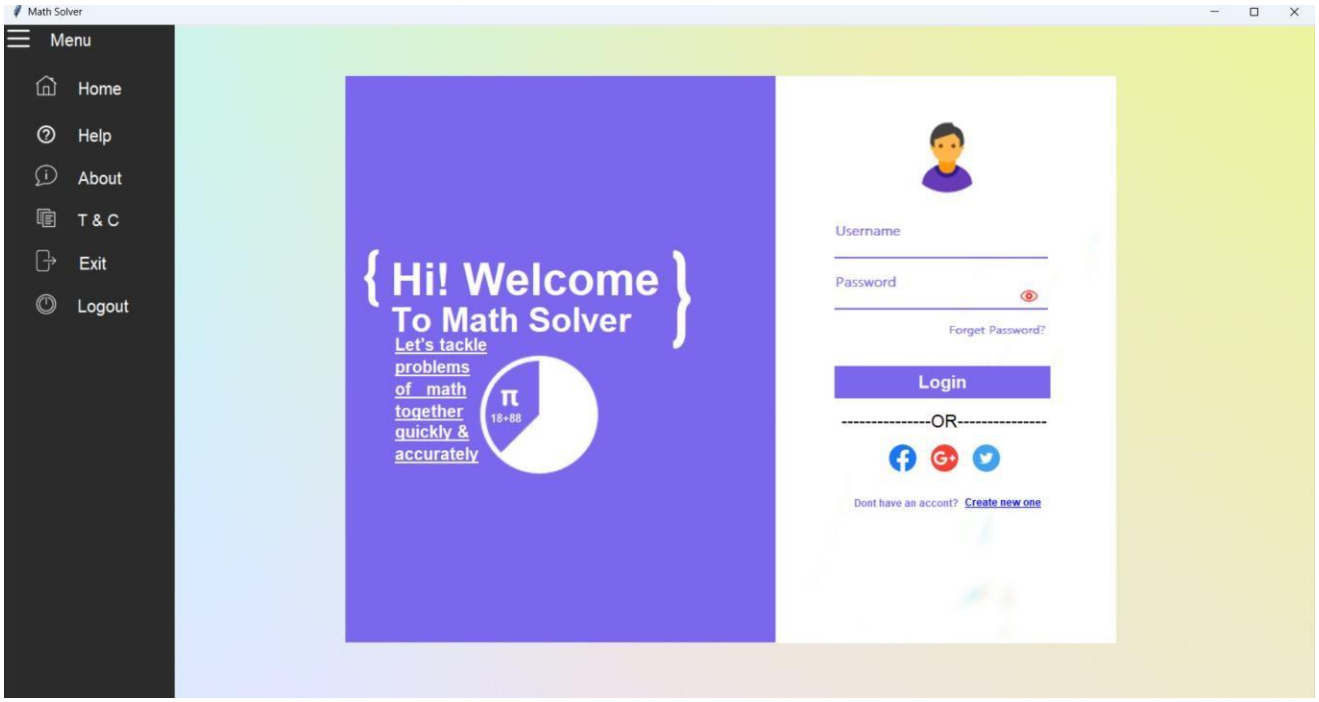
- **Data Flow Diagram**



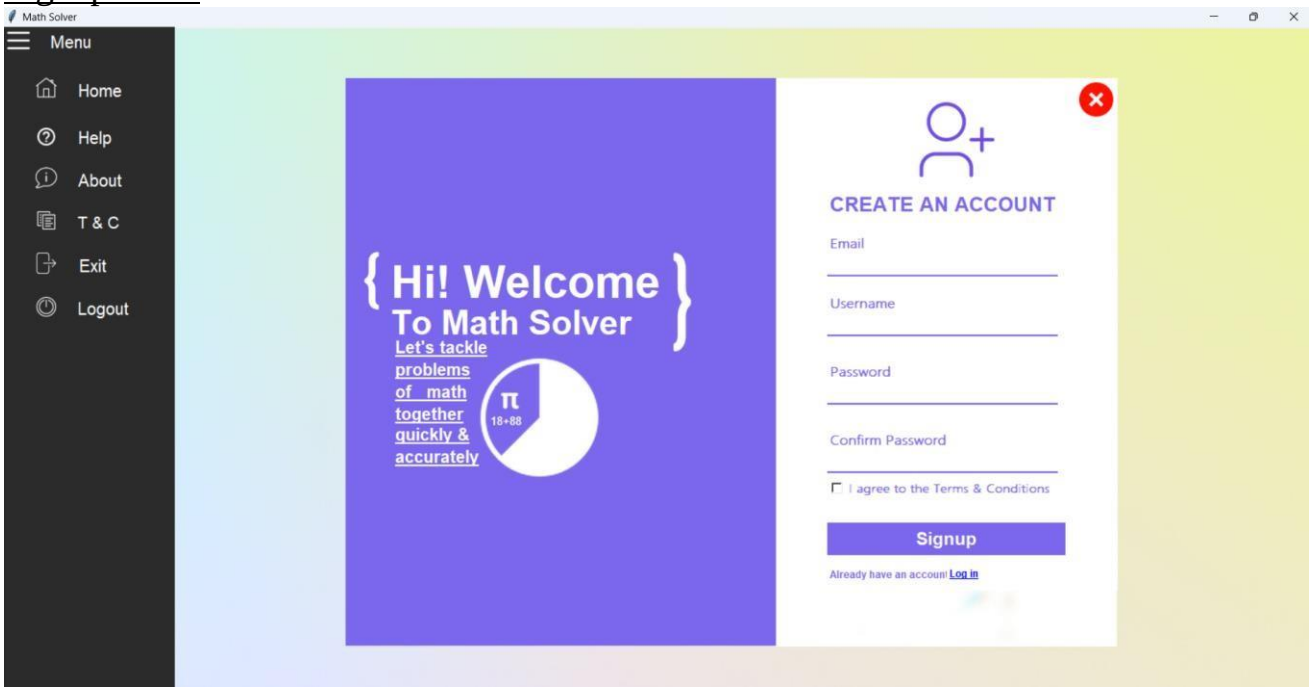
System Design

6.1) Form Design:

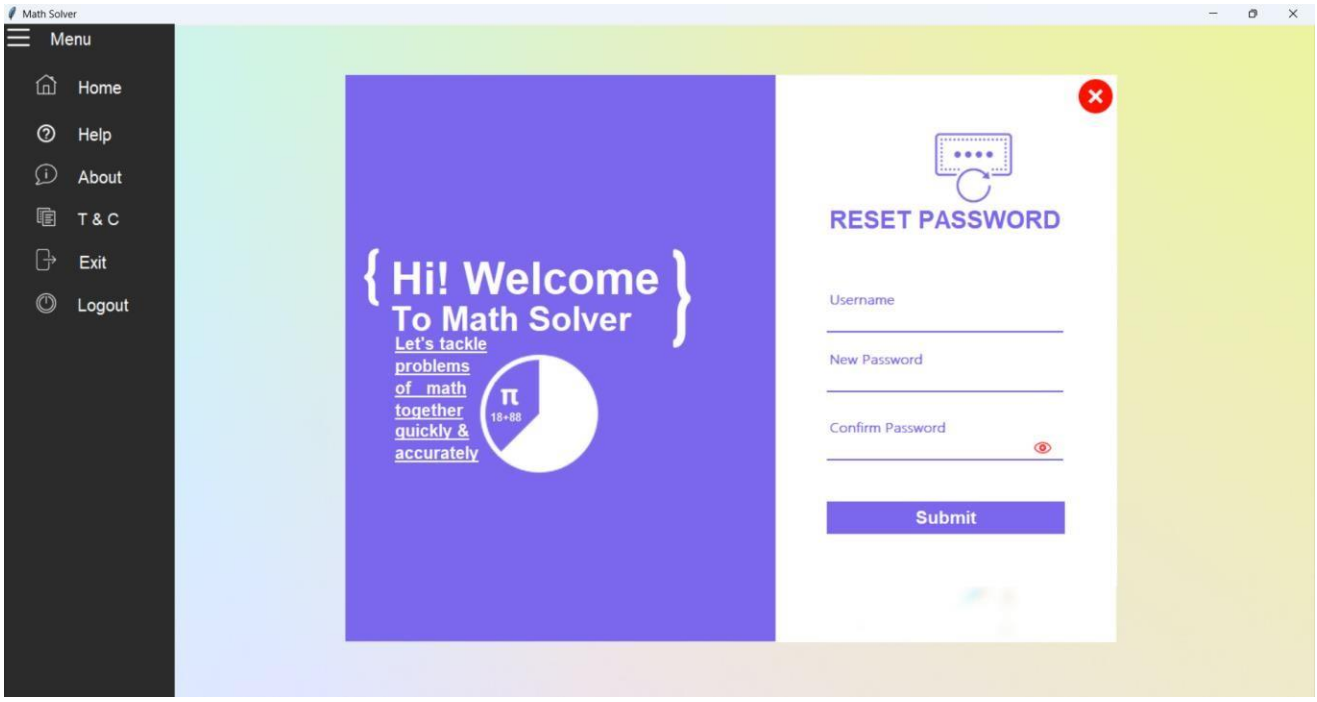
Login Form



Signup Form

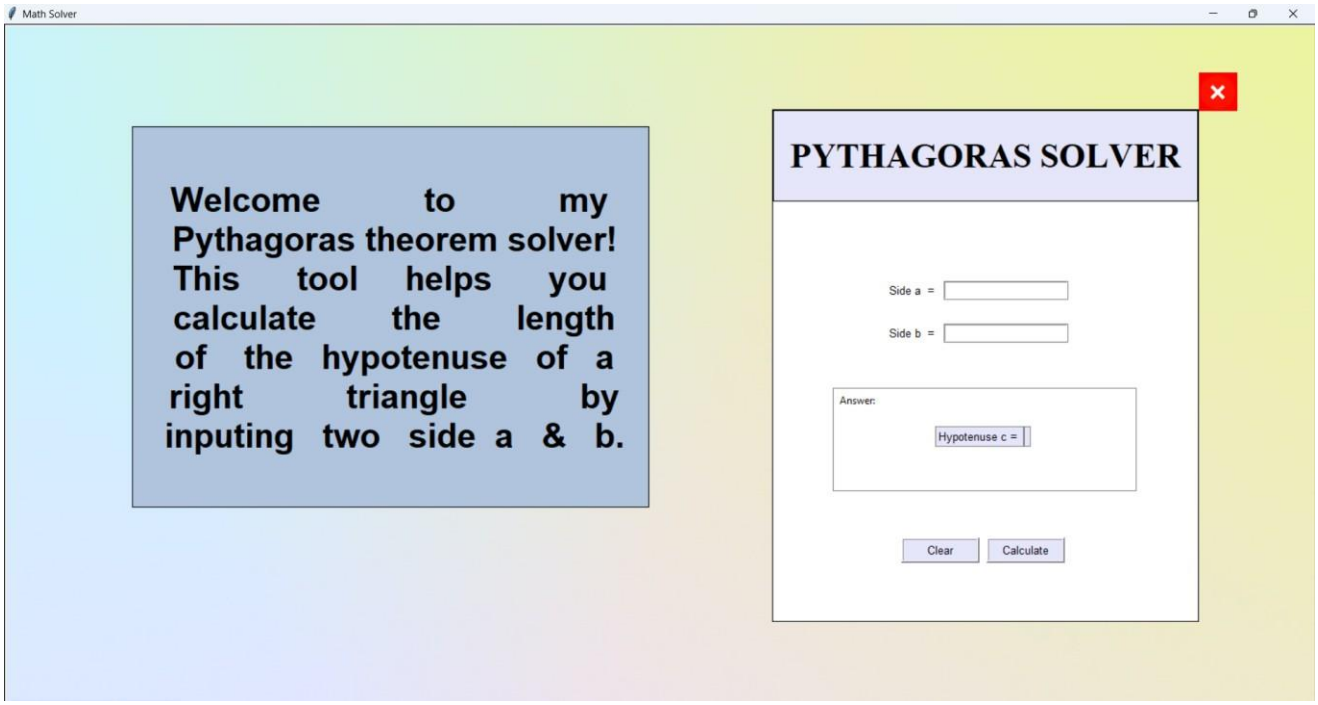


Forget Password Form

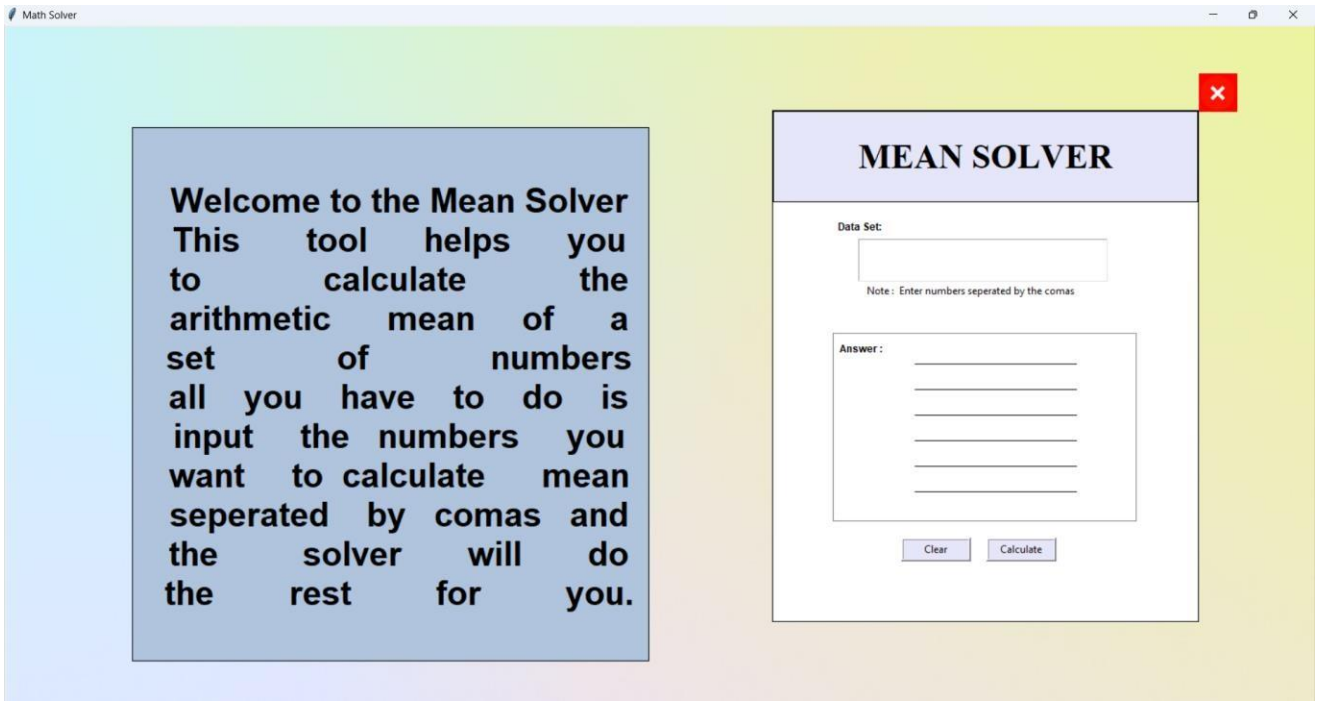


Selection of Solver Form

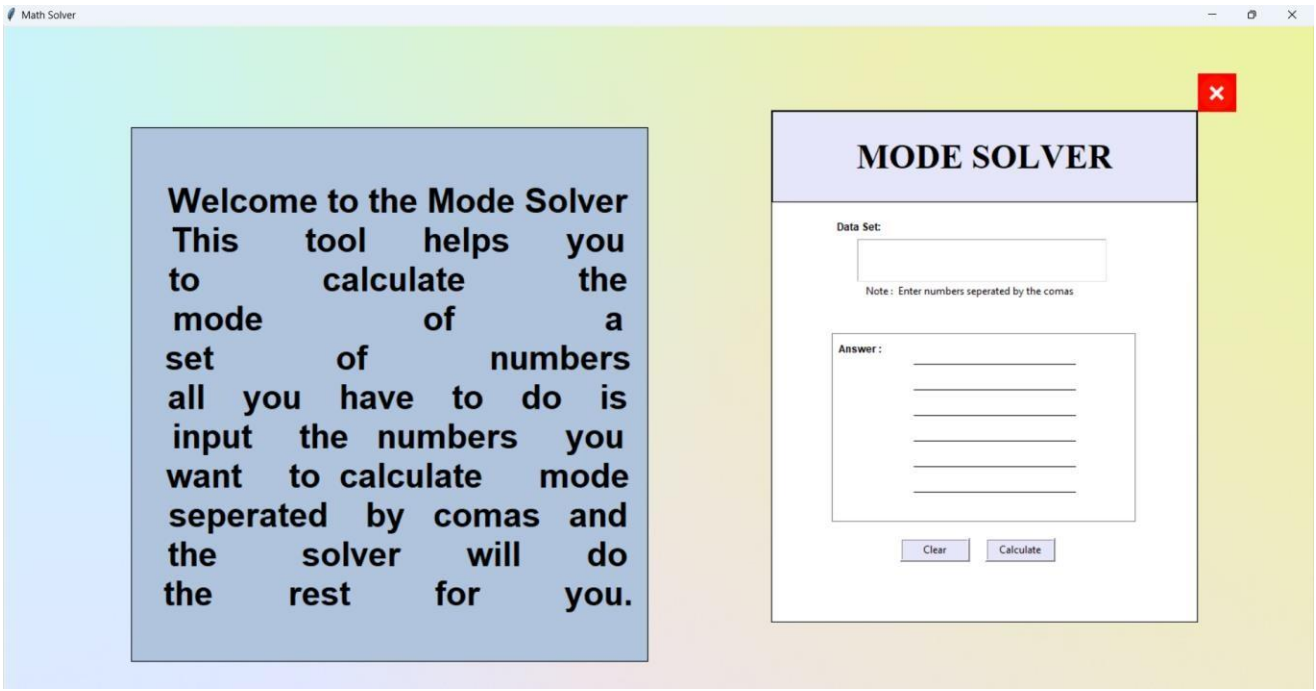
Pythagoras Solver Page



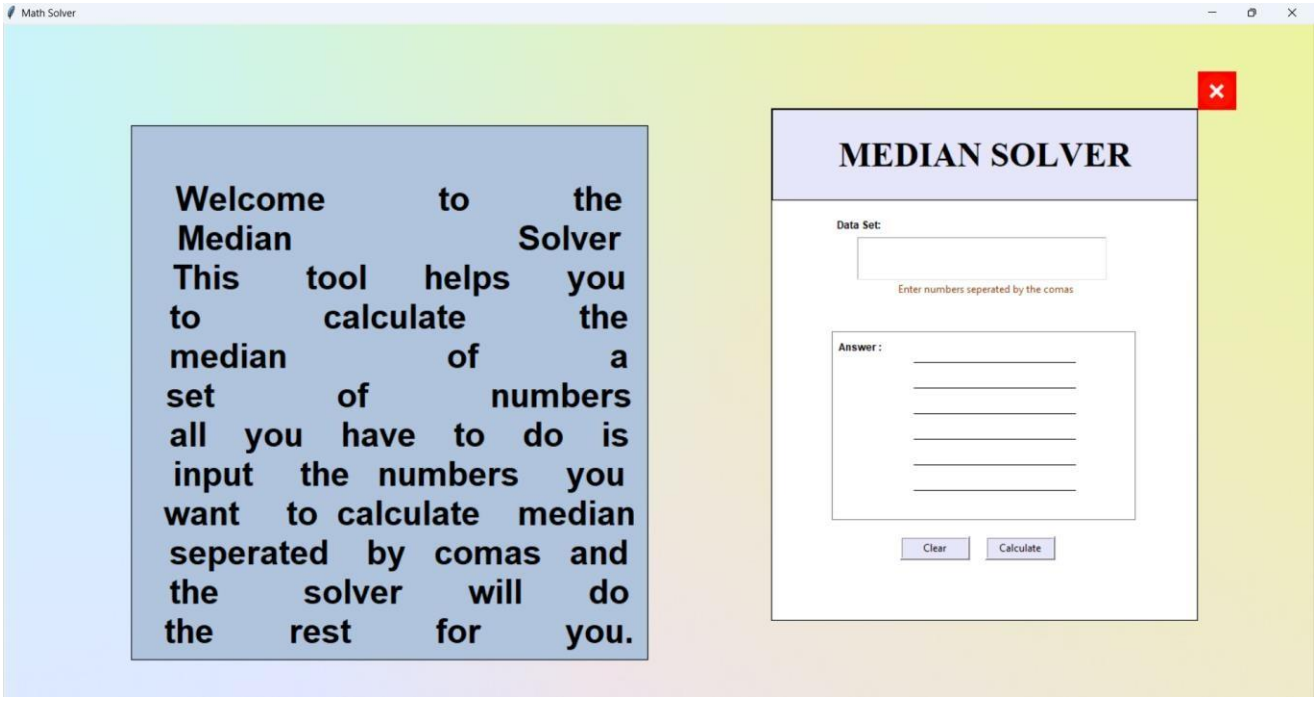
Mean Solver Page



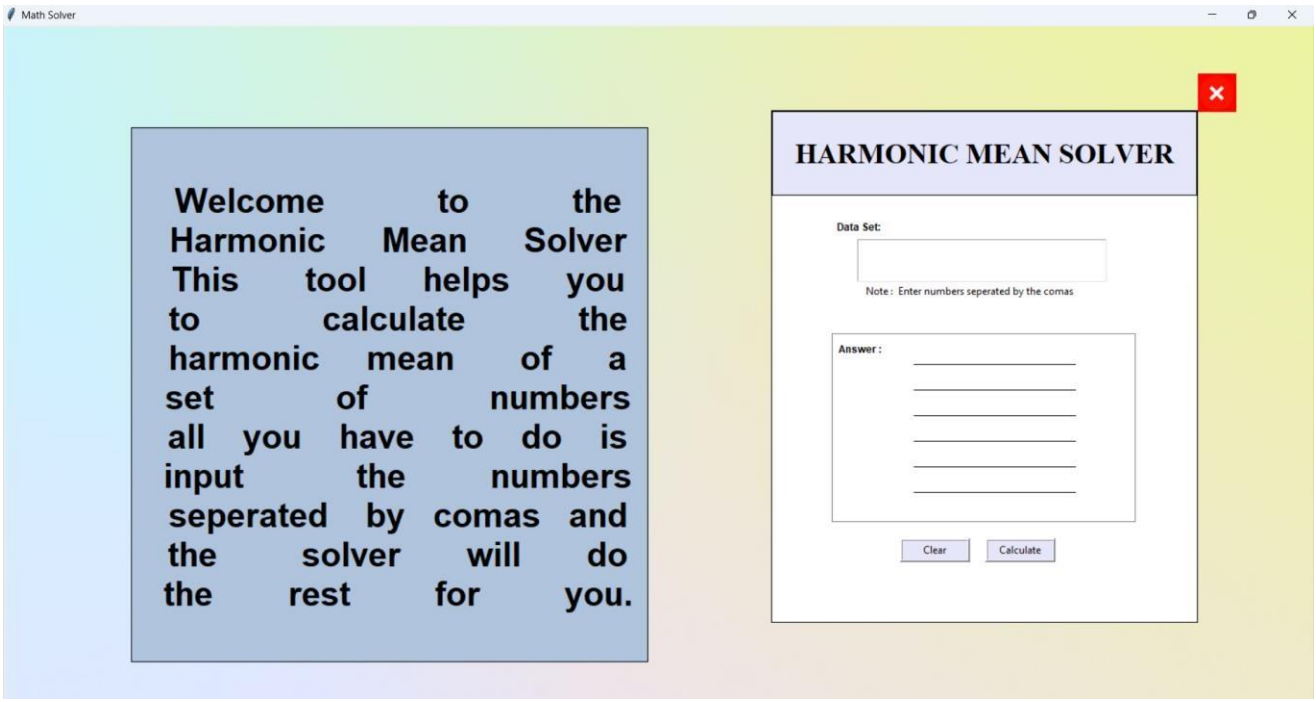
Mode Solver Page



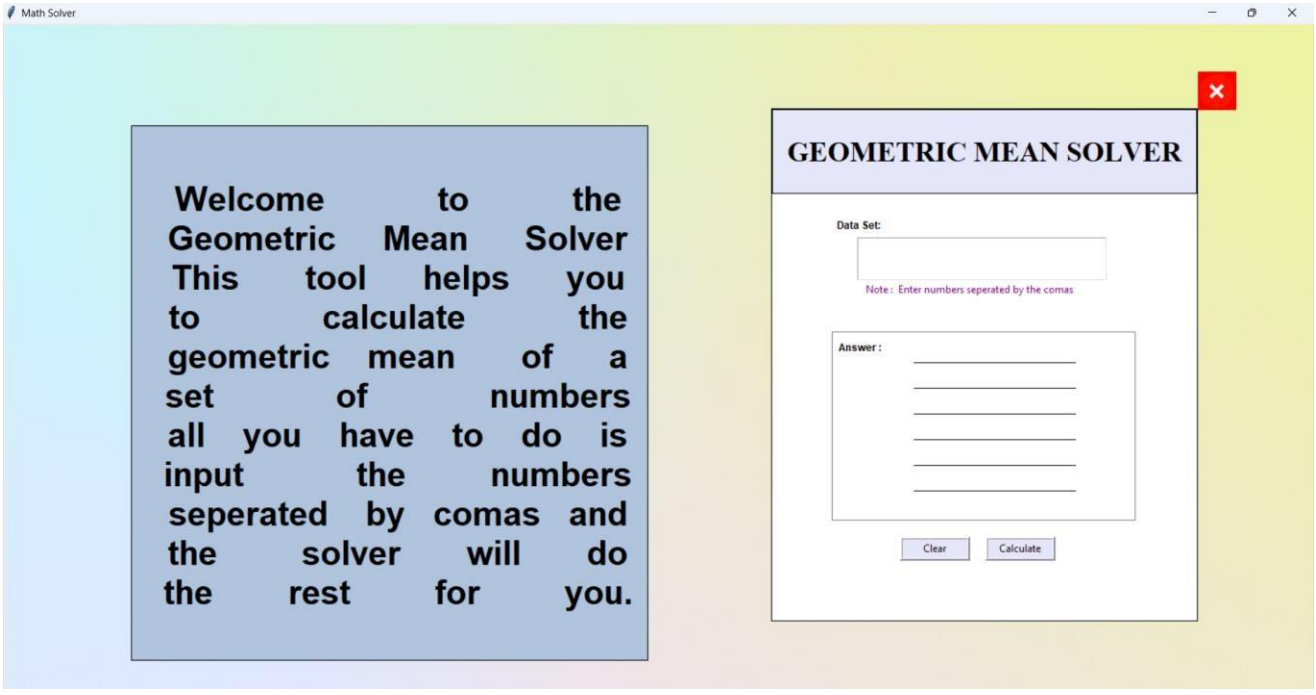
Median Solver Page



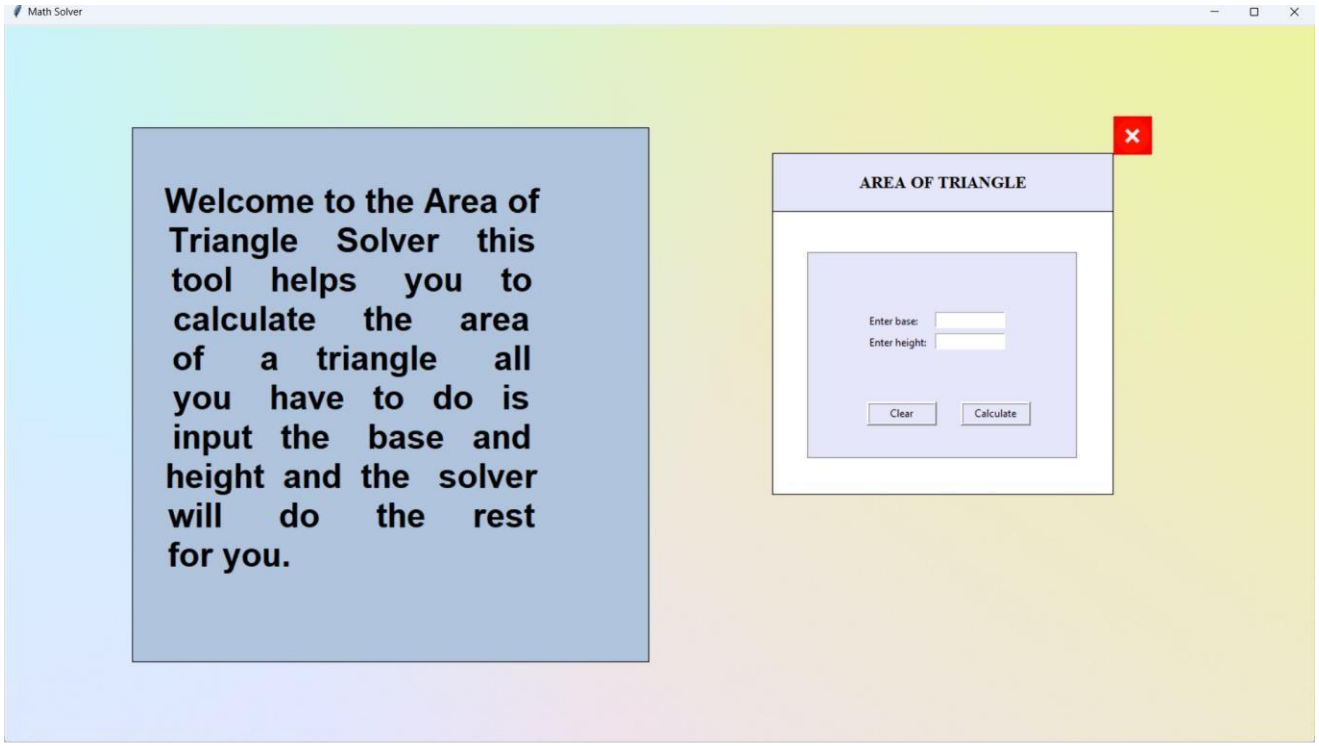
Harmonic Solver Page



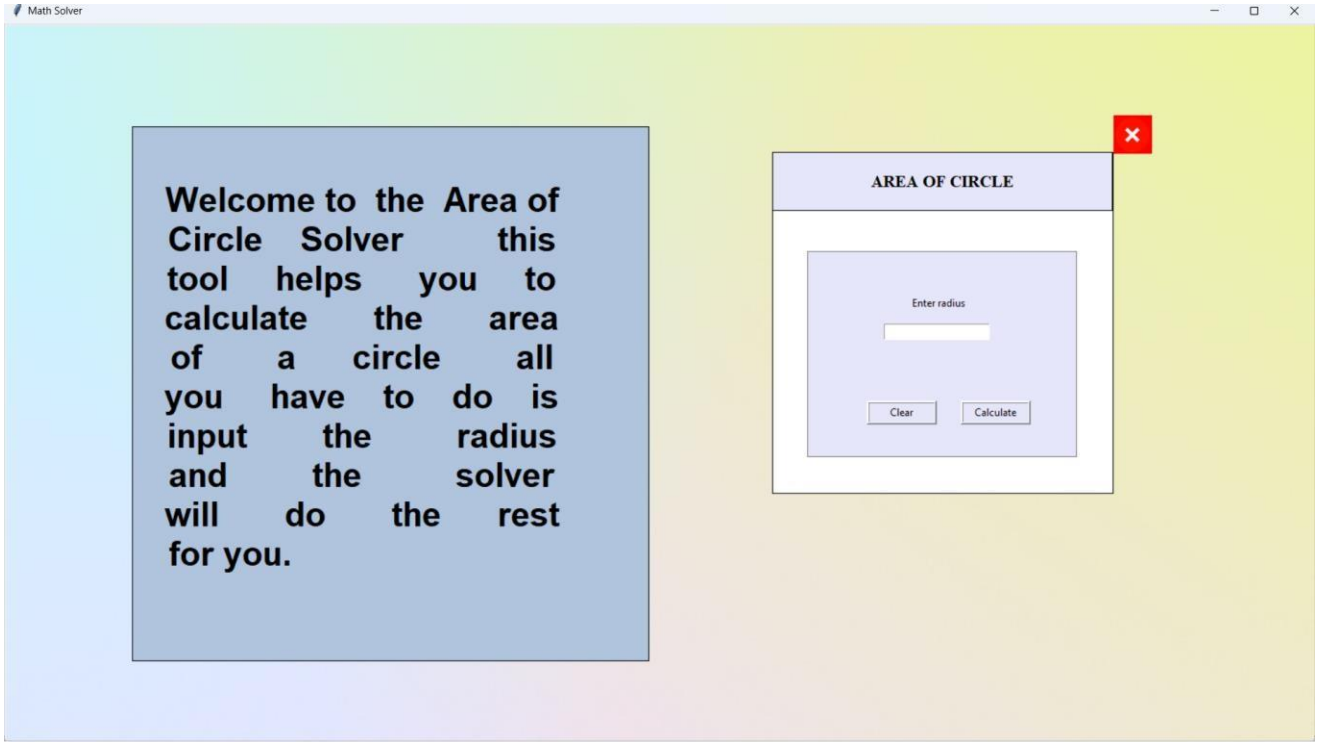
Geometric mean Solver Page



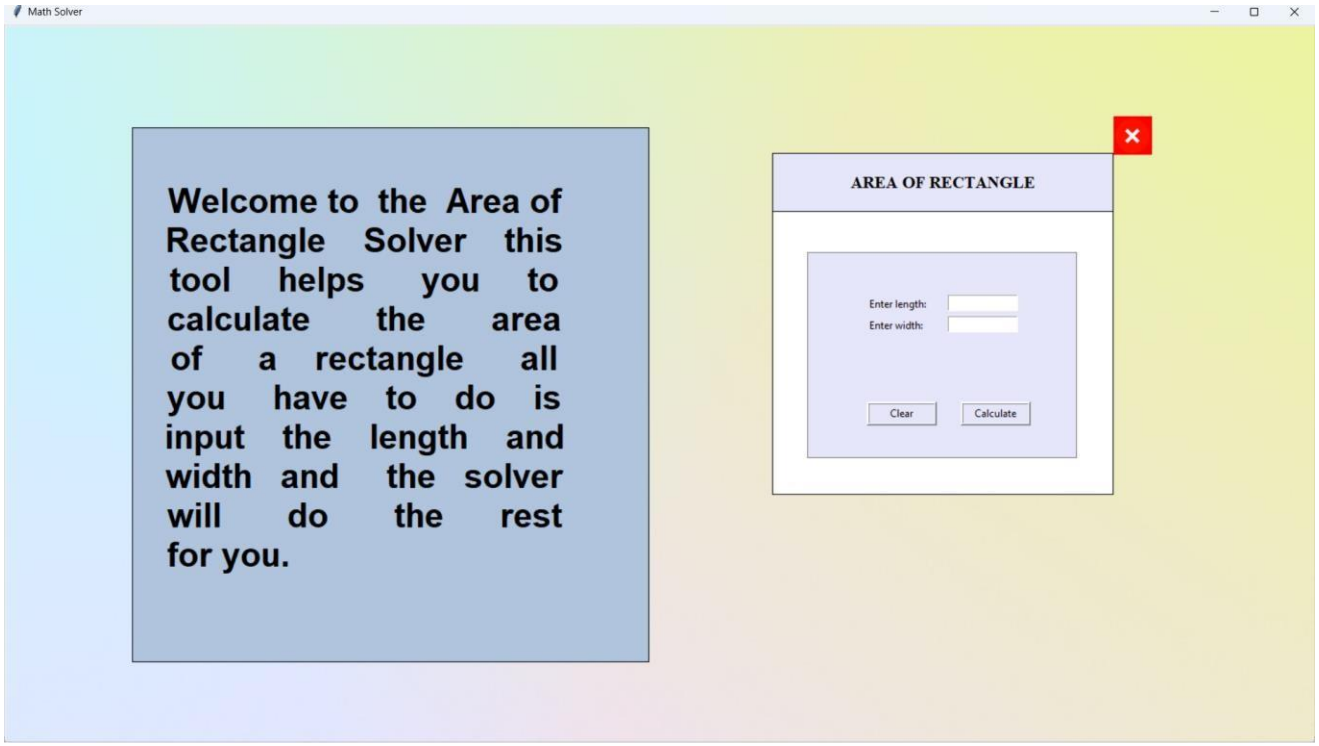
Area of Triangle Solver Page



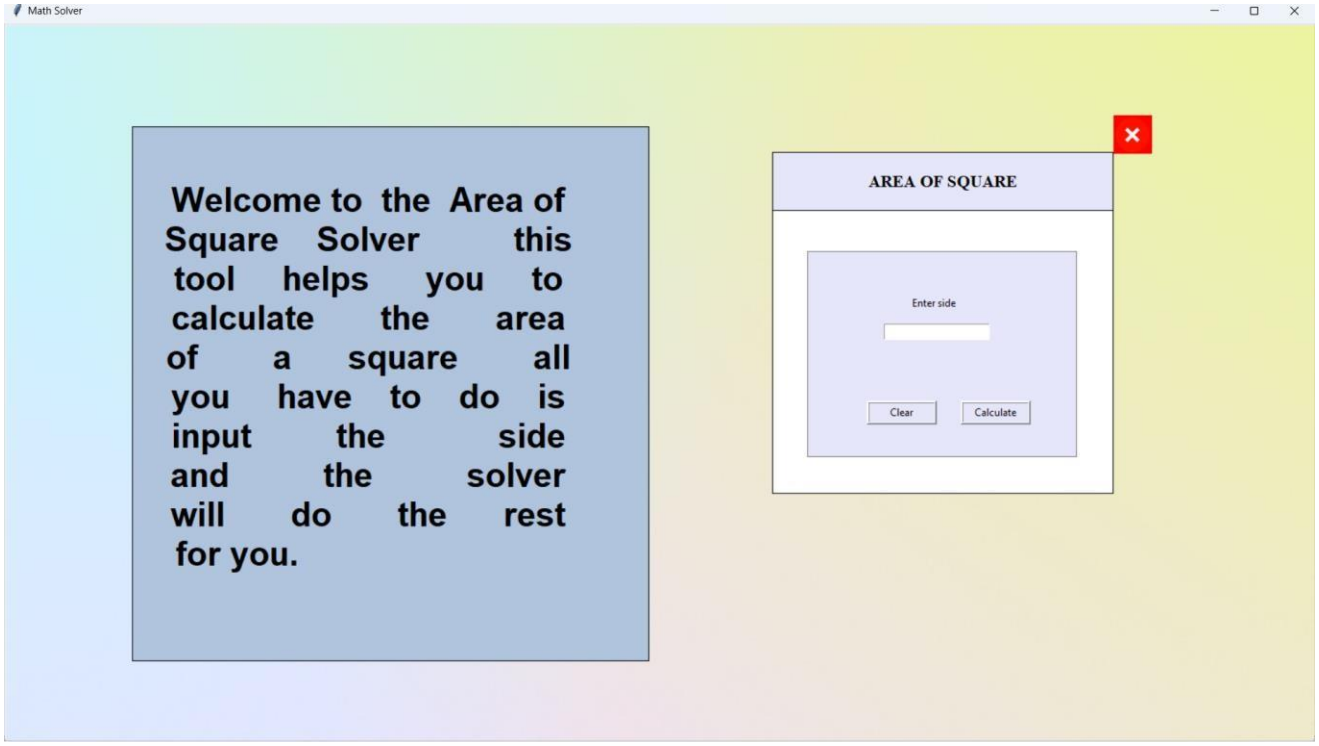
Area of Circle Solver Page



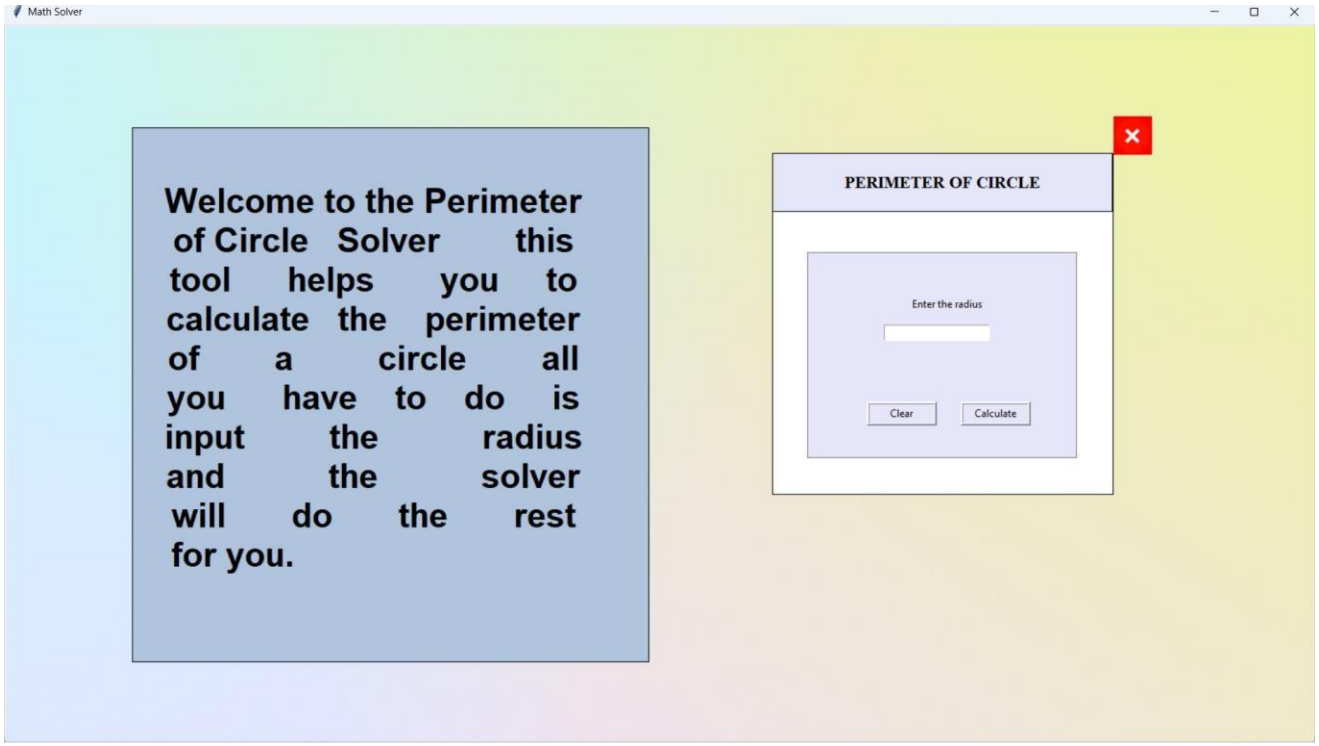
Area of Rectangle Solver Page



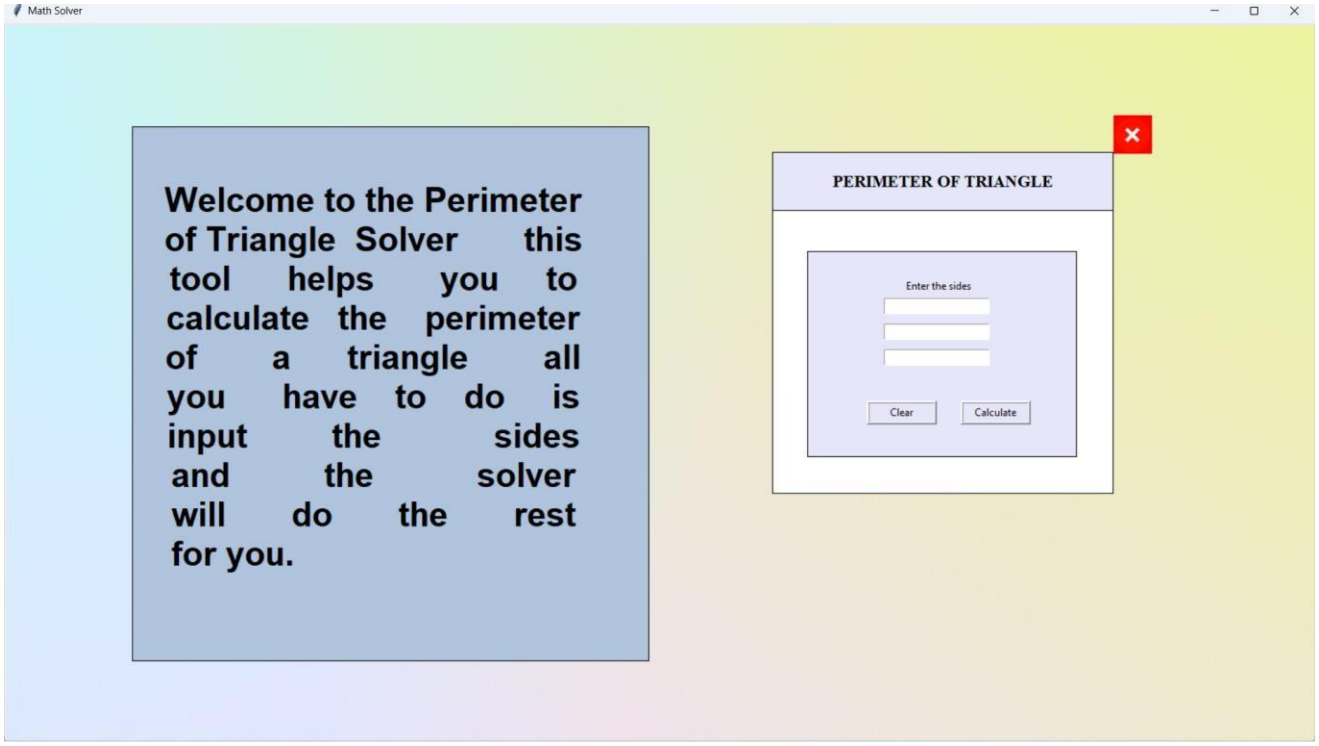
Area of Square Solver Page



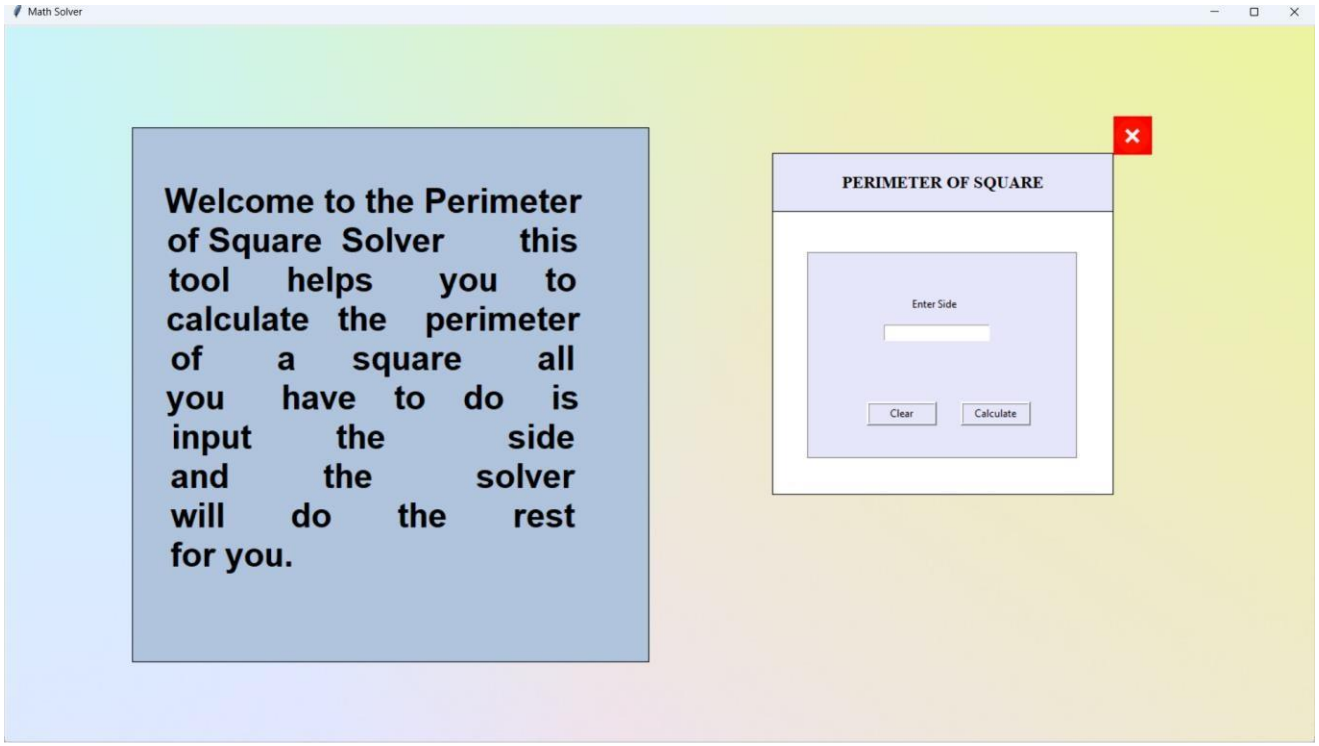
Perimeter of Circle Page



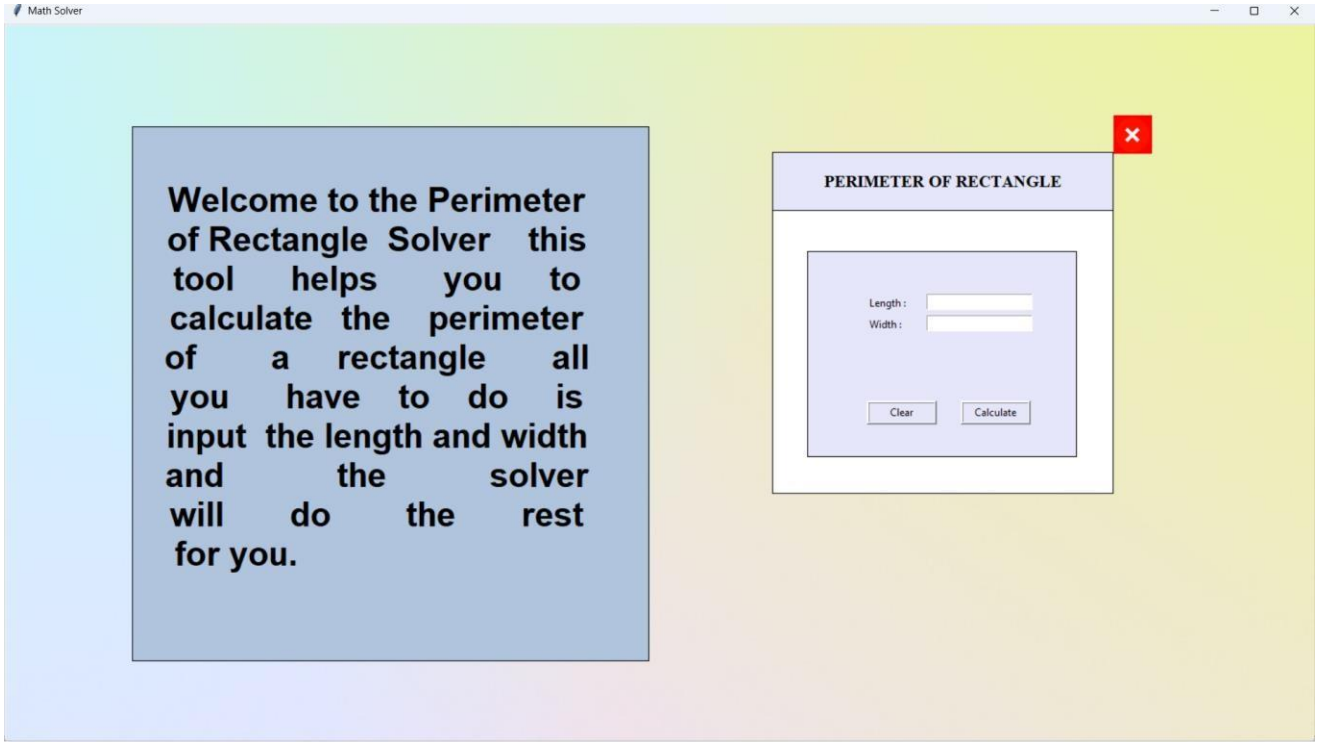
Perimeter of Triangle Page



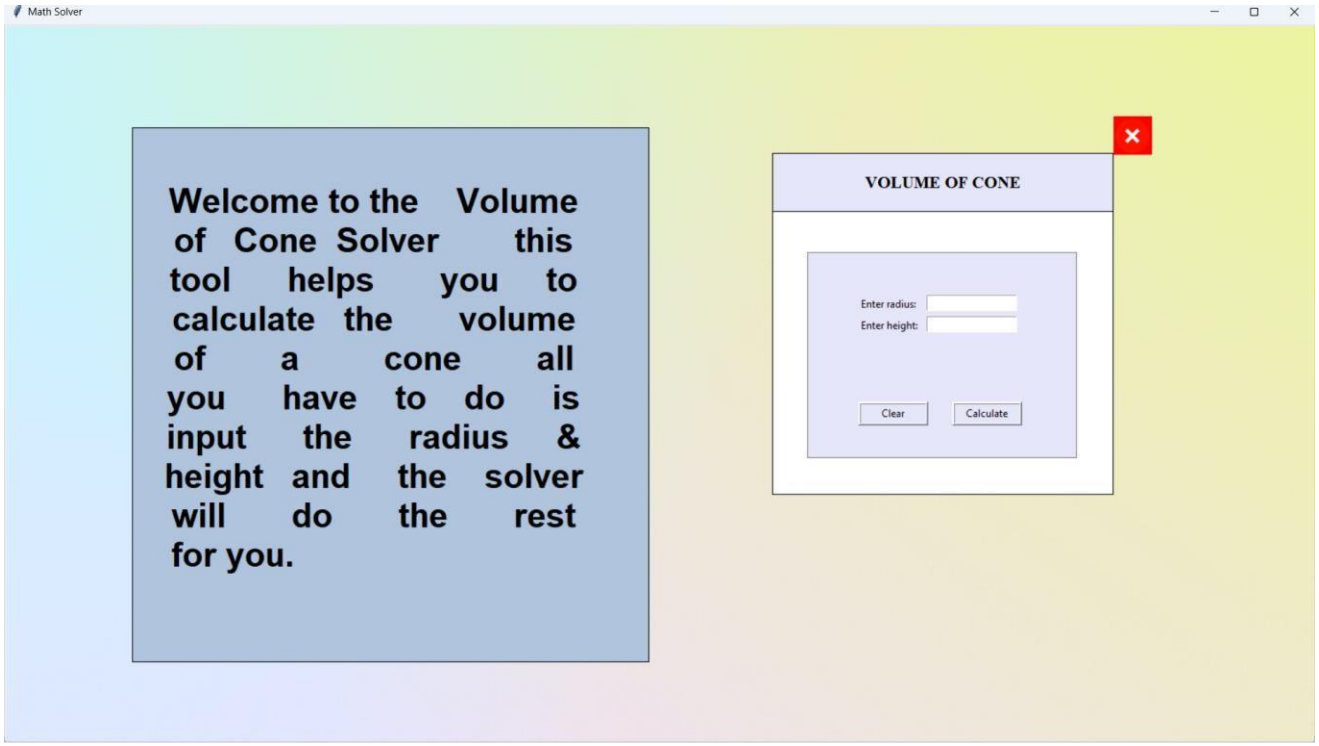
Perimeter of Square Page



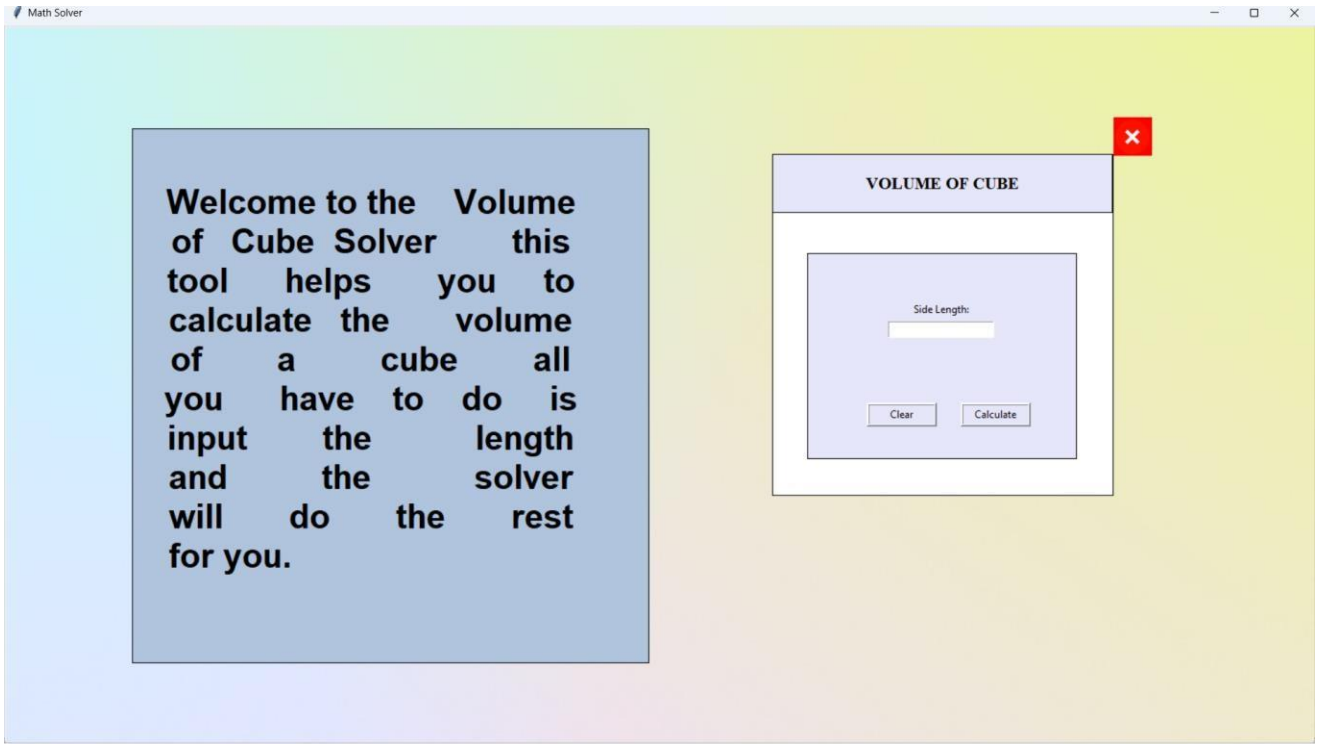
Perimeter of Rectangle Page



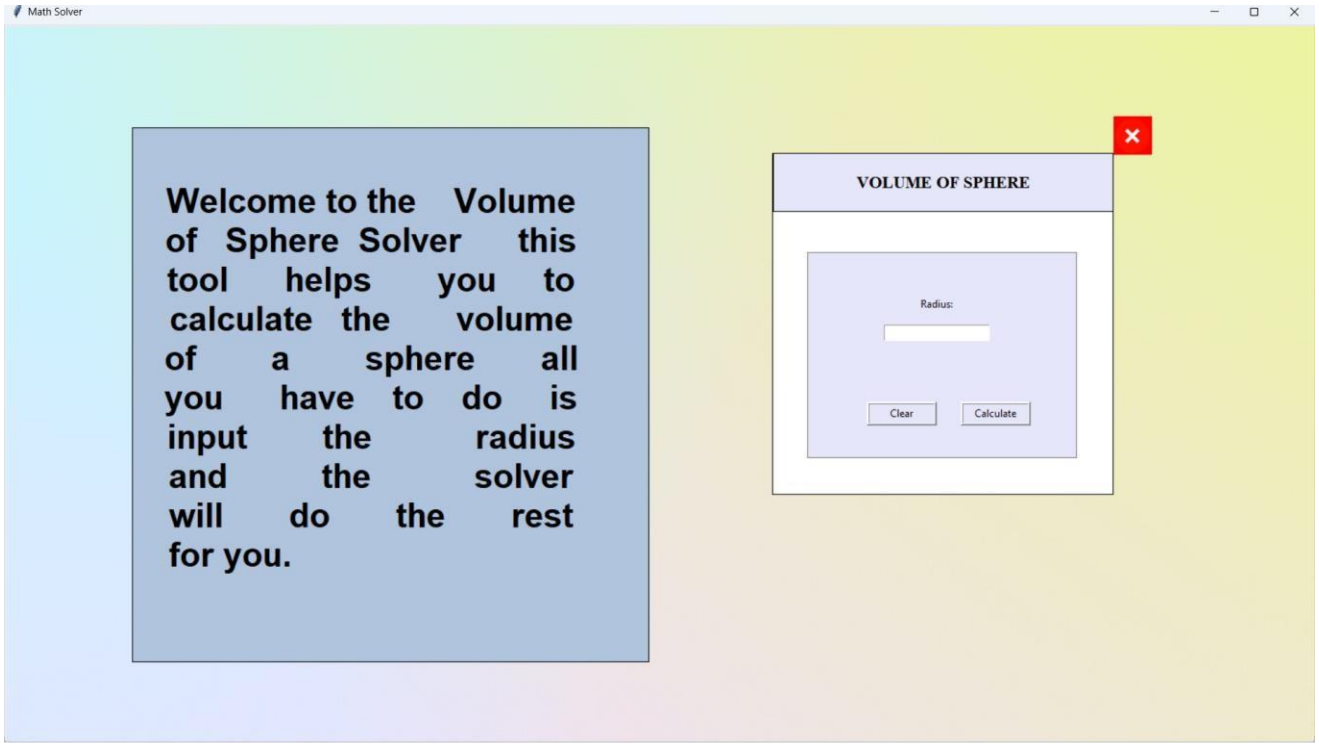
Volume of Cone Page



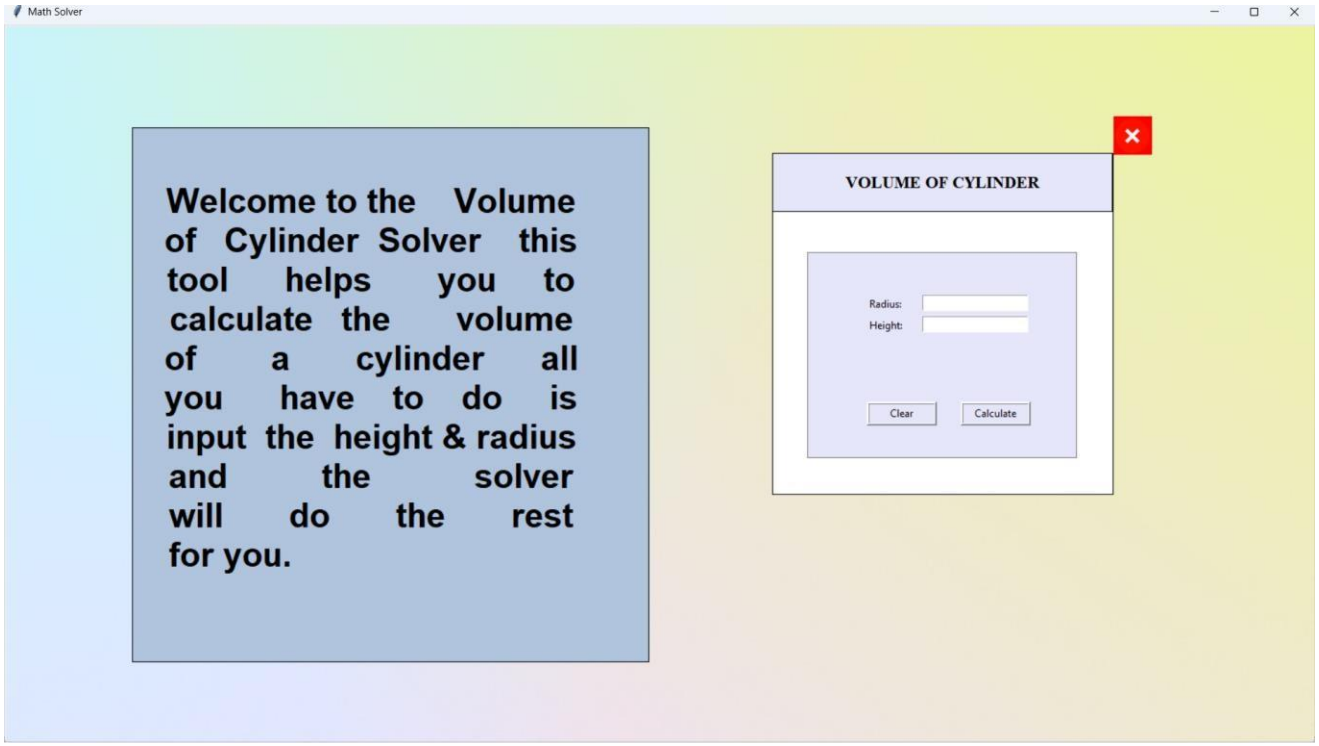
Volume of Cube Page



Volume of Sphere Page



Volume of Cylinder Page



6.2) Source Code

```
from tkinter import *
from PIL import ImageTk,Image
from tkinter import messagebox
import pymysql
import statistics
import math

# functions
def pythagoras_page():
    bgLabel1 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
                     highlightbackground="black",highlightthickness=1)
    bgLabel1.place(x=0, y=0)
    frame6 = Frame(login_window, width=500, height=600,
bg="white",highlightbackground="black",highlightthickness=1)
    frame6.place(x=900, y=100)
    def closethepythagoras():
        bgLabel1.destroy()
        frame29.destroy()
        pythagoras_intro.destroy()
        lblinpythagoras.destroy()
        frame6.destroy()
    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",highlightthickness=1, bd=0)
    frame29.place(x=1400, y=57)
    btn10 = Button(frame29, image=closeiicon, command=closethepythagoras,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

    def calculate_pythagoras():
        side_1_length = float(side_1.get())
        side_2_length = float(side_2.get())
        result.set((side_1_length ** 2 + side_2_length ** 2) ** (0.5))

    def clear_pythagoras():
        side_1.set('')
```

```
        side_2.set('')
        result.set('')
    result = StringVar()
    side_1 = StringVar(value='')
    side_2 = StringVar(value='')
    lblinpythagoras = Label(login_window, text="", bg="light steel blue", font=("Calibri
", 40),padx=300, pady=190, highlightcolor="black"
,highlightbackground="black",highlightthickness=1)
    lblinpythagoras.place(x=150, y=120)

    pythagoras_intro = Label(login_window, text="Welcome to my
\nPythagoras theorem solver!\n"
                             "This tool helps you \ncalculate the
length\nof the hypotenuse of a\n"
                             "right triangle by\ninputing two side a &
b.", bg="light steel blue", font=("Calibri ", 30,"bold"),fg = "black")
    pythagoras_intro.place(x=185, y=180)

    lbl2 = Label(frame6, text="PYTHAGORAS SOLVER", bg="lavender", fg="black",
borderwidth=1, relief="solid",font=('Times', 30, "bold"), padx=19, pady=30)
    lbl2.place(x=0, y=0)

    lbl3 = Label(frame6, text="", highlightbackground="grey", highlightthickness=1,
bg="white", padx=175, pady=50)
    lbl3.place(x=70, y=325)

    lbl4 = Label(frame6, text="Answer:", bg="white")
    lbl4.place(x=75, y=329)

    lbl5 = Label(frame6, text="Side a  = ", font=('Arial Rounded MT', 10), relief="ridge",
borderwidth=0, bg="white")
    lbl5.place(x=135, y=202)

    lbl6 = Label(frame6, text="Side b  = ", font=('Arial Rounded MT', 10), relief="ridge",
borderwidth=0, bg="white")
    lbl6.place(x=135, y=252)

    lbl7 = Label(frame6, text="Hypotenuse c = ", font=('Arial Rounded MT', 10),
bg="lavender",highlightbackground="grey", highlightthickness=1, pady=1)
    lbl7.place(x=190, y=370)
```

```
    lbl8 = Label(frame6, textvariable=result, bg="lavender", font=('Arial Rounded MT',
10), highlightbackground="grey",highlightthickness=1)
    lbl8.place(x=294, y=370)

    txt1 = Entry(frame6, textvariable=side_1, font=('Arial Rounded MT', 10),
highlightbackground="grey",highlightthickness=1)

    txt1.place(x=200, y=200)

    txt2 = Entry(frame6, textvariable=side_2, highlightbackground="grey",
highlightthickness=1, font=('Arial Rounded MT', 10))
    txt2.place(x=200, y=250)

    btn1 = Button(frame6, text="Calculate",command=calculate_pythagoras,relief="ridge",
bd=3, font=('Arial Rounded MT', 10), bg="lavender",width=10,cursor="hand2")
    btn1.place(x=250, y=500)

    btn2 = Button(frame6, text="Clear", command=clear_pythagoras, relief="ridge", bd=3,
font=('Arial Rounded MT', 10), bg="lavender",width=10,cursor="hand2")
    btn2.place(x=150, y=500)

def mean_page():
    bgLabel2 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel2.place(x=0, y=0)
    global  frame4
frame4=Frame(login_window,width=500,height=600,bg="white",highlightbackground="bl
ack",highlightthickness=1)
    frame4.place(x=900, y=100)

    lblinmean= Label(login_window, text="", bg="light steel blue",
                font=("Calibri ", 40), padx=300, pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblinmean.place(x=150, y=120)
    mean_intro = Label(login_window,

                text="Welcome to the Mean Solver\nThis tool helps you\n"
                    "to calculate the\narithmetic mean of a\n"
                    "set of numbers\nall you have to do is\n"
```

```
            "input the numbers you\nwant to calculate mean\nseperated "
            " by comas and\nthe solver will do\nthe rest"
            "for you.",
        bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    mean_intro.place(x=185, y=180)
    def closethemean():
        bgLabel2.destroy()
        frame29.destroy()
        mean_intro.destroy()

        lblinmean.destroy()
        frame4.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",highlightthickness=1, bd=0)
    frame29.place(x=1400, y=57)
    btn10 = Button(frame29, image=closeiicon, command=closethemean, cursor="hand2",
bg="red", bd=0)
    btn10.place(x=-4, y=-3)
    def calculate_mean():
        # Get the numbers_median from the entry field and convert them to a list of floats
        numbers_mean = list(map(float, mean_number_entry.get().split(",")))

        # Calculate the statistics and display them in the labels
        mean_count = len(numbers_mean)
        mean_minimum = min(numbers_mean)
        mean_maximum = max(numbers_mean)
        mean_total = sum(numbers_mean)
        mean = statistics.mean(numbers_mean)
        mean_count_label.config(text=f"Count {mean_count}")
        mean_minimum_label.config(text=f"Minimum {mean_minimum:.2f}")
        mean_maximum_label.config(text=f"Maximum {mean_maximum:.2f}")
        mean_total_label.config(text=f"Sum {mean_total:.2f}")
        mean_label.config(text=f"Mean {mean:.2f}")

    def clear_input_mean():
        # Clear the entry field and the result labels
        mean_number_entry.delete(0, END)
        mean_count_label.config(text="")
        mean_minimum_label.config(text="")
```

```
        mean_maximum_label.config(text="")
        mean_total_label.config(text="")
        mean_label.config(text="")

    # note
    lbl_note2 = Label(frame4, text="Note :", highlightbackground="grey",
highlightthickness=0, fg="black",bg="white").place(x=107, y=200)
    lbl_note2 = Label(frame4, text="Enter numbers seperated by the comas", fg="black",
highlightbackground="grey",
                highlightthickness=0, bg="white").place(x=145, y=200)


# design
    lbl2 = Label(frame4, text="MEAN SOLVER", bg="lavender", fg="black",
borderwidth=1, relief="solid",font=('Times', 30, "bold"), padx=99, pady=30).place(x=0,
y=0)
    lbl3 = Label(frame4, text="", highlightbackground="grey", highlightthickness=1,
bg="white", padx=175, pady=100).place(x=70, y=260)
    lbl4 = Label(frame4, text="Answer :", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=75, y=268)
    lbl4 = Label(frame4, text="Data Set:", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=73, y=125)

    # Create labels and entry widgets
    mean_number_entry = Entry(frame4, bg="white")
    mean_number_entry.place(x=100, y=150, height=50, width=292  )
    # creating line with underscore label
    lbl1 = Label(frame4, text=" _____", 
bg="white").place(x=160, y=280)
    lbl1 = Label(frame4, text=" _____", 
bg="white").place(x=160, y=310)
    lbl2 = Label(frame4, text=" _____", 
bg="white").place(x=160, y=340)
    lbl3 = Label(frame4, text=" _____", 
bg="white").place(x=160, y=370)
    lbl4 = Label(frame4, text=" _____", 
bg="white").place(x=160, y=400)
    lbl6 = Label(frame4, text=" _____", 
bg="white").place(x=160, y=430)
```

```
    # Create labels for the statistics
    mean_label = Label(frame4, text="", bg="white")
    mean_label.place(x=190, y=300)

    mean_total_label = Label(frame4, text="", bg="white")
    mean_total_label.place(x=190, y=330)

    mean_minimum_label = Label(frame4, text="", bg="white")
    mean_minimum_label.place(x=190, y=360)

    mean_maximum_label = Label(frame4, text="", bg="white")
    mean_maximum_label.place(x=190, y=390)
    mean_count_label = Label(frame4, text="", bg="white")
    mean_count_label.place(x=190, y=420)

    # Create a button to calculate the statistics
    mean_calculate_button = Button(frame4, text="Calculate",
command=calculate_mean,width = 10,relief="ridge", bd=3, "lavender",cursor="hand2")
    mean_calculate_button.place(x=250, y=500)

    # Create a button to clear the input and result labels
mean_clear_button=Button(frame4text="Clearcommand=clear_input_mean,width=10,relief
="ridge", bd=3,bg = "lavender",cursor="hand2")
mean_clear_button.place(x=150, y=500)

def median_page():
    bgLabel3 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel3.place(x=0, y=0)
    frame7 = Frame(login_window, width=500, height=600, bg="white",
highlightbackground="black", highlightthickness=1)
    frame7.place(x=900, y=100)

    lblinmedian = Label(login_window, text="", bg="light steel blue", font=("Calibri ",
40),padx=300,pady=280,highlightcolor="black",highlightbackground="black",highlightthi
ckness=1)
    lblinmedian.place(x=150, y=120)

    median_intro = Label(login_window,
```

```
                text="Welcome to the\nMedian "
                    "Solver\nThis tool "
                    "helps you\nto calculate "
                    "the\nmedian of a\nset "
                    "of numbers\nall you have to do "
                    "is\ninput the numbers you\nwant to calculate "
                    "median\nseperated by comas and\nthe solver will"
                    "do\nthe rest for you.",
                bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    median_intro.place(x=185, y=180)

    def closethemedian():
        bgLabel3.destroy()
        frame29.destroy()
        median_intro.destroy()
        lblinmedian.destroy()
        frame7.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",highlightthickness=1, bd=0)
    frame29.place(x=1400, y=57)
    btn10 = Button(frame29, image=closeiicon, command=closethemedian, cursor="hand2",
bg="red", bd=0)
    btn10.place(x=-4, y=-3)
    def calculate_median():
        # Get the numbers_mean from the entry field and convert them to a list of floats
        numbers_median = list(map(float, median_number_entry.get().split(",")))
        # Calculate the statistics and display them in the labels
        median_count = len(numbers_median)
        median_minimum = min(numbers_median)
        median_maximum = max(numbers_median)
        median_total = sum(numbers_median)
        median = statistics.median(numbers_median)
        median_count_label.config(text=f"Count {median_count}")
        median_minimum_label.config(text=f"Minimum {median_minimum:.2f}")
        median_maximum_label.config(text=f"Maximum {median_maximum:.2f}")
        median_total_label.config(text=f"Sum {median_total:.2f}")
        median_label.config(text=f"Median {median:.2f}")

    def clear_input_median():
```

```
        # Clear the entry field and the result labels
        median_number_entry.delete(0, END)
        median_count_label.config(text="")
        median_minimum_label.config(text="")
        median_maximum_label.config(text="")
        median_total_label.config(text="")
        median_label.config(text=f"")
    # note
    lbl_note = Label(frame7, text="Note :", highlightbackground="grey",
highlightthickness=0, fg="white",bg="white").place(x=107, y=200)
    lbl_note = Label(frame7, text="Enter numbers seperated by the comas", fg="saddle
brown", highlightbackground="red",highlightthickness=0, bg="white").place(x=145,
y=200)
    # design
    lbl2 = Label(frame7, text="MEDIAN SOLVER", bg="lavender", fg="black",
borderwidth=1, relief="solid",
                 font=('Times', 30, "bold"), padx=77, pady=30).place(x=0, y=0)
    lbl3 = Label(frame7, text="", highlightbackground="grey", highlightthickness=1,
bg="white", padx=175,
                 pady=100).place(x=70, y=260)

    lbl4 = Label(frame7, text="Answer :", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=75, y=268)

    lbl4 = Label(frame7, text="Data Set:", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=73, y=125)

    # Create labels and entry widgets
    median_number_entry = Entry(frame7, bg="white")
    median_number_entry.place(x=100, y=150, height=50, width=292)

    # creating line with underscore label

    lbl1 = Label(frame7, text=" _____", 
bg="white").place(x=160, y=280)
    lbl1 = Label(frame7, text=" _____", 
bg="white").place(x=160, y=310)
    lbl2 = Label(frame7, text=" _____", 
bg="white").place(x=160, y=340)
    lbl3 = Label(frame7, text=" _____",
```

```
bg="white").place(x=160, y=370)
   lbl4 = Label(frame7, text=" _____", 
bg="white").place(x=160, y=400)
   lbl6 = Label(frame7, text=" _____", 
bg="white").place(x=160, y=430)

   # Create labels for the statistics

   median_label = Label(frame7, text="", bg="white")
   median_label.place(x=190, y=300)

   median_total_label = Label(frame7, text="", bg="white")
   median_total_label.place(x=190, y=330)

   median_minimum_label = Label(frame7, text="", bg="white")
   median_minimum_label.place(x=190, y=360)

   median_maximum_label = Label(frame7, text="", bg="white")
   median_maximum_label.place(x=190, y=390)

   median_count_label = Label(frame7, text="", bg="white")
   median_count_label.place(x=190, y=420)

   # Create a button to calculate the statistics
median_calculate_button=Button(frame7,text="Calculate",command=calculate_median,wi
dth=10,relief="ridge",bd=3,bg="lavender",cursor="hand2")
   median_calculate_button.place(x=250, y=500)

   # Create a button to clear the input and result labels
   median_clear_button = Button(frame7, text="Clear", 
command=clear_input_median,width=10,relief="ridge", bd=3,bg = 
"lavender",cursor="hand2")
   median_clear_button.place(x=150, y=500)

def mode_page():
   bgLabel4 = Label(login_window, image=bgImage, bd=2, highlightcolor="black", 
highlightbackground="black")
   bgLabel4.place(x=0, y=0)
   frame8 = Frame(login_window, width=500, height=600, bg="white", 
highlightbackground="black", highlightthickness=1)
```

```
    frame8.place(x=900, y=100)

    def closethemode():
        bgLabel4.destroy()
        frame29.destroy()
        mode_intro.destroy()
        lblinmode.destroy()
        frame8.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1400, y=57)
    btn10 = Button(frame29, image=closeiicon, command=closethemode, cursor="hand2",
bg="red", bd=0)
    btn10.place(x=-4, y=-3)
    lblinmode = Label(login_window, text="", bg="light steel blue", font=("Calibri ", 40),
padx=300, pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblinmode.place(x=150, y=120)

    mode_intro = Label(login_window,
                       text="Welcome to the Mode Solver\nThis tool helps you\nto
calculate the\nmode of a\nset of numbers\nall
you have to do is\ninput the numbers you\nwant to calculate
    mode\nseperated by comas and\nthe solver will do\nthe rest
for you.",
                       bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    mode_intro.place(x=185, y=180)
    def calculate_mode():
        # Get the numbers_mean from the entry field and convert them to a list of floats
        numbers_mode = list(map(float, mode_number_entry.get().split(",")))

        # Calculate the statistics and display them in the labels
        mode_count = len(numbers_mode)
        mode_minimum = min(numbers_mode)
        mode_maximum = max(numbers_mode)
        mode_total = sum(numbers_mode)
        mode = statistics.mode(numbers_mode)
        mode_count_label.config(text=f"Count          {mode_count}")
```

```
        mode_minimum_label.config(text=f"Minimum      {mode_minimum:.2f}")
        mode_maximum_label.config(text=f"Maximum      {mode_maximum:.2f}")
        mode_total_label.config(text=f"Sum            {mode_total:.2f}")
        mode_label.config(text=f"Mode           {mode:.2f}")

    def clear_input_mode():
        # Clear the entry field and the result labels
        mode_number_entry.delete(0, END)
        mode_count_label.config(text="")
        mode_minimum_label.config(text="")
        mode_maximum_label.config(text="")
        mode_total_label.config(text="")
        mode_label.config(text=f"")

    # note
    lbl_note = Label(frame8, text="Note :", highlightbackground="grey",
highlightthickness=0, fg="black",
                     bg="white").place(x=107, y=200)
    lbl_note = Label(frame8, text="Enter numbers seperated by the comas", fg="black",
highlightbackground="grey",
                     highlightthickness=0, bg="white").place(x=145, y=200)

    # design
    lbl2 = Label(frame8, text="MODE SOLVER", bg="lavender", fg="black",
borderwidth=1, relief="solid",
                 font=('Times', 30, "bold"), padx=98, pady=30).place(x=0, y=0)
    lbl3 = Label(frame8, text="", highlightbackground="grey", highlightthickness=1,
bg="white", padx=175,
                 pady=100).place(x=70, y=260)
    lbl4 = Label(frame8, text="Answer :", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=75, y=268)
    lbl4 = Label(frame8, text="Data Set:", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=73, y=125)

    # Create labels and entry widgets
    mode_number_entry = Entry(frame8, bg="white")
    mode_number_entry.place(x=100, y=150, height=50, width=292)

    # creating line with underscore label
```

```
    lbl1 = Label(frame8, text=" _____", 
bg="white").place(x=160, y=280)
    lbl1 = Label(frame8, text=" _____", 
bg="white").place(x=160, y=310)
    lbl2 = Label(frame8, text=" _____", 
bg="white").place(x=160, y=340)
    lbl3 = Label(frame8, text=" _____", 
bg="white").place(x=160, y=370)
    lbl4 = Label(frame8, text=" _____", 
bg="white").place(x=160, y=400)
    lbl6 = Label(frame8, text=" _____", 
bg="white").place(x=160, y=430)

    # Create labels for the statistics

    mode_label = Label(frame8, text="", bg="white")
    mode_label.place(x=190, y=300)

    mode_total_label = Label(frame8, text="", bg="white")
    mode_total_label.place(x=190, y=330)

    mode_minimum_label = Label(frame8, text="", bg="white")
    mode_minimum_label.place(x=190, y=360)

    mode_maximum_label = Label(frame8, text="", bg="white")
    mode_maximum_label.place(x=190, y=390)

    mode_count_label = Label(frame8, text="", bg="white")
    mode_count_label.place(x=190, y=420)

    # Create a button to calculate the statistics
    mode_calculate_button = Button(frame8, text="Calculate", 
command=calculate_mode,width=10,relief="ridge", bd=3,bg = 
"lavender",cursor="hand2")
    mode_calculate_button.place(x=250, y=500)

    # Create a button to clear the input and result labels
    mode_clear_button = Button(frame8, text="Clear", 
command=clear_input_mode,width=10,relief="ridge", bd=3,bg = 
"lavender",cursor="hand2")
```

```
    mode_clear_button.place(x=150, y=500)

def harmonic_mean_page():
    bgLabel5 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel5.place(x=0, y=0)
    frame9 = Frame(login_window, width=500, height=600, bg="white",
highlightbackground="black", highlightthickness=1)
    frame9.place(x=900, y=100)
    lblinharmonic = Label(login_window, text="", bg="light steel blue", font=("Calibri ",
40), padx=300, pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblinharmonic.place(x=150, y=120)

    harmonic_intro = Label(login_window,

                text="Welcome to the\nHarmonic Mean Solver\nThis tool helps you\nto calculate the\nharmonic mean of a\nset of numbers\nall you have to do is\ninput the numbers\nseperated by comas and\nthe solver will do\nthe rest for you.",
                bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    harmonic_intro.place(x=185, y=180)
    def clear_harmonic_mean():
        harmonic_mean_entry.delete(0, END)
        # result_value.config(text="")
        harmonic_mean_count_label.config(text="")
        harmonic_mean_minimum_label.config(text="")
        harmonic_mean_maximum_label.config(text="")
        harmonic_mean_total_label.config(text="")
        harmonic_mean_label.config(text=f"")


    def calculate_harmonic_mean():
        # Get the numbers from the entry field and convert them to a list of floats
        numbers = list(map(float, harmonic_mean_entry.get().split(",")))

        # Calculate the harmonic mean
        harmonic_mean = statistics.harmonic_mean(numbers)
```

```
        # Display the result
        # result_value.config(text=f"{harmonic_mean:.2f}")
        harmonic_mean = statistics.harmonic_mean(numbers)
        harmonic_mean_count = len(numbers)
        harmonic_mean_minimum = min(numbers)
        harmonic_mean_maximum = max(numbers)
        harmonic_mean_total = sum(numbers)
        harmonic_mean_count_label.config(text=f"Count
{harmonic_mean_count}")
        harmonic_mean_minimum_label.config(text=f"Minimum
{harmonic_mean_minimum:.2f}")
        harmonic_mean_maximum_label.config(text=f"Maximum
{harmonic_mean_maximum:.2f}")
        harmonic_mean_total_label.config(text=f"Sum
{harmonic_mean_total:.2f}")
        harmonic_mean_label.config(text=f"Harmonic Mean          {harmonic_mean:.2f}")

    def closetheharmonic():
        bgLabel5.destroy()
        frame29.destroy()
        harmonic_intro.destroy()
        lblinharmonic.destroy()
        frame9.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1400, y=57)
    btn10 = Button(frame29, image=closeiicon, command=closetheharmonic,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)
    lbl2 = Label(frame9, text="HARMONIC MEAN SOLVER", bg="lavender",
fg="black", borderwidth=1, relief="solid",
                 font=('Times', 25, "bold"), padx=26, pady=30).place(x=0, y=0)

    lbl_note = Label(frame9, text="Note :", highlightbackground="grey",
highlightthickness=0, fg="black",
                     bg="white").place(x=107, y=200)
    lbl_note = Label(frame9, text="Enter numbers seperated by the comas", fg="black",
highlightbackground="grey",
```

```
        highlightthickness=0, bg="white").place(x=145, y=200)

    lbl3 = Label(frame9, text="", highlightbackground="grey", highlightthickness=1,
bg="white", padx=175,
                 pady=100).place(x=70, y=260)
    lbl4 = Label(frame9, text="Answer :", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=75, y=268)

    lbl4 = Label(frame9, text="Data Set:", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=73, y=125)

    lbl1 = Label(frame9, text=" _____",
bg="white").place(x=160, y=280)
    lbl1 = Label(frame9, text=" _____",
bg="white").place(x=160, y=310)
    lbl2 = Label(frame9, text=" _____",
bg="white").place(x=160, y=340)
    lbl3 = Label(frame9, text=" _____",
bg="white").place(x=160, y=370)
    lbl4 = Label(frame9, text=" _____",
bg="white").place(x=160, y=400)
    lbl6 = Label(frame9, text=" _____",
bg="white").place(x=160 ,y=430)

    harmonic_mean_label = Label(frame9, text="", bg="white")
    harmonic_mean_label.place(x=190, y=300)

    harmonic_mean_count_label = Label(frame9, text="", bg="white")
    harmonic_mean_count_label.place(x=190, y=330)
    harmonic_mean_total_label = Label(frame9, text="", bg="white")
    harmonic_mean_total_label.place(x=190, y=360)

    harmonic_mean_minimum_label = Label(frame9, text="", bg="white")
    harmonic_mean_minimum_label.place(x=190, y=390)
    harmonic_mean_maximum_label = Label(frame9, text="", bg="white")
    harmonic_mean_maximum_label.place(x=190, y=420)

    # Create the input label and entry
    harmonic_mean_entry = Entry(frame9,bg = "white")
    harmonic_mean_entry.place(x=100, y=150, height=50, width=292)
```

```
    # Create the output label and result label
    # Create the calculate and clear buttons
    calculate_button = Button(frame9, text="Calculate",
command=calculate_harmonic_mean,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    calculate_button.place(x=250, y=500)

    harmonic_mean_clear_button = Button(frame9, text="Clear",
command=clear_harmonic_mean,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    harmonic_mean_clear_button.place(x=150, y=500)

def geometric_mean_page():
    bgLabel6 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel6.place(x=0, y=0)
    frame10 = Frame(login_window, width=500, height=600, bg="white",
highlightbackground="black", highlightthickness=1)
    frame10.place(x=900, y=100)
    lblingeometric = Label(login_window, text="", bg="light steel blue", font=("Calibri ",
40), padx=300, pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblingeometric.place(x=150, y=120)

    geometric_intro = Label(login_window,

                text="Welcome to the\nGeometric Mean Solver\nThis
tool helps you\nto calculate the\ngeometric mean of a\nset
of numbers\nall you have to do is\ninput the
numbers\nseperated by comas and\nthe solver will do\nthe rest
for you.",
                bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    geometric_intro.place(x=185, y=180)

    def closethegeometric():
        bgLabel6.destroy()
        frame29.destroy()
        geometric_intro.destroy()
```

```
        lblingeometric.destroy()
        frame10.destroy()
    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1400, y=57)
    btn10 = Button(frame29, image=closeiicon, command=closethegeometric,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

    def clear_geometric_mean():
        geometric_mean_entry.delete(0, END)
        # result_value.config(text="")
        geometric_mean_count_label.config(text="")
        geometric_mean_minimum_label.config(text="")
        geometric_mean_maximum_label.config(text="")
        geometric_mean_total_label.config(text="")
        geometric_mean_label.config(text="")

    # Create a function to calculate the geometric mean
    def calculate_geometric_mean():
        # Get the input values from the Entry widget
        numbers = geometric_mean_entry.get().split(",")
        # Convert the values to floats using the map function
        numbers = list(map(float, numbers))
        # Calculate the product of all the values using the math library
        product = math.prod(numbers)
        # Calculate the geometric mean
        geometric_mean = product ** (1 / len(numbers))
        # Display the result to the user using a Label widget
        geometric_mean_label.config(text=f"Geometric Mean: {geometric_mean}")

        # Calculate the geometric mean
        # geometric_mean = statistics.geometric_mean(numbers)
        # Display the result
        # result_value.config(text=f"{geometric_mean:.2f}")
        # geometric_mean = statistics.geometric_mean(numbers)
        geometric_mean_count = len(numbers)
        geometric_mean_minimum = min(numbers)
```

```
        geometric_mean_maximum = max(numbers)
        geometric_mean_total = sum(numbers)

        geometric_mean_count_label.config(text=f"Count
{geometric_mean_count}")
        geometric_mean_minimum_label.config(text=f"Minimum
{geometric_mean_minimum:.2f}")
        geometric_mean_maximum_label.config(text=f"Maximum
{geometric_mean_maximum:.2f}")
        geometric_mean_total_label.config(text=f"Sum
{geometric_mean_total:.2f}")

    lbl2 = Label(frame10, text="GEOMETRIC MEAN SOLVER", bg="lavender",
fg="black", borderwidth=1, relief="solid",
                 font=('Times', 25, "bold"), padx=17, pady=30).place(x=0, y=0)
    lbl_note = Label(frame10, text="Note :", highlightbackground="grey",
highlightthickness=0, fg="purple",
                     bg="white").place(x=107, y=200)
    lbl_note = Label(frame10, text="Enter numbers seperated by the comas",
fg="purple", highlightbackground="grey",
                     highlightthickness=0, bg="white").place(x=145, y=200)

    lbl3 = Label(frame10, text="", highlightbackground="grey", highlightthickness=1,
bg="white", padx=175,
                 pady=100).place(x=70, y=260)
    lbl4 = Label(frame10, text="Answer :", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=75, y=268)

    lbl4 = Label(frame10, text="Data Set:", bg="white", font=('Arial Rounded MT', 9,
'bold')).place(x=73, y=125)

    lbl1 = Label(frame10, text=" _____",
bg="white").place(x=160, y=280)
    lbl1 = Label(frame10, text=" _____",
bg="white").place(x=160, y=310)
    lbl2 = Label(frame10, text=" _____",
bg="white").place(x=160, y=340)
    lbl3 = Label(frame10, text=" _____",
bg="white").place(x=160, y=370)
    lbl4 = Label(frame10, text=" _____",
```

```
bg="white").place(x=160, y=400)
    lbl6 = Label(frame10, text=" _____", 
bg="white").place(x=160, y=430)
    geometric_mean_label = Label(frame10, text="", bg="white")

    geometric_mean_label.place(x=190, y=300)

    geometric_mean_count_label = Label(frame10, text="", bg="white")
    geometric_mean_count_label.place(x=190, y=330)

    geometric_mean_total_label = Label(frame10, text="", bg="white")
    geometric_mean_total_label.place(x=190, y=360)

    geometric_mean_minimum_label = Label(frame10, text="", bg="white")
    geometric_mean_minimum_label.place(x=190, y=390)

    geometric_mean_maximum_label = Label(frame10, text="", bg="white")
    geometric_mean_maximum_label.place(x=190, y=420)

    # Create the input label and entry

    geometric_mean_entry = Entry(frame10,bg="white")
    geometric_mean_entry.place(x=100, y=150, height=50, width=292)

    # Create the output label and result label
    # Create the calculate and clear buttons
    calculate_button = Button(frame10, text="Calculate", 
command=calculate_geometric_mean,width=10,relief="ridge", bd=3,bg = 
"lavender",cursor="hand2")
    calculate_button.place(x=250, y=500)

    clear_button = Button(frame10, text="Clear", 
command=clear_geometric_mean,width=10,relief="ridge", bd=3,bg = 
"lavender",cursor="hand2")
    clear_button.place(x=150, y=500)

def areaoftraingle_page():
    bgLabel7 = Label(login_window, image=bgImage, bd=2, highlightcolor="black", 
highlightbackground="black")
    bgLabel7.place(x=0, y=0)
```

```
    frame11 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame11.place(x=900, y=150)
    lblin_area_of_triangle = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300, pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)

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

    area_of_triangle_intro = Label(login_window,

                        text="Welcome to the Area of\nTriangle  Solver  this\n"
                            "tool  helps  you  to\ncalculate  the  area\n"
                            "of  a  triangle  all\nyou  have  to  do  is\n"
                            "input  the  base  and\nheight and the  solver\n"
                            "will  do  the  rest"
                            "\nfor you.                    ",
                        bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    area_of_triangle_intro.place(x=185, y=180)

    def calc_triangle():
        base = float(entry4.get())
        height = float(entry5.get())
        area = 0.5 * base * height
        result_label_triangle.config(text=f"Area of triangle: {area:.2f}")

    def clear_triangle():
        entry4.delete(0, END)
        entry5.delete(0, END)
        result_label_triangle.config(text="")
    lbl11 = Label(frame11, text="AREA OF TRIANGLE", highlightbackground="black",
highlightthickness=1, bg="lavender", padx=99, pady=20,font=('Times', 15, "bold"))
    lbl11.place(x=-1, y=-1)
    lbl12 = Label(frame11, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)
    lbl12.place(x=40, y=115)
    label4 = Label(frame11, text="Enter base:",bg = "lavender")
    label4.place(x=110, y=185)
    entry4 = Entry(frame11,width = 13)
    entry4.place(x=190, y=185)
```

```
    label5 = Label(frame11, text="Enter height:",bg = "lavender")
    label5.place(x=110, y=210)
    entry5 = Entry(frame11,width = 13)
    entry5.place(x=190, y=210)
    button3 = Button(frame11, text="Calculate", command=calc_triangle,relief="ridge",
bd=3,bg = "lavender",width=10,cursor="hand2")
    button3.place(x=220, y=290)
    result_label_triangle = Label(frame11, text="",bg = "lavender")

    result_label_triangle.place(x=100, y=240)
    clear_button = Button(frame11, text="Clear",
command=clear_triangle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    clear_button.place(x=110, y=290)

    def closethetriangle():
        bgLabel7.destroy()
        frame29.destroy()
        area_of_triangle_intro.destroy()
        lblin_area_of_triangle.destroy()
        frame11.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethetriangle, cursor="hand2",
bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def areaofcircle_page():
    bgLabel8 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel8.place(x=0, y=0)
    frame12 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame12.place(x=900, y=150)
    lblin_area_of_circle = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                                 pady=280, highlightcolor="black",
```

```
highlightbackground="black",highlightthickness=1)
    lblin_area_of_circle.place(x=150, y=120)

    area_of_circle_intro = Label(login_window,

                                 text="Welcome to the Area of\nCircle Solver this\n"
                                 "tool helps you to\ncalculate the area\n"
                                 "of a circle all\nyou have to do is\n"
                                 "input the radius\nand the solver\n"
                                 "will do the rest"
                                 "\nfor you.                    ",
                                  bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    area_of_circle_intro.place(x=185, y=180)

    def calc_circle():
        radius = float(entry1.get())
        area = math.pi * (radius ** 2)
        result_label_circle.config(text=f"Area of circle: {area:.2f}")

    def clear_circle():
        entry1.delete(0, END)
        result_label_circle.config(text="")


    lbl11 = Label(frame12, text="AREA OF CIRCLE", highlightbackground="black",
highlightthickness=1, bg="lavender",
                  padx=113, pady=20,font=('Times',15,'bold'))
    lbl11.place(x=-1, y=-1)

    lbl13 = Label(frame12, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)
    lbl13.place(x=40, y=115)

    label1 = Label(frame12, text="Enter radius", bg="lavender")
    label1.place(x=160, y=165)
    entry1 = Entry(frame12)
    entry1.place(x=130, y=200)
    button1 = Button(frame12, text="Calculate",
command=calc_circle,width=10,relief="ridge", bd=3,bg = "lavender",cursor="hand2")
    button1.place(x=220, y=290)
```

```
    result_label_circle = Label(frame12, text="", bg="lavender")
    result_label_circle.place(x=130, y=240)
    clear_button = Button(frame12, text="Clear",
command=clear_circle,width=10,relief="ridge", bd=3,bg = "lavender",cursor="hand2")
    clear_button.place(x=110, y=290)

    def closethecircle():
        bgLabel8.destroy()
        frame29.destroy()
        area_of_circle_intro.destroy()
        lblin_area_of_circle.destroy()
        frame12.destroy()


    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethecircle, cursor="hand2",
bg="red", bd=0)
    btn10.place(x=-4, y=-3)
def areaofrectangle_page():
    bgLabel9 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel9.place(x=0, y=0)
    frame13 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame13.place(x=900, y=150)
    lblin_area_of_rectangle = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                          pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_area_of_rectangle.place(x=150, y=120)

    area_of_rectangle_intro = Label(login_window,

                                    text="Welcome to the Area of\nRectangle Solver this\n"
                                    "tool helps you to\ncalculate the area\n"
                                    "of a rectangle all\nyou have to do is\n"
                                    "input the length and\nwidth and the solver\n"
```

```
                    "will do the rest"
                    "\nfor you.                    ",
                    bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    area_of_rectangle_intro.place(x=185, y=180)

    def calc_rectangle():
        length = float(entry2.get())
        width = float(entry3.get())
        area = length * width
        result_label_rectangle.config(text=f"Area of rectangle: {area:.2f}")

    def clear_rectangle():
        entry2.delete(0, END)
        entry3.delete(0, END)
        result_label_rectangle.config(text="")


    lbl9 = Label(frame13, text="AREA OF RECTANGLE",
highlightbackground="black", highlightthickness=1, bg="lavender",
                 padx=89, pady=20,font=("Times",15,"bold"))
    lbl9.place(x=-1, y=-1)

    lbl10 = Label(frame13, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)
    lbl10.place(x=40, y=115)

    label2 = Label(frame13, text="Enter length:", bg="lavender")
    label2.place(x=110, y=165)
    entry2 = Entry(frame13, width=13)
    entry2.place(x=205, y=165)
    label3 = Label(frame13, text="Enter width:", bg="lavender")
    label3.place(x=110, y=190)
    entry3 = Entry(frame13, width=13)
    entry3.place(x=205, y=190)
    button2 = Button(frame13, text="Calculate",
command=calc_rectangle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    button2.place(x=220, y=290)
    button5 = Button(frame13, text="Clear", command=clear_rectangle,
width=10,relief="ridge", bd=3,bg = "lavender",cursor="hand2")
```

```
    button5.place(x=110, y=290)

    result_label_rectangle = Label(frame13, text="", bg="lavender")
    result_label_rectangle.place(x=110, y=240)

    def closetherectangle():
        bgLabel9.destroy()
        frame29.destroy()
        area_of_rectangle_intro.destroy()
        lblin_area_of_rectangle.destroy()
        frame13.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closetherectangle,
cursor="hand2", bg="red", bd=0)

    btn10.place(x=-4, y=-3)
def areaofsquare_page():
    bgLabel10 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel10.place(x=0, y=0)
    frame14 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame14.place(x=900, y=150)
    lblin_area_of_square = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                         pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_area_of_square.place(x=150, y=120)

    area_of_square_intro = Label(login_window,

                          text="Welcome to the Area of\nSquare Solver this\n"
                          "tool helps you to\ncalculate the area\n"
                          "of a square all\nyou have to do is\n"
                          "input the side\nand the solver\n"
                          "will do the rest"
```

```
                    "\nfor you.                    ",
                    bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    area_of_square_intro.place(x=185, y=180)

    def calc_square():
        side = float(entry6.get())
        area = side ** 2
        result_label_square.config(text=f"Area of square: {area:.2f}")

    def clear_square():
        entry6.delete(0, END)
        result_label_square.config(text="")


    lbl7 = Label(frame14, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)
    lbl7.place(x=40, y=115)

    lbl8 = Label(frame14, text="AREA OF SQUARE", highlightbackground="black",
highlightthickness=1, bg="lavender", padx=110, pady=20,font=("Times",15,"bold"))

    lbl8.place(x=-1, y=-1)

    label6 = Label(frame14, text="Enter side", bg="lavender")
    label6.place(x=160, y=165)
    entry6 = Entry(frame14)
    entry6.place(x=130, y=200)
    button4 = Button(frame14, text="Calculate", command=calc_square,relief="ridge",
bd=3,bg = "lavender",width=10,cursor="hand2")
    button4.place(x=220, y=290)
    button4 = Button(frame14, text="Clear", command=clear_square,
width=10,relief="ridge", bd=3,bg = "lavender",cursor="hand2")
    button4.place(x=110, y=290)

    result_label_square = Label(frame14, text="", bg="lavender")
    result_label_square.place(x=130, y=240)

    def closethesquare():
        bgLabel10.destroy()
        frame29.destroy()
```

```
        area_of_square_intro.destroy()
        lblin_area_of_square.destroy()
        frame14.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethesquare, cursor="hand2",
bg="red", bd=0)
    btn10.place(x=-4, y=-3)
def perimeterofsquare_page():
    bgLabel11 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel11.place(x=0, y=0)
    frame15 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame15.place(x=900, y=150)
    lblin_perimeter_of_square = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                              pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)

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

    perimeter_of_square_intro = Label(login_window,

                              text="Welcome to the Perimeter\nof Square Solver this\n"
                              "tool helps you to\ncalculate the perimeter\n"
                              "of a square all\nyou have to do is\n"
                              "input the side\nand the solver\n"
                              "will do the rest"
                              "\nfor you.                    ",
                              bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    perimeter_of_square_intro.place(x=185, y=180)

    def calculate_perimeter_square():
        perimeter_square = 4 * float(side_entry.get())
        perimeter_value_square.config(text=f"Perimeter: {perimeter_square}")
```

```
    def clear_content_square():
        side_entry.delete(0, END)
        perimeter_value_square.config(text="")


    lbl11 = Label(frame15, text="PERIMETER OF SQUARE",
highlightbackground="black", highlightthickness=1, bg="lavender",
                  padx=79, pady=20, font=('Times', 15, "bold"))
    lbl11.place(x=-1, y=-1)

    lbl12 = Label(frame15, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)
    lbl12.place(x=40, y=115)

    side_label = Label(frame15, text="Enter Side", bg="lavender")
    side_label.place(x=160, y=165)

    side_entry = Entry(frame15)
    side_entry.place(x=130, y=200)

    calculate_button = Button(frame15, text="Calculate",
command=calculate_perimeter_square,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    calculate_button.place(x=220, y=290)

    perimeter_value_square = Label(frame15, text="", bg="lavender")
    perimeter_value_square.place(x=130, y=240)

    clear_button = Button(frame15, text="Clear",
command=clear_content_square,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    clear_button.place(x=110, y=290)

    def closethesquare_perimeter():
        bgLabel11.destroy()
        frame29.destroy()
        perimeter_of_square_intro.destroy()
        lblin_perimeter_of_square.destroy()
        frame15.destroy()
```

```
    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethesquare_perimeter,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def perimeter_of_circle_page():
    bgLabel12 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel12.place(x=0, y=0)
    frame16 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame16.place(x=900, y=150)
    lblin_perimeter_of_circle = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                          pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_perimeter_of_circle.place(x=150, y=120)

    perimeter_of_circle_intro = Label(login_window,

                          text="Welcome to the Perimeter\nof Circle Solver this\n"
                        "tool helps you to\ncalculate the perimeter\n"
                        "of a circle all\nyou have to do is\n"
                        "input the radius\nand the solver\n"

                        "will do the rest"
                        "\nfor you.                    ",
                          bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    perimeter_of_circle_intro.place(x=185, y=180)
    def calculate_perimeter_circle():
        # Get the radius from the input field
        radius = float(radius_entry.get())
        # Calculate the perimeter of the circle
        perimeter_circle = 2 * 3.14159 * radius
        # Update the label text with the result
        perimeter_label_circle.config(text=f"Perimeter: {perimeter_circle:.2f}")
```

```
    # Create a function to clear the input field and label
    def clear_fields_circle():
        radius_entry.delete(0, END)
        perimeter_label_circle.config(text="")

    lbl9 = Label(frame16, text="PERIMETER OF CIRCLE",
highlightbackground="black", highlightthickness=1, bg="lavender",
                 padx=82, pady=20, font=('Times', 15, "bold"))
    lbl9.place(x=-1, y=-1)
    lbl10 = Label(frame16, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)
    lbl10.place(x=40, y=115)

    radius_label_circle = Label(frame16, text="Enter the radius", bg="lavender")
    radius_label_circle.place(x=160, y=165)

    # Create an input field for the radius
    radius_entry = Entry(frame16)
    radius_entry.place(x=130, y=200)

    # Create a button to calculate the perimeter
    calculate_button = Button(frame16, text="Calculate",
command=calculate_perimeter_circle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    calculate_button.place(x=220, y=290)
    # Create a button to clear the input field and label
    clear_button = Button(frame16, text="Clear",
command=clear_fields_circle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")

    clear_button.place(x=110, y=290)

    # Create a label to display the perimeter
    perimeter_label_circle = Label(frame16, text="", bg="lavender")
    perimeter_label_circle.place(x=130, y=240)

    def closethecircle_perimeter():
        bgLabel12.destroy()
        frame29.destroy()
        perimeter_of_circle_intro.destroy()
```

```
        lblin_perimeter_of_circle.destroy()
        frame16.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethecircle_perimeter,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def perimeter_of_rectangle_page():
    bgLabel13 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel13.place(x=0, y=0)
    frame17 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame17.place(x=900, y=150)
    lblin_perimeter_of_rectangle = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                                pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_perimeter_of_rectangle.place(x=150, y=120)
    perimeter_of_rectangle_intro = Label(login_window,
                                text="Welcome to the Perimeter\nof Rectangle Solver this\n"
                            "tool helps you to\ncalculate the perimeter\n"
                            "of a rectangle all\nyou have to do is\n"
                            "input the length and width\nand the solver\n"
                            "will do the rest"
                            "\nfor you. ",
                                bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")

    perimeter_of_rectangle_intro.place(x=185, y=180)

    def calculate_perimeter_rectangle():
        length = float(length_entry.get())
        width = float(width_entry.get())
        perimeter_rectangle = 2 * (length + width)
        perimeter_value_rectangle.config(text=f"Perimeter: {perimeter_rectangle}")
```

```
    def clear_content_rectangle():
        length_entry.delete(0, END)
        width_entry.delete(0, END)
        perimeter_value_rectangle.config(text="")

    lblpr = Label(frame17, text="", highlightbackground="black", highlightthickness=1,
bg="lavender", padx=155,
                  pady=110)
    lblpr.place(x=40, y=115)

    length_label = Label(frame17, text="Length :", bg="lavender")
    length_label.place(x=110, y=165)

    length_entry = Entry(frame17)

    length_entry.place(x=180, y=165)

    width_label = Label(frame17, text="Width :", bg="lavender")
    width_label.place(x=110, y=190)

    width_entry = Entry(frame17)

    width_entry.place(x=180, y=190)

    calculate_button = Button(frame17, text="Calculate",
command=calculate_perimeter_rectangle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    calculate_button.place(x=220, y=290)

    perimeter_value_rectangle = Label(frame17, text="", bg="lavender")
    perimeter_value_rectangle.place(x=160, y=240)

    clear_button = Button(frame17, text="Clear",
command=clear_content_rectangle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    clear_button.place(x=110, y=290)

    lbl11 = Label(frame17, text="PERIMETER OF RECTANGLE",
highlightbackground="black", highlightthickness=1, bg="lavender",
                  padx=58, pady=20, font=('Times', 15, "bold"))
```

```
    lbl11.place(x=-1, y=-1)

    def closetherectangle_perimeter():
        bgLabel13.destroy()
        frame29.destroy()
        perimeter_of_rectangle_intro.destroy()
        lblin_perimeter_of_rectangle.destroy()
        frame17.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closetherectangle_perimeter,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def perimeter_of_triangle_page():
    bgLabel14 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel14.place(x=0, y=0)
    frame18 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black", highlightthickness=1)
    frame18.place(x=900, y=150)
    lblin_perimeter_of_triangle = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                                pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_perimeter_of_triangle.place(x=150, y=120)

    #lbl4 = Label(frame18, text="", highlightbackground="black", highlightthickness=1,
bg="white", padx=120, pady=100)
    #lbl4.place(x=210, y=120)
    lbl10 = Label(frame18, text="", highlightbackground="black", highlightthickness=1,
bg="lavender", padx=155, pady=110)

    lbl10.place(x=40, y=115)

    perimeter_of_triangle_intro = Label(login_window,
```

```
                    text="Welcome to the Perimeter\nof Triangle Solver this\n"
                    "tool helps you to\ncalculate the perimeter\n"
                    "of a triangle all\nyou have to do is\n"
                    "input the sides\nand the solver\n"
                    "will do the rest"
                    "\nfor you.                    ",
                    bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    perimeter_of_triangle_intro.place(x=185, y=180)

    lbl7 = Label(frame18, text="", highlightbackground="grey", highlightthickness=1,
bg="yellow1", padx=100, pady=60)
    lbl7.place(x=330, y=410)

    lbl8 = Label(frame18, text="PERIMETER OF TRIANGLE",
highlightbackground="black", highlightthickness=1, bg="lavender",
            padx=68, pady=20, font=('times', 15, "bold"))
    lbl8.place(x=-1, y=-1)

    def calculate_perimeter_triangle():
        # Get the sides from the input fields
        side1 = float(side1_entry.get())
        side2 = float(side2_entry.get())
        side3 = float(side3_entry.get())
        # Calculate the perimeter of the triangle
        perimeter_triangle = side1 + side2 + side3
        # Update the label text with the result
        perimeter_label_triangle.config(text=f"Perimeter: {perimeter_triangle:.2f}")

    # Create a function to clear the input fields and label
    def clear_fields_triangle():
        side1_entry.delete(0, END)
        side2_entry.delete(0, END)
        side3_entry.delete(0, END)
        perimeter_label_triangle.config(text="")
    # Create a label for the sides input fields
    sides_label = Label(frame18, text="Enter the sides", bg="lavender")
    sides_label.place(x=153, y=145)

    # Create input fields for the sides of the triangle
    side1_entry = Entry(frame18)
```

```
    side1_entry.place(x=130, y=170)

    side2_entry = Entry(frame18)
    side2_entry.place(x=130, y=200)

    side3_entry = Entry(frame18)
    side3_entry.place(x=130, y=230)

    # Create a button to calculate the perimeter
    calculate_button = Button(frame18, text="Calculate",
command=calculate_perimeter_triangle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    calculate_button.place(x=220, y=290)

    # Create a button to clear the input fields and label
    clear_button = Button(frame18, text="Clear",
command=clear_fields_triangle,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    clear_button.place(x=110, y=290)

    # Create a label to display the perimeter
    perimeter_label_triangle = Label(frame18, text="", bg="lavender")
    perimeter_label_triangle.place(x=145, y=255)

    def closethetriangle_perimeter():
        bgLabel14.destroy()
        frame29.destroy()
        perimeter_of_triangle_intro.destroy()
        lblin_perimeter_of_triangle.destroy()
        frame18.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethetriangle_perimeter,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def volume_of_cube_page():
```

```
    bgLabel15 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel15.place(x=0, y=0)
    frame19 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black",
                    highlightthickness=1)
    frame19.place(x=900, y=150)
    lblin_volume_of_cube = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                                 pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_volume_of_cube.place(x=150, y=120)

    # lbl4 = Label(frame18, text="", highlightbackground="black", highlightthickness=1,
bg="white", padx=120, pady=100)
    # lbl4.place(x=210, y=120)
    lbl10 = Label(frame19, text="", highlightbackground="black", highlightthickness=1,
bg="lavender", padx=155,
                  pady=110)
    lbl10.place(x=40, y=115)
    volume_of_cube_intro = Label(login_window,

                                 text="Welcome to the  Volume\nof  Cube Solver  this\n"
                                 "tool  helps  you  to\ncalculate  the  volume\n"
                                 "of  a  cube  all\nyou  have  to  do  is\n"
                                 "input  the  length\nand  the  solver\n"
                                 "will  do  the  rest"
                                 "\nfor you.                    ",
                                 bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    volume_of_cube_intro.place(x=185, y=180)

    lbl7 = Label(frame19, text="", highlightbackground="grey", highlightthickness=1,
bg="lightslateblue", padx=100,
                 pady=60)
    lbl7.place(x=330, y=410)

    lbl8 = Label(frame19, text="VOLUME OF CUBE", highlightbackground="black",
highlightthickness=1, bg="lavender",
                 padx=107, pady=20, font=('times', 15, "bold"))
```

```
    lbl8.place(x=-1, y=-1)


    def calculate_volume():
        # get user input for side length
        side_length = float(entry_side_length.get())

        # calculate volume of cube
        volume_cube = side_length ** 3

        # update label with result
        label_result_cube.config(text=f"Volume: {volume_cube}")

    def clear_inputs():
        # clear the side length entry and result label
        entry_side_length.delete(0, END)
        label_result_cube.config(text="")
    # cube

    label_side_length = Label(frame19, text="Side Length:", bg="lavender")
    label_side_length.place(x=162, y=170)

    entry_side_length = Entry(frame19)
    entry_side_length.place(x=135, y=195)

    # create button to calculate volume
    button_calculate = Button(frame19, text="Calculate",
command=calculate_volume,width = 10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    button_calculate.place(x=220, y=290)

    # create button to clear inputs
    button_clear = Button(frame19, text="Clear",
command=clear_inputs,width=10,relief="ridge", bd=3,bg = "lavender",cursor="hand2")
    button_clear.place(x=110, y=290)

    # create label to display result
    label_result_cube = Label(frame19, text="", bg="lavender")
    label_result_cube.place(x=150, y=230)
```

```
    def closethecube_volume():
        bgLabel15.destroy()
        frame29.destroy()
        volume_of_cube_intro.destroy()

        lblin_volume_of_cube.destroy()
        frame19.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethecube_volume,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def volume_of_cylinder_page():
    bgLabel16 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel16.place(x=0, y=0)
    frame20 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black",
                    highlightthickness=1)
    frame20.place(x=900, y=150)
    lblin_volume_of_cylinder = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                                     pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_volume_of_cylinder.place(x=150, y=120)

    lbl10 = Label(frame20, text="", highlightbackground="black", highlightthickness=1,
bg="lavender", padx=155,
                  pady=110)
    lbl10.place(x=40, y=115)
    volume_of_cylinder_intro = Label(login_window,

                                     text="Welcome to the  Volume\nof  Cylinder  Solver  this\n"
                                     "tool  helps  you  to\ncalculate  the  volume\n"
                                     "of  a  cylinder  all\nyou  have  to  do  is\n"
                                     "input  the  height & radius\nand  the  solver\n"
```

```
                    "will do the rest"
                    "\nfor you.                    ",
                    bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    volume_of_cylinder_intro.place(x=185, y=180)
    lbl4 = Label(frame20, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)

    lbl4.place(x=40, y=115)
    lbl8 = Label(frame20, text="VOLUME OF CYLINDER",
highlightbackground="black", highlightthickness=1,
            bg="lavender",
            padx=83, pady=20, font=('times', 15, "bold"))
    lbl8.place(x=-1, y=-1)

    def calculate():
        radius = float(radius_entry.get())
        height = float(height_entry.get())
        volume1 = 3.14 * radius ** 2 * height
        volume_label.config(text="Volume: {:.2f}".format(volume1))

    def clear():
        radius_entry.delete(0, END)
        height_entry.delete(0, END)
        volume_label.config(text="")

    radius_label = Label(frame20, text="Radius:", bg="lavender")
    radius_label.place(x=110, y=165)

    radius_entry = Entry(frame20)
    radius_entry.place(x=175, y=165)

    height_label = Label(frame20, text="Height:", bg="lavender")
    height_label.place(x=110, y=190)

    height_entry = Entry(frame20)
    height_entry.place(x=175, y=190)

    calculate_button = Button(frame20, text="Calculate",
command=calculate,width=10,relief="ridge", bd=3,bg = "lavender",cursor="hand2")
    calculate_button.place(x=220, y=290)
```

```
    clear_button = Button(frame20, text="Clear", command=clear,width=10,relief="ridge",
bd=3,bg = "lavender",cursor="hand2")
    clear_button.place(x=110, y=290)

    volume_label = Label(frame20, bg="lavender")
    volume_label.place(x=145, y=240)


    def closethecylinder_volume():
        bgLabel16.destroy()
        frame29.destroy()
        volume_of_cylinder_intro.destroy()
        lblin_volume_of_cylinder.destroy()
        frame20.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethecylinder_volume,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def volume_of_sphere_page():
    bgLabel17 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel17.place(x=0, y=0)
    frame21 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black",
                    highlightthickness=1)
    frame21.place(x=900, y=150)

    lblin_volume_of_sphere = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
                                   pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_volume_of_sphere.place(x=150, y=120)

    volume_of_sphere_intro = Label(login_window,
```

```
                    text="Welcome to the Volume\nof Sphere Solver this\n"
                    "tool helps you to\ncalculate the volume\n"
                    "of a sphere all\nyou have to do is\n"
                    "input the radius\nand the solver\n"
                    "will do the rest"
                    "\nfor you.                    ",
                    bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    volume_of_sphere_intro.place(x=185, y=180)
    lbl4 = Label(frame21, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)

    lbl4.place(x=40, y=115)
    lbl8 = Label(frame21, text="VOLUME OF SPHERE", highlightbackground="black",
highlightthickness=1,
            bg="lavender",
            padx=95, pady=20, font=('times', 15, "bold"))
    lbl8.place(x=0, y=-1)

    def calculate_volume_sphere():
        # get user input for radius
        radius = float(entry_radius.get())
        # calculate volume of sphere
        volume_sphere = (4 / 3) * 3.14 * radius ** 3

        # update label with result
        label_result_sphere.config(text=f"Volume: {volume_sphere:.2f}")

    def clear_inputs_sphere():
        # clear the radius entry and result label
        entry_radius.delete(0, END)
        label_result_sphere.config(text="", bg="lavender")

    # create label and entry for radius
    label_radius = Label(frame21, text="Radius:", bg="lavender")
    label_radius.place(x=170, y=165)

    entry_radius = Entry(frame21)
    entry_radius.place(x=130, y=200)
    # create button to calculate volume
```

```
    button_calculate = Button(frame21, text="Calculate",
command=calculate_volume_sphere,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    button_calculate.place(x=220, y=290)
    # create button to clear inputs
    button_clear = Button(frame21, text="Clear",
command=clear_inputs_sphere,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    button_clear.place(x=110, y=290)

    # create label to display result
    label_result_sphere = Label(frame21, text="", bg="lavender")
    label_result_sphere.place(x=140, y=240)


    def closethesphere_volume():
        bgLabel17.destroy()
        frame29.destroy()
        volume_of_sphere_intro.destroy()
        lblin_volume_of_sphere.destroy()
        frame21.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                    highlightthickness=1, bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethesphere_volume,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def volume_of_cone_page():
    bgLabel18 = Label(login_window, image=bgImage, bd=2, highlightcolor="black",
highlightbackground="black")
    bgLabel18.place(x=0, y=0)
    frame22 = Frame(login_window, width=400, height=400, bg="white",
highlightbackground="black",
                    highlightthickness=1)
    frame22.place(x=900, y=150)
    lblin_volume_of_cone = Label(login_window, text="", bg="light steel blue",
font=("Calibri ", 40), padx=300,
```

```
                        pady=280, highlightcolor="black",
highlightbackground="black",highlightthickness=1)
    lblin_volume_of_cone.place(x=150, y=120)

    volume_of_cone_intro = Label(login_window,

                          text="Welcome to the  Volume\nof  Cone Solver  this\n"
                         "tool  helps  you  to\ncalculate  the  volume\n"
                         "of  a  cone  all\nyou  have  to  do  is\n"
                         "input  the  radius  &\nheight  and  the  solver\n"
                         "will  do  the  rest"
                         "\nfor you.                    ",
                          bg="light steel blue", font=("Calibri ", 30, "bold"), fg="black")
    volume_of_cone_intro.place(x=185, y=180)
    lbl4 = Label(frame22, text="", highlightbackground="grey", highlightthickness=1,
bg="lavender", padx=155, pady=110)

    lbl4.place(x=40, y=115)

    lbl8 = Label(frame22, text="VOLUME OF CONE", highlightbackground="black",
highlightthickness=1,
                 bg="lavender",
                 padx=106, pady=20, font=('times', 15, "bold"))
    lbl8.place(x=-1, y=-1)

    def calculate_volume():
        radius = float(radius_entry.get())
        height = float(height_entry.get())
        volume = (1 / 3) * math.pi * radius ** 2 * height
        volume_label.config(text="Volume: {:.2f}".format(volume))

    # Function to clear entry fields and result label
    def clear_fields():
        radius_entry.delete(0, END)
        height_entry.delete(0, END)
        volume_label.config(text="")

    # Create labels and entries
    radius_label = Label(frame22, text="Enter radius:",bg ="lavender")
    radius_label.place(x=100, y=165)
```

```
    radius_entry = Entry(frame22,width=17)
    radius_entry.place(x=180, y=165)

    height_label = Label(frame22, text="Enter height:",bg ="lavender")
    height_label.place(x=100, y=190)
    height_entry = Entry(frame22,width=17)
    height_entry.place(x=180, y=190)

    # Create button to calculate volume
    calculate_button = Button(frame22, text="Calculate",
command=calculate_volume,width=10,relief="ridge", bd=3,bg =
"lavender",cursor="hand2")
    calculate_button.place(x=210, y=290)

    # Create button to clear fields
    clear_button = Button(frame22, text="Clear",
command=clear_fields,width=10,relief="ridge", bd=3,bg = "lavender",cursor="hand2")
    clear_button.place(x=100, y=290)

    # Create label to display result
    volume_label = Label(frame22, text="",bg = "lavender")
    volume_label.place(x=145, y=240)

    def closethecone_volume():
        bgLabel18.destroy()
        frame29.destroy()
        volume_of_cone_intro.destroy()
        lblin_volume_of_cone.destroy()
        frame22.destroy()

    frame29 = Frame(login_window, width=45, height=45, bg="white",
highlightbackground="lemon chiffon2",
                highlightthickness=1,bd=0)
    frame29.place(x=1300, y=107)
    btn10 = Button(frame29, image=closeiicon, command=closethecone_volume,
cursor="hand2", bg="red", bd=0)
    btn10.place(x=-4, y=-3)

def helpp():
    global  frame4
```

```
    frame4 = Frame(login_window, width=905, height=665, bg="alice blue",bd =
1,highlightcolor="black",highlightthickness=2,highlightbackground="black")
    frame4.place(x=400, y=60)
    lbl6 = Label(frame4,text = "Welcome to the Help section of the MathSolver,",bg =
"alice blue",font=("arial",20,"bold"))
    lbl6.place(x = 10,y = 50+80)

    lbl7= Label(frame4,text = "If you are facing any issues and you need help to ",bg =
"alice blue",font=("arial",15,"bold"))
    lbl7.place(x = 10,y = 100+80)

    lbl8 = Label(frame4,text = "resolve the issue the you can contact us through :",bg =
"alice blue",fg = "black",font=("arial",15,"bold"))
    lbl8.place(x = 10,y = 140+80)

    lbl9 = Label(frame4,text="E-mail: xxxxxxxx@gmail.com",fg = "black",bg = "alice
blue"

,font=("arial",15),highlightcolor="red",highlightthickness=2,highlightbackground="red",
width = 26)
    lbl9.place(x= 270, y = 190+110)

    lbl10 = Label(frame4, text="facebook: xxxxxxxxxxxxx", bg="alice blue", fg="black",
                  font=("arial",
15,),highlightcolor="yellow",highlightthickness=2,highlightbackground="yellow",width =
26)
    lbl10.place(x=270, y=220+110+20)

    lbl11 = Label(frame4, text="phone no: xxxxxxxxxx", bg="alice blue",
                  fg="black",
                  font=("arial",
15,),highlightcolor="green",highlightthickness=2,highlightbackground="green",width =
26)
    lbl11.place(x=270, y=300+110+40)

    lbl12 = Label(frame4, text="Instagram: xxxxxxxxxx", bg="alice blue",
                  fg="black",
                  font=("arial",
15,),highlightcolor="orange",highlightthickness=2,highlightbackground="orange",width
= 26)
```

```
    lbl12.place(x=270, y=250+110+40)

    lbl13 = Label(frame4, text="Help", bg="alice blue",
                  fg="black",
                  font=("arial", 25,"bold","underline"))
    lbl13.place(x=10, y=30)

    lbl13 = Label(frame4, text="Website: wwww.xxxxxxxxxx.com", bg="alice blue",
                  fg="black",
                  font=("arial", 15,),highlightcolor="slate
blue2",highlightthickness=2,highlightbackground="slate blue2",width = 26)
    lbl13.place(x=270, y=310+110+80)

    lbl13 = Label(frame4, text="Version  1.0", bg="alice blue",
                  fg="black",
                  font=("arial",
15,),highlightcolor="firebrick2",highlightthickness=2,highlightbackground="sky
blue",width = 26)
    lbl13.place(x=270, y=340+110+100)

    def closethehelp():
        frame4.destroy()
    btn6 = Button(frame4,image=closeiicon,command=closethehelp,cursor="hand2",bg =
"alice blue",bd = 0)

    btn6.place(x = 835,y = 0)

def aboutt():
    global frame30
    frame30 = Frame(login_window, width=905, height=665, bg="alice blue", bd=1,
highlightcolor="black",
                    highlightthickness=2, highlightbackground="black")
    frame30.place(x=400, y=60)

    def closetheaboutt():
        frame30.destroy()

    lbl13 = Label(frame30, text="About", bg="alice blue",
                  fg="black",
                  font=("arial", 25, "bold", "underline"))
```

```
    lbl13.place(x=10, y=30)

    btn6 = Button(frame30, image=closeiicon, command=closetheaboutt, cursor="hand2",
bg="alice blue", bd=0)
    btn6.place(x=835, y=0)

    lbl6 = Label(frame30, text="Welcome to the About section of the MathSolver",
bg="alice blue",font=("arial", 20, "bold"))
    lbl6.place(x=130, y=100)

    lbl7 = Label(frame30,text="      MathSolver is a graphical user interface
application\nthat allows  you to solve mathematical and statistical\nproblems
with  ease. It  provides  a  user - friendly\ninterface  for  performing
various  mathematical\noperations and statistical calculations.                "
                , bg="alice blue",font=("arial", 15,
"bold"),highlightcolor="red",highlightthickness=2,highlightbackground="red",width =
50)
    lbl7.place(x=140,y=180)
    lbl8 = Label(frame30,
                text="      MathSolver is designed to be simple, making\nit suitable for both
beginners and advanced  users.\nthis provides accurate results and supports a wide
\nrange  of mathematical operations, making it  a \nhandy tool for students,
educators,  researchers,\nand professionals alike.                "
                , bg="alice blue", font=("arial", 15,
"bold"),highlightcolor="orange",highlightthickness=2,highlightbackground="orange",wi
dth = 50)

    lbl8.place(x=140, y=325)

    lbl8 = Label(frame30,
                text="      MathSolver aims to make mathematical and
statistical\nCalculations  easy, convenient,  and accessible  to everyone.\nWe  hope
you  find  MathSolver  useful  in  solving  your\nmathematical and statistical
problems!                "
                , bg="alice blue", font=("arial", 15,
"bold"),highlightcolor="green",highlightthickness=2,highlightbackground="green",width
= 50)
    lbl8.place(x=140, y=495)
def tandc():
    def closethetandc():
```

```
    frame31.destroy()
  def previous_page_of_tandc():
    #lblnextpage.destroy()

    lbl6first_page_of_tandc = Label(frame31, text="Terms and Conditions\n\n"
                "Welcome to Math Solver! By accessing and using our MathSolver
GUI,\n"
                "you  are  agreeing  to be bound by the following terms and
conditions.\n"
                "Please read them carefully before using our services.
\n\n"
                "Acceptance of Terms:By accessing and using Math Solver,you
agree to\n"
                "These terms and conditions, as  well as  any  additional terms that
may \n"
                "be provided within the GUI.
\n\n"
                "Use of the Math Solver GUI:  Math  Solver  is  a  tool  designed  to
solve\n"
                "Mathematical  and statistical  problems. It  is  intended for
educational\n"
                "and informational  purposes only. While  we strive for  accuracy,
we do\n"
                "not guarantee the accuracy,completeness, or reliability of the
solutions\n"
                "generated  by the math solver GUI. It is your responsibility to
verify the\n"
                "accuracy of the results obtained using the GUI.
                    \n\n"
                "Privacy:  We  respect  your  privacy  and  will  handle  any
personal\n"
                "information you provide in accordance with our Privacy Policy. By
using\n"
                "Math Solver, you consent to the collection, use, and  disclosure  of
your\n"
                "personal information as described in our Privacy Policy.
\n\n"
                "Intellectual  Property:  You  are  granted  a  limited ,  non –
exclusive ,\n"
                "Non - transferable,  revocable  license  to  use Math Solver for
```

```
personal,\n"
                    "Non - commercial use only . You may not copy , modify ,
distribute,", bg="alice blue",font=("arial",15, "bold"))
      lbl6first_page_of_tandc.place(x=100, y=50)

   def next_page_of_tandc():
      lbl6first_page_of_tandc.destroy()

      lblnxtpage=Label(frame31,text=
                    "Display , transmit , or otherwise exploit Math Solver or its
content\n"
                    "without our prior written consent.
\n\n"
                    "Prohibited Use: You agree not to use Math Solver for any
unlawful\n"
                    "or unauthorized purpose , including  but not limited to
Uploading or\n"
                    "transmitting any content that is illegal,
offensive,abusive,discriminatory,\n"
                    "or harmful in any way . Attempting  to gain unauthorized
access to\n"
                    "Math Solver, its servers, or any other systems or networks
connected to\n"
                    "Math Solver. Interfering  with or disrupting the operation  of Math
Solver\n"
                    "or its  associated  services . Reverse  engineering  ,  decompiling ,
or\n"
                    "disassembling Math Solver, or attempting to derive  its source
code.\n"
                    "Limitation of Liability:  Solver  and  its  owners , officers
,employees, and\n"
                    "Affiliates shall not  be  liable  for  any  direct, indirect, incidental,
special,\n"
                    "or consequential damages arising out of or in connection with the
use of\n"
                    "Math Solver,including but not limited to damages for loss of profits,
data,\n"
                    "or  other  intangible  losses  ,  even  if  Math  Solver  has
been\n"
                    "advised of the possibility of such damages.
```

```
\n",
                    bg="alice blue",font=("arial",15, "bold"),height=25)
        lblnxtpage.place(x = 100,y = 50)

    global frame31
    frame31 = Frame(login_window, width=905, height=665, bg="alice blue", bd=1,
highlightcolor="black",
                highlightthickness=2, highlightbackground="black")
    frame31.place(x=400, y=60)

    lbl6first_page_of_tandc = Label(frame31, text="Terms and Conditions\n\n"
                    "Welcome to Math Solver! By accessing and using our MathSolver
GUI,\n"
                    "you  are  agreeing  to be bound by the following terms and
conditions.\n"
                    "Please read them carefully before using our services.
\n\n"
                    "Acceptance of Terms:By accessing and using Math Solver,you
agree to\n"
                    "These terms and conditions, as  well as  any  additional terms that
may \n"
                    "be provided within the GUI.
\n\n"
                    "Use of the Math Solver GUI:  Math  Solver  is  a  tool  designed  to
solve\n"
                    "Mathematical  and statistical  problems. It  is  intended for
educational\n"
                    "and informational  purposes only. While  we strive for  accuracy,
we do\n"
                    "not guarantee the accuracy,completeness, or reliability of the
solutions\n"
                    "generated  by the math solver GUI. It is your responsibility to
verify the\n"
                    "accuracy of the results obtained using the GUI.
\n\n"
                    "Privacy:  We  respect  your  privacy  and  will  handle  any
personal\n"
                    "information you provide in accordance with our Privacy Policy. By
using\n"
                    "Math Solver, you consent to the collection, use, and  disclosure  of
```

```
your\n"
                    "personal information as described in our Privacy Policy.
\n\n"
                    "Intellectual Property: You are granted a limited , non –
exclusive ,\n"
                    "Non - transferable, revocable license to use Math Solver for
personal,\n"
                    "Non - commercial use only . You may not copy , modify ,
distribute,", bg="alice blue",font=("arial",15, "bold"))
    lbl6first_page_of_tandc.place(x=100, y=50)

    btn2 = Button(frame31,text="▲",bd= 0,bg = "alice
blue",cursor="hand2",font=("arial",15),command=previous_page_of_tandc)
    btn2.place(x = 860,y = 590)

    btn3 = Button(frame31, text="▼", bd=0, bg="alice blue", cursor="hand2",
font=("arial", 15),
                  command=next_page_of_tandc)
    btn3.place(x = 860,y = 620)

    btn6 = Button(frame31, image=closeiicon, command=closethetandc, cursor="hand2",
bg="alice blue", bd=0)
    btn6.place(x=835, y=0)

def forget_pass():
    def change_password():
        if user_entry.get()== '' or password_entry.get() =='' or confirmpass_entry.get() =='':
            messagebox.showerror('Error','All Fields Are Required',Parent = frame24)

        elif password_entry.get()!=confirmpass_entry.get():
            messagebox.showerror('Error','Password and Confirm Password are not
matching',Parent = frame24)
        else:
            con = pymysql.connect(host='localhost', user='root',
password='root',database='userdata')

            mycursor = con.cursor()
            query = 'select * from data where username =%s'
            mycursor.execute(query,(user_entry.get()))
            row = mycursor.fetchone()
```

```
        if row == None:
            messagebox.showerror('Error','Incorrect Username',parent= frame24)

        else:
            query="update data set password=%s where username=%s"
            mycursor.execute(query,(password_entry.get(),user_entry.get()))
            con.commit()
            con.close()
            messagebox.showinfo('Success','Password is reset,please login with new
password', parent = frame24)
            frame24.destroy()

    global frame24
    frame24 = Frame(login_window, width=400, height=600, bg="white")
    frame24.place(x=904, y=60)

    my_pic5 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\reset_password2.png
")
    resized5 = my_pic5.resize((100, 100),Image.ANTIALIAS)

    global  signinIm5
    signinIm5 = ImageTk.PhotoImage(resized5)

    img1Labe5 = Label(frame24, image=signinIm5, bg="white")
    img1Labe5.place(x=180, y=50)

    heading_label = Label(frame24, text="RESET PASSWORD", font=('arial', 22, 'bold'),
bg="white"
                          , fg="slateblue2")
    heading_label.place(x=60, y=150)

    userLabel = Label(frame24, text="Username", font=("Microsoft Yahei UI Light", 11,
'bold'), bg="white",
                      fg="slateblue2")
    userLabel.place(x=60, y=250)
    user_entry = Entry(frame24, width=25, fg="slateblue2", font=("Microsoft Yahei UI
Light", 11, 'bold'), bd=0)
    user_entry.place(x=60, y=279)
```

```
    Frame(frame24, width=277, height=2, bg="slateblue2").place(x=60, y=300)

    passwordLabel = Label(frame24, text="New Password", font=("Microsoft Yahei UI
Light", 11, 'bold'), bg="white",
                          fg="slateblue2")
    passwordLabel.place(x=60, y=320)

    password_entry = Entry(frame24, width=25, fg="slateblue2", font=("Microsoft Yahei
UI Light", 11, 'bold'), bd=0)
    password_entry.place(x=60, y=350)

    Frame(frame24, width=277, height=2, bg="slateblue2").place(x=60, y=370)

    confirmpassLabel = Label(frame24, text="Confirm Password", font=("Microsoft
Yahei UI Light", 11, 'bold'),
                             bg="white",
                             fg="slateblue2")
    confirmpassLabel.place(x=60, y=400)

    confirmpass_entry = Entry(frame24, width=25, fg="slateblue2", font=("Microsoft
Yahei UI Light", 11, 'bold'), bd=0)
    confirmpass_entry.place(x=60, y=430)

    Frame(frame24, width=277, height=2, bg="slateblue2").place(x=60, y=450)

    submitButton = Button(frame24, text="Submit", bd=0, bg="slateblue2", fg="white",
font=('Open Sans', 16, 'bold')
                          , width=21, cursor='hand2', activebackground="slateblue2"
                          , activeforeground="white",command=change_password)

    submitButton.place(x=60, y=500)

    def hide_confirm_pass():
        openeye2.config(file='closeye.png')
        confirmpass_entry.config(show='*')

        eyeButton_for_confirm_pass.config(command=show_confirm_pass)
    def show_confirm_pass():
        openeye2.config(file='openeye.png')
```

```
        confirmpass_entry.config(show='')
        eyeButton_for_confirm_pass.config(command=hide_confirm_pass)

    def closetheforgetpass():
        passwordEntrylogin.delete(0, END)
        usernameEntrylogin.delete(0, END)
        frame24.destroy()

    openeye2 = PhotoImage(file="openeye.png")

    eyeButton_for_confirm_pass = Button(frame24, image=openeye2, bd=0, bg="white",
activebackground="white"
                    , cursor="hand2", command=hide_confirm_pass)
    eyeButton_for_confirm_pass.place(x=300, y=424)

    btn20 = Button(frame24, image=closeiicon, command=closetheforgetpass,
cursor="hand2", bg="white", bd=0)
    btn20.place(x=350, y=0)
def login_user():

    if usernameEntrylogin.get()=='' or passwordEntrylogin.get()='':
        messagebox.showerror('Error','All Fields Are Required')

    else:
        try:
            con=pymysql.connect(host='localhost',user='root',password='root')
            mycursor=con.cursor()

        except:
            messagebox.showerror('Error','Connection is not established try again')
            return
        query = "use userdata"
        mycursor.execute(query)
        query = 'select * from data where username=%s and password = %s'
        mycursor.execute(query, (usernameEntrylogin.get(),passwordEntrylogin.get()))
        row=mycursor.fetchone()
        if row == None:
            messagebox.showerror('Error','Invalid username or password')
```

```
    else:
        messagebox.showinfo('Welcome','Login is successful')


        #login_window.destroy()
        #bgLabel = Label(login_window, image=bgImage,bd =
2,highlightcolor="black",highlightbackground="black")
        #bgLabel.place(x=100, y=0)1520x850
        frame28 = Frame(login_window,width=1720,height=950,bg = "lemon chiffon")
        frame28.place(x=0, y=0)


        thelabel2 = Label(frame28,text="SELECT THE SOLVER",bg = "lemon
chiffon",font= ("arial",30,"bold"))
        thelabel2.place(x = 600,y = 50)
        #import project_all_prblms

        pythagoras_btn = Button(frame28,text="Pythagoras Solver",bg  = "black",fg =
"white",width=21,font=("Times New Roman",15,"bold"),command =
pythagoras_page,cursor="hand2")
        pythagoras_btn.place(x = 200+150,y = 200)

        mean_btn = Button(frame28, text="Mean Solver", bg="black",fg =
"white",width=21,font=("Times New Roman",15,"bold"),command =
mean_page,cursor="hand2")
        mean_btn.place(x=500+150 , y=200)

        Mode_btn = Button(frame28,text="Mode Solver",bg  = "black",fg =
"white",width=21,font=("Times New
Roman",15,"bold"),command=mode_page,cursor="hand2")
        Mode_btn.place(x = 800+150,y = 200)

        Median_btn = Button(frame28,text="Median Solver",bg  = "black",fg =
"white",width=21,font=("Times New
Roman",15,"bold"),command=median_page,cursor="hand2")
        Median_btn.place(x = 200+150,y = 250+20)

        Harmonic_mean_btn = Button(frame28,text="Harmonic mean Solver",bg  =
"black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=harmonic_mean_page,cursor="hand2")
```

```
        Harmonic_mean_btn.place(x = 500+150,y = 250+20)

        geometric_mean_btn = Button(frame28,text="geometric mean Solver",bg  =
"black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=geometric_mean_page,cursor="hand2")
        geometric_mean_btn.place(x = 800+150,y = 250+20)

        Area_of_triangle_btn = Button(frame28,text="Area of Triangle Solver",bg  =
"black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=areaoftraingle_page,cursor="hand2")
        Area_of_triangle_btn.place(x = 200+150,y = 300+40)

        Area_of_circle_btn = Button(frame28, text="Area of Circle Solver",
bg="black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=areaofcircle_page,cursor="hand2")
        Area_of_circle_btn.place(x=500+150, y=300+40)

        Area_of_rectangle_btn = Button(frame28,text="Area of Rectangle Solver",bg  =
"black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=areaofrectangle_page,cursor="hand2")
        Area_of_rectangle_btn.place(x = 800+150,y = 300+40)

        Area_of_Square_btn = Button(frame28,text="Area of Square Solver",bg  =
"black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=areaofsquare_page,cursor="hand2")
        Area_of_Square_btn.place(x = 200+150,y = 350+60)

        Perimeter_of_Square_btn = Button(frame28, text="Perimeter of Square Solver",
bg="black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=perimeterofsquare_page,cursor="hand2")
        Perimeter_of_Square_btn.place(x=500+150, y=350+60)

        Perimeter_of_Rectangle_btn = Button(frame28, text="Perimeter of Rectangle
Solver", bg="black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=perimeter_of_rectangle_page,cursor="hand2")
        Perimeter_of_Rectangle_btn.place(x=800+150, y=350+60)

        Perimeter_of_Circle_btn = Button(frame28, text="Perimeter of Circle Solver",
bg="black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=perimeter_of_circle_page,cursor="hand2")
```

```
        Perimeter_of_Circle_btn.place(x=200+150, y=400+80)

        Perimeter_of_Triangle_btn = Button(frame28, text="Perimeter of Triangle
Solver", bg="black",fg = "white",width=21,font=("Times New
Roman",15,"bold"),command=perimeter_of_triangle_page,cursor="hand2")
        Perimeter_of_Triangle_btn.place(x=500+150, y=400+80)

        Volume_of_Cube_btn = Button(frame28, text="Volume of Cube Solver",
bg="black", fg="white", width=21,

                            font=("Times New Roman", 15, "bold"),
command=volume_of_cube_page,cursor="hand2")
        Volume_of_Cube_btn.place(x=800+150, y=400+80)

        Volume_of_Cylinder_btn = Button(frame28, text="Volume of Cylindr Solver",
bg="black", fg="white", width=21,
                              font=("Times New Roman", 15, "bold"),
command=volume_of_cylinder_page,cursor="hand2")
        Volume_of_Cylinder_btn.place(x=200+150, y=450+100)

        Volume_of_Sphere_btn = Button(frame28, text="Volume of Sphere Solver",
bg="black", fg="white", width=21,
                               font=("Times New Roman", 15, "bold"),
command=volume_of_sphere_page,cursor="hand2")
        Volume_of_Sphere_btn.place(x=500+150, y=450+100)

        Volume_of_Cone_btn = Button(frame28, text="Volume of Cone Solver",
bg="black", fg="white", width=21,
                            font=("Times New Roman", 15, "bold"),
command=volume_of_cone_page,cursor="hand2")
        Volume_of_Cone_btn.place(x=800+150, y=450+100)

        def closethesums():
            frame28.destroy()
            passwordEntrylogin.delete(0,END)
            usernameEntrylogin.delete(0, END)
        btn6 = Button(frame28, image=closeiicon, command=closethesums,
cursor="hand2", bg="lemon chiffon", bd=0)
        btn6.place(x=1400, y=0)
```

```
def hide():
    openeye.config(file='closeye.png')
    passwordEntrylogin.config(show='*')

    eyeButton.config(command=show)

def show():
    openeye.config(file='openeye.png')
    passwordEntrylogin.config(show='')
    eyeButton.config(command=hide)
def user_enter(event):

    if usernameEntrylogin.get()=="Username":
        usernameEntrylogin.delete(0,END)

def password_enter(event):
    if passwordEntrylogin.get()=="Password":
        passwordEntrylogin.delete(0,END)

login_window=Tk()
login_window.geometry("1520x850")
bgImage = ImageTk.PhotoImage(file ="bg2.jpg")

bgLabel = Label(login_window,image=bgImage)
bgLabel.place(x= 0,y=0)
my_pic = Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\img8.jpg")

resized = my_pic.resize((400,665),Image.ANTIALIAS)

new_pic = ImageTk.PhotoImage(resized)

my_label = Label(login_window,image=new_pic,bd=0)
my_label.place(x = 803+100,y = 60)
#lbl1 = Label(login_window,text = "",bg = 'white',pady=350,padx=250)
#lbl1.place(x = 800, y = 200)

my_pic2 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\img10.png")
resized2 = my_pic2.resize((150,150),Image.ANTIALIAS)
signinImg= ImageTk.PhotoImage(resized2)
```

```
img1Label = Label(login_window,image=signinImg,bg = "white")
img1Label.place(x = 928+100,y = 70)
login_username_label=Label(login_window,text="Username",font=("Microsoft Yahei UI
Light",11,'bold'),bd = 0,fg = "slateblue2",bg="white")
login_username_label.place(x=873+100,y=230)
usernameEntrylogin = Entry(login_window,width=25,font=("Microsoft Yahei UI
Light",11,'bold'),bd = 0
                ,fg = "slateblue2")
usernameEntrylogin.place(x= 873+100,y= 250)

frame1=Frame(login_window,width=250,height=2,bg = "slateblue2")
frame1.place(x= 873+100, y = 272)

login_password_label=Label(login_window,text="Password",font=("Microsoft Yahei UI
Light",11,'bold'),bd = 0,fg = "slateblue2",bg="white")
login_password_label.place(x=873+100,y=290)

passwordEntrylogin = Entry(login_window,width=25,font=("Microsoft Yahei UI
Light",11,'bold'),bd = 0
                ,fg = "slateblue2")
passwordEntrylogin.place(x= 873+100,y= 310)
frame2=Frame(login_window,width=250,height=2,bg = "slateblue2")
frame2.place(x= 873+100, y = 332)

def create_new_account_page():
   def connect_database():
      if emailEntry.get() == "" or usernameEntry.get() == '' or passwordEntry.get() == '' or
confirmEntry.get() == '':
         messagebox.showerror('Error', 'All Fields Are Required')

      elif passwordEntry.get() != confirmEntry.get():
         messagebox.showerror('Error', 'Password Mismatch')

      elif check.get() == 0:
         messagebox.showerror('Error', 'Please Accept Terms & Conditions')

      else:
         try:
            con = pymysql.connect(host="localhost", user='root', password='root')
```

```
            mycursor = con.cursor()
        except:
            messagebox.showerror('Error', "Database Connectivity Issue, Please Try
Again")
            return
        try:
            query = "create database userdata"
            mycursor.execute(query)
            query = "use userdata"
            mycursor.execute(query)
            query = 'create table data(id int auto_increment primary key not null,email
varchar(50),username varchar(100),password varchar(20))'
            mycursor.execute(query)
        except:
            mycursor.execute("use userdata")
        query = 'select * from data where username =%s'

        mycursor.execute(query, (usernameEntry.get()))

        row = mycursor.fetchone()
        if row != None:
            messagebox.showerror('Error', "Username Already exixts")

        else:
            query = "insert into data(email,username,password) values(%s,%s,%s)"
            mycursor.execute(query, (emailEntry.get(), usernameEntry.get(),
passwordEntry.get()))
            con.commit()
            con.close()
            messagebox.showinfo('Sucess', 'Registration Successful')
            clear()
            frame25.destroy()
    def closethesignup():
        passwordEntrylogin.delete(0, END)
        usernameEntrylogin.delete(0, END)
        frame25.destroy()

    global frame25
    frame25 = Frame(login_window, width=400, height=600, bg="white")
    frame25.place(x=904, y=60)
```

```
    my_pic5 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\signup_icon.png")
    resized5 = my_pic5.resize((100, 100), Image.ANTIALIAS)

    global signinIm5
    signinIm5 = ImageTk.PhotoImage(resized5)

    img1Labe5 = Label(frame25, image=signinIm5, bg="white")
    img1Labe5.place(x=160, y=20)

    heading = Label(frame25, text="CREATE AN ACCOUNT", font=('arial', 18, 'bold'),
bg="white"
                    , fg="slateblue2")
    heading.place(x=60, y=130)

    emailLabel = Label(frame25, text="Email", font=("Microsoft Yahei UI Light", 11,
"bold")

                       , bg="white", fg="slateblue2")
    emailLabel.place(x=60, y=180)

    emailEntry = Entry(frame25, width=25, font=("Microsoft Yahei UI Light", 11,
"bold"), bg="white", fg="slateblue2",
                       bd=0)
    emailEntry.place(x=60, y=209)

    Frame(frame25, width=270, height=2, bg="slateblue2").place(x=60, y=230)

    usernameLabel = Label(frame25, text="Username", font=("Microsoft Yahei UI
Light", 11, "bold")
                          , bg="white", fg="slateblue2")
    usernameLabel.place(x=60, y=250)

    usernameEntry = Entry(frame25, width=25, font=("Microsoft Yahei UI Light", 11,
"bold"), bg="white", fg="slateblue2",
                          bd=0)
    usernameEntry.place(x=60, y=280)

    Frame(frame25, width=270, height=2, bg="slateblue2").place(x=60, y=300)
```

```
    passwordLabel = Label(frame25, text="Password", font=("Microsoft Yahei UI Light",
11, "bold")
                          , bg="white", fg="slateblue2")
    passwordLabel.place(x=60, y=330)

    passwordEntry = Entry(frame25, width=25, font=("Microsoft Yahei UI Light", 11,
"bold"), bg="white", fg="slateblue2",
                          bd=0)
    passwordEntry.place(x=60, y=360)

    Frame(frame25, width=270, height=2, bg="slateblue2").place(x=60, y=380)

    confirmLabel = Label(frame25, text="Confirm Password", font=("Microsoft Yahei UI
Light", 11, "bold")
                         , bg="white", fg="slateblue2")
    confirmLabel.place(x=60, y=410)
    confirmEntry = Entry(frame25, width=25, font=("Microsoft Yahei UI Light", 11,
"bold"), bg="white", fg="slateblue2",
                         bd=0)

    confirmEntry.place(x=60, y=440)
    check = IntVar()

    Frame(frame25, width=270, height=2, bg="slateblue2").place(x=60, y=460)

    termsandconditions = Checkbutton(frame25, text="I agree to the Terms & Conditions",
                                     font=("Microsoft Yahei UI Light", 10, "bold")
                                     , bg="white", fg="slateblue2", activeforeground="slateblue2"
                                     , activebackground="white", cursor="hand2", variable=check)
    termsandconditions.place(x=60, y=465)

    signupButton = Button(frame25, text="Signup", font=("Open Sans", 16, "bold")
                          , bd=0, bg="slateblue2", fg="white", activeforeground="white"
                          , activebackground="slateblue2", width=21, cursor="hand2",command =
connect_database)
    signupButton.place(x=60, y=520)

    alreadyaccount = Label(frame25, text="Already have an account?", font=("Open
Sans", 9, "bold")
```

```
                    , bg="white", fg="slateblue2")
    alreadyaccount.place(x=60, y=570)

    loginButton = Button(frame25, text="Log in", font=("Open Sans", 9, "bold
underline"), bg="white"
                    , fg="blue", bd=0, activebackground="white", activeforeground="blue",
command=closethesignup,cursor="hand2")
    loginButton.place(x=200, y=569)

    btn20 = Button(frame25, image=closeiicon, command=closethesignup, cursor="hand2",
bg="white", bd=0)
    btn20.place(x=350, y=0)

    def clear():
        emailEntry.delete(0, END)
        usernameEntry.delete(0, END)
        passwordEntry.delete(0, END)
        confirmEntry.delete(0, END)
        check.set(0)

openeye=PhotoImage(file="openeye.png")
closeiicon = PhotoImage(file="closeicon2.png")

eyeButton = Button(login_window,image=openeye,bd = 0,bg =
"white",activebackground="white"
                ,cursor="hand2",command=hide)
eyeButton.place(x= 1088+100,y = 305)

forgetButton = Button(login_window,text="Forget Password?",bd = 0,bg =
"white",activebackground="white"
                ,cursor="hand2",font=("Microsoft Yahei UI Light",9,'bold')
                  ,fg ='slateblue2',activeforeground="slateblue2",command=forget_pass)
forgetButton.place(x= 1003+100,y = 345)

loginButton = Button(login_window,text="Login",font= ("Open Sans",16,"bold")
                ,fg = "white",bg
="slateblue2",activebackground="slateblue2",activeforeground="white",width= 19
                ,cursor="hand2",bd = 0,command=login_user)

loginButton.place( x = 873+100,y = 400)
```

```
orLabel = Label(login_window,text="----------OR----------",font = ("Open
Sans",16),fg = "black"
            ,bg = "white")
orLabel.place(x = 878+100, y= 450)

facebook_logo = PhotoImage(file="facebook.png")
fbLabel= Label(login_window,image=facebook_logo,bg = "white")
fbLabel.place(x= 1035,y = 490)

google_logo = PhotoImage(file="google.png")
googleLabel= Label(login_window,image=google_logo,bg = "white")
googleLabel.place(x= 984+100,y = 490)

twitter_logo = PhotoImage(file="twitter.png")
twitterLabel= Label(login_window,image=twitter_logo,bg = "white")
twitterLabel.place(x= 1033+100,y = 490)


signupLabel = Label(login_window,text="Dont have an accont?",font = ("Open
Sans",9,"bold"),fg = "slateblue2"
            ,bg = "white")
signupLabel.place(x = 893+100, y= 550)
newaccountButton = Button(login_window,text="Create new one",font= ("Open
Sans",9,"bold underline")

                ,fg = "blue",bg
="white",activebackground="white",activeforeground="blue"
                ,cursor="hand2",bd = 0,command=create_new_account_page)

newaccountButton.place( x = 1023+100,y = 549)

lbl2 = Label(login_window,text= "",bg = "slateblue2",font = ("Calibri
",40),padx=250,pady=300)

lbl2.place(x = 300+100,y = 60)

my_pic22 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\ma.png")
resized22 = my_pic22.resize((150,150),Image.ANTIALIAS)
```

```
signinImg22= ImageTk.PhotoImage(resized22)

img1Label22 = Label(login_window,image=signinImg22,bg = "slate blue2")
img1Label22.place(x = 550,y = 380)

my_pic21 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\curly_brace_open.png")
resized21 = my_pic21.resize((50,70),Image.ANTIALIAS)


signinImg21= ImageTk.PhotoImage(resized21)

img1Label21 = Label(login_window,image=signinImg21,bg = "slate blue2")
img1Label21.place(x = 400,y = 260)

my_pic23 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\curly_brace_close.png")
resized23 = my_pic23.resize((60,120),Image.ANTIALIAS)

signinImg23= ImageTk.PhotoImage(resized23)

img1Label23 = Label(login_window,image=signinImg23,bg = "slate blue2")
img1Label23.place(x = 770,y = 260)
my_pic24 = Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\pie.png")

resized24 = my_pic24.resize((30,30),Image.ANTIALIAS)

lblintro = Label(login_window,text = "Hi!
Welcome",font=("arial",40,"bold"),bg="slateblue2",fg = "white")
lblintro.place(x = 450,y = 265)

lblintro = Label(login_window,text = "To Math
Solver",font=("arial",30,"bold"),bg="slateblue2",fg = "white")
lblintro.place(x = 450,y = 320)

lblintro = Label(login_window,text = "Let's
tackle",font=("arial",15,"bold","underline"),bg="slateblue2",fg = "white")
```

```
lblintro.place(x = 455,y = 360)

lblintro = Label(login_window,text =
"problems",font=("arial",15,"bold","underline"),bg="slateblue2",fg = "white")
lblintro.place(x = 455,y = 386)

lblintro = Label(login_window,text = "of
math",font=("arial",15,"bold","underline"),bg="slateblue2",fg = "white")
lblintro.place(x = 455,y = 412)

lblintro = Label(login_window,text = "quickly
&",font=("arial",15,"bold","underline"),bg="slateblue2",fg = "white")
lblintro.place(x = 455,y = 462)

lblintro = Label(login_window,text =
"together",font=("arial",15,"bold","underline"),bg="slateblue2",fg = "white")
lblintro.place(x = 455,y = 438)

lblintro = Label(login_window,text =
"accurately",font=("arial",15,"bold","underline"),bg="slateblue2",fg = "white")
lblintro.place(x = 455,y = 488)

signinImg24= ImageTk.PhotoImage(resized24)

img1Label24 = Label(login_window,image=signinImg24,bg = "slate blue2")
img1Label24.place(x = 575,y = 420)

lbl106 = Label(login_window,text="18+88",font=("arial",10,"bold"),bg="slateblue2",fg
= "white")

lbl106.place(x = 567,y=450)

def homeredirect():
    try:
        if frame24.winfo_exists():
            frame24.destroy()

    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)
```

```
    try:
        if frame25.winfo_exists():
            frame25.destroy()
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)

    try:
        if frame30.winfo_exists():
            frame30.destroy()
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)

    try:
        if frame4.winfo_exists():
            frame4.destroy()
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)

    try:
        if frame31.winfo_exists():
            frame31.destroy()
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)

#Frame(login_window, width = 2, height = 500, bg = "black").place(x = 771 , y = 60)
frame3 = Frame(login_window,width=200,height=10000,bg = "#2B2B2B")

frame3.place(x=0,y=0)

lbl5 = Label(login_window,text="Menu",bg = "#2B2B2B",font=("arial",15),bd=0,fg =
"white",width=10)
lbl5.place(x = 22,y = 5)

btn1 = Button(login_window,text="     Home",bg = "#2B2B2B",fg =
"white",font=('arial',15)
```

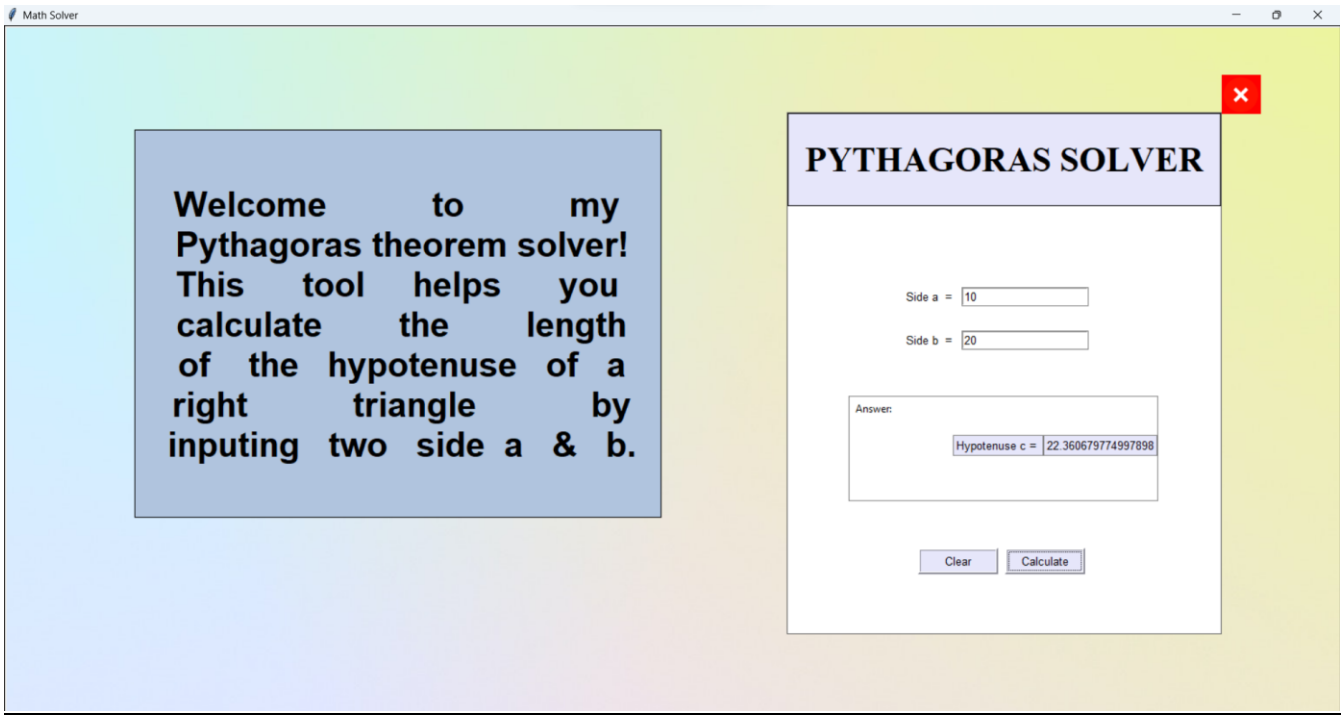
```
        ,width=18,activebackground="black",activeforeground="white",command =
homeredirect,bd=0,cursor="hand2")
btn1.place(x = -4 , y = 52+5)

btn2 = Button(login_window,text="   Help",bg = "#2B2B2B",fg =
"white",font=('arial',15),width=18,activebackground="black",activeforeground="white",
command = helpp,bd=0,cursor="hand2")
btn2.place(x = -4 , y = 102+10)

btn3 = Button(login_window,text="    About",bg = "#2B2B2B",fg =
"white",font=('arial',15),width=18,activebackground="black",activeforeground="white",
bd=0,command = aboutt,cursor="hand2")
btn3.place(x = -4 , y = 152+10)

btn4 = Button(login_window,text="    T & C",bg = "#2B2B2B",fg =
"white",font=('arial',14),width=18,activebackground="black",activeforeground="white",
bd=0,command=tandc,cursor="hand2")
btn4.place(x = -4 , y = 202+10)

def exit_to_desktop():
    login_window.destroy()
btn5 = Button(login_window,text="  Exit",bg = "#2B2B2B",fg =
"white",font=('arial',15),width=18,activebackground="black",activeforeground="white",
bd=0,command=exit_to_desktop,cursor="hand2")
btn5.place(x = -4 , y = 252+10)

def logoutt():
    passwordEntrylogin.delete(0, END)
    usernameEntrylogin.delete(0, END)
    try:
        if frame24.winfo_exists():
            frame24.destroy()
    except Exception as e:

        # Handle the error, e.g. print an error message
        print("Error:", e)

    try:
        if frame25.winfo_exists():
            frame25.destroy()
```

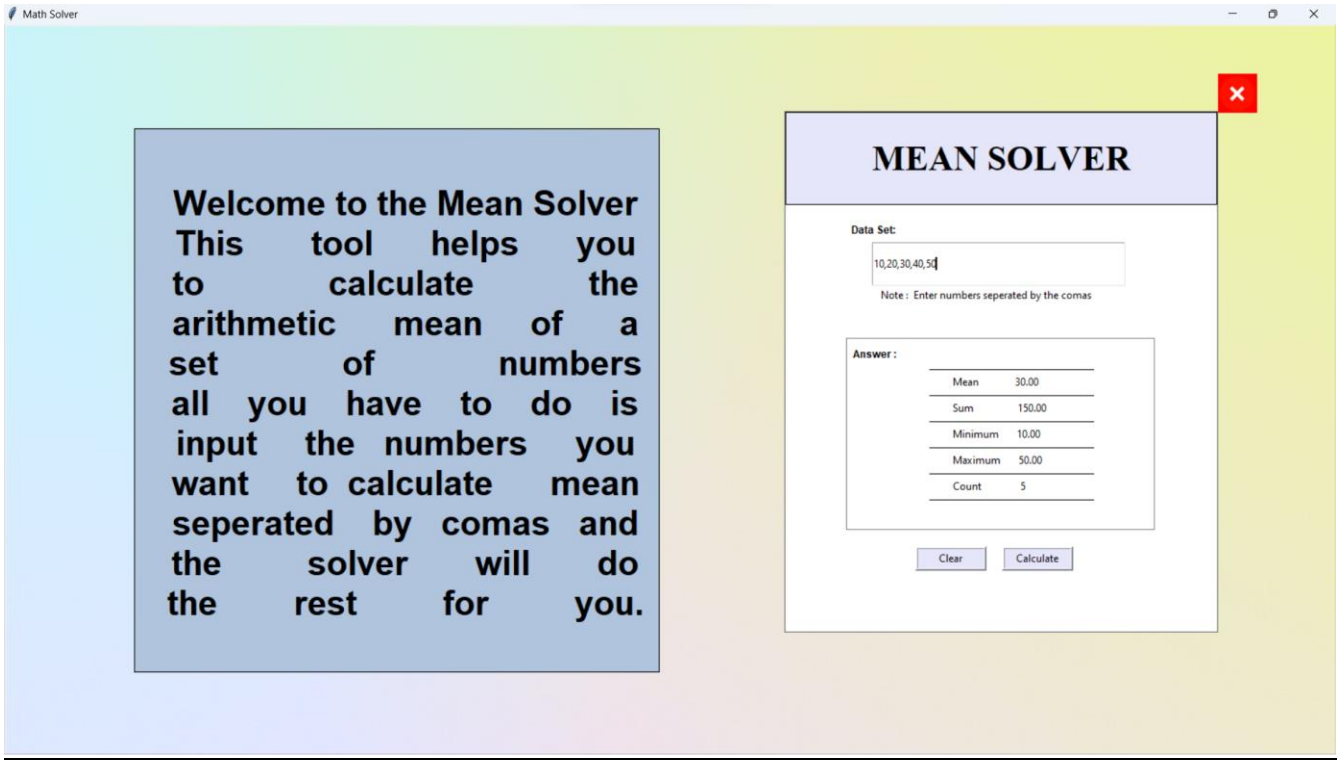
```
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)

    try:
        if frame30.winfo_exists():
            frame30.destroy()
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)

    try:
        if frame4.winfo_exists():
            frame4.destroy()
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)

    try:
        if frame31.winfo_exists():
            frame31.destroy()
    except Exception as e:
        # Handle the error, e.g. print an error message
        print("Error:", e)
btn6 = Button(login_window,text="      Logout",bg = "#2B2B2B",fg =
"white",font=('arial',15),

width=18,activebackground="black",activeforeground="white",bd=0,command=logoutt,c
ursor="hand2")
btn6.place(x = -4 , y = 302+10)

my_pic3 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\menu_open_icon2.pn
g")
resized3 = my_pic3.resize((30,30),Image.ANTIALIAS)

signinIm3= ImageTk.PhotoImage(resized3)

img3Label = Label(login_window,image=signinIm3,bg = "#2B2B2B")
img3Label.place(x = 0,y = -1)
```

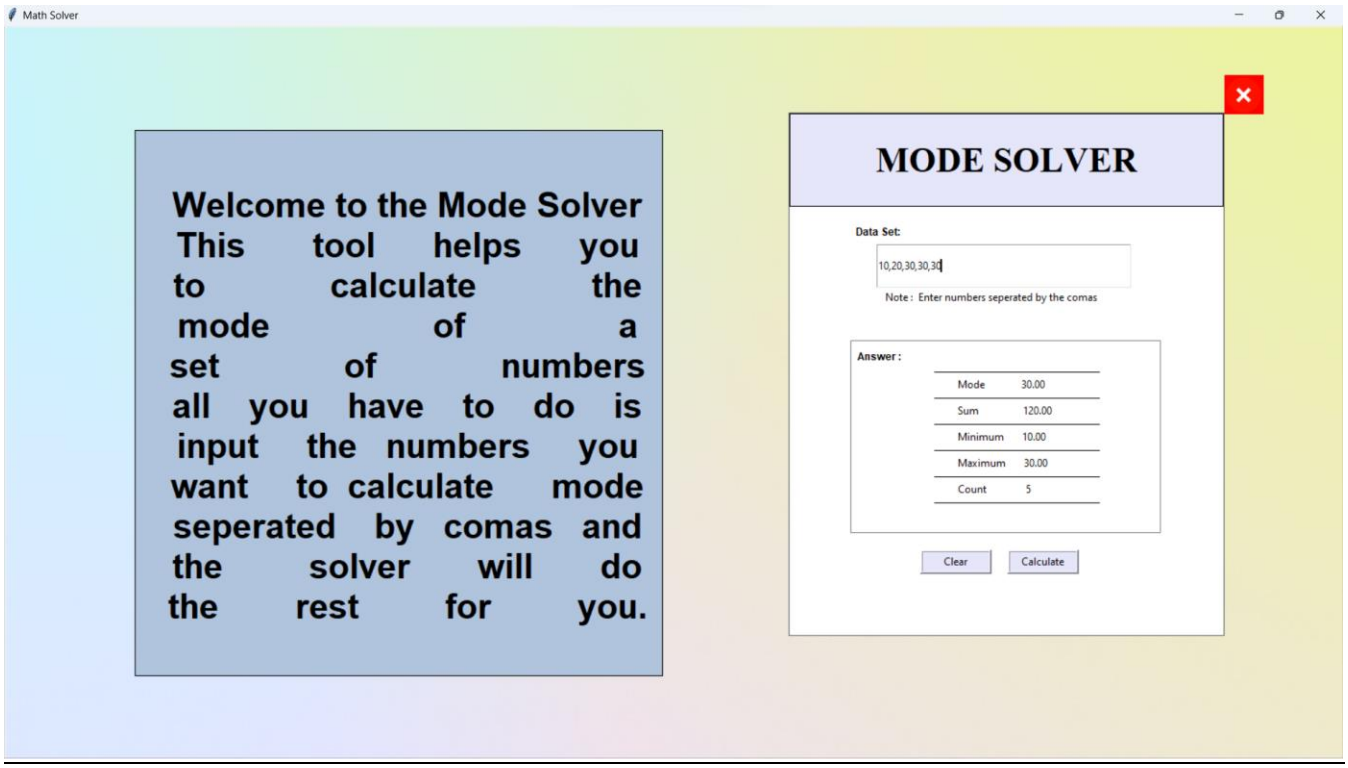
```
my_pic4 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\home_icon.png")
resized4 = my_pic4.resize((25,25),Image.ANTIALIAS)
signinIm4= ImageTk.PhotoImage(resized4)

img4Label = Label(login_window,image=signinIm4,bg = "#2B2B2B")
img4Label.place(x = 35,y = 52+5)

my_pic55 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\help_icon.png")
resized55 = my_pic55.resize((25,25),Image.ANTIALIAS)
signinIm55= ImageTk.PhotoImage(resized55)

img55Label = Label(login_window,image=signinIm55,bg = "#2B2B2B")
img55Label.place(x = 35,y = 104+10)

my_pic6 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\about_icon.png")
resized6 = my_pic6.resize((25,25),Image.ANTIALIAS)
signinIm6= ImageTk.PhotoImage(resized6)

img6Label = Label(login_window,image=signinIm6,bg = "#2B2B2B")
img6Label.place(x = 35,y = 152+10)

my_pic7 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\t&c_icon.png")
resized7 = my_pic7.resize((25,25),Image.ANTIALIAS)
signinIm7= ImageTk.PhotoImage(resized7)

img7Label = Label(login_window,image=signinIm7,bg = "#2B2B2B")
img7Label.place(x = 35,y = 202+10)

my_pic8 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\exit_icon.png")
resized8 = my_pic8.resize((25,25),Image.ANTIALIAS)
signinIm8= ImageTk.PhotoImage(resized8)
img8Label = Label(login_window,image=signinIm8,bg = "#2B2B2B")

img8Label.place(x = 35,y = 252+10)
```

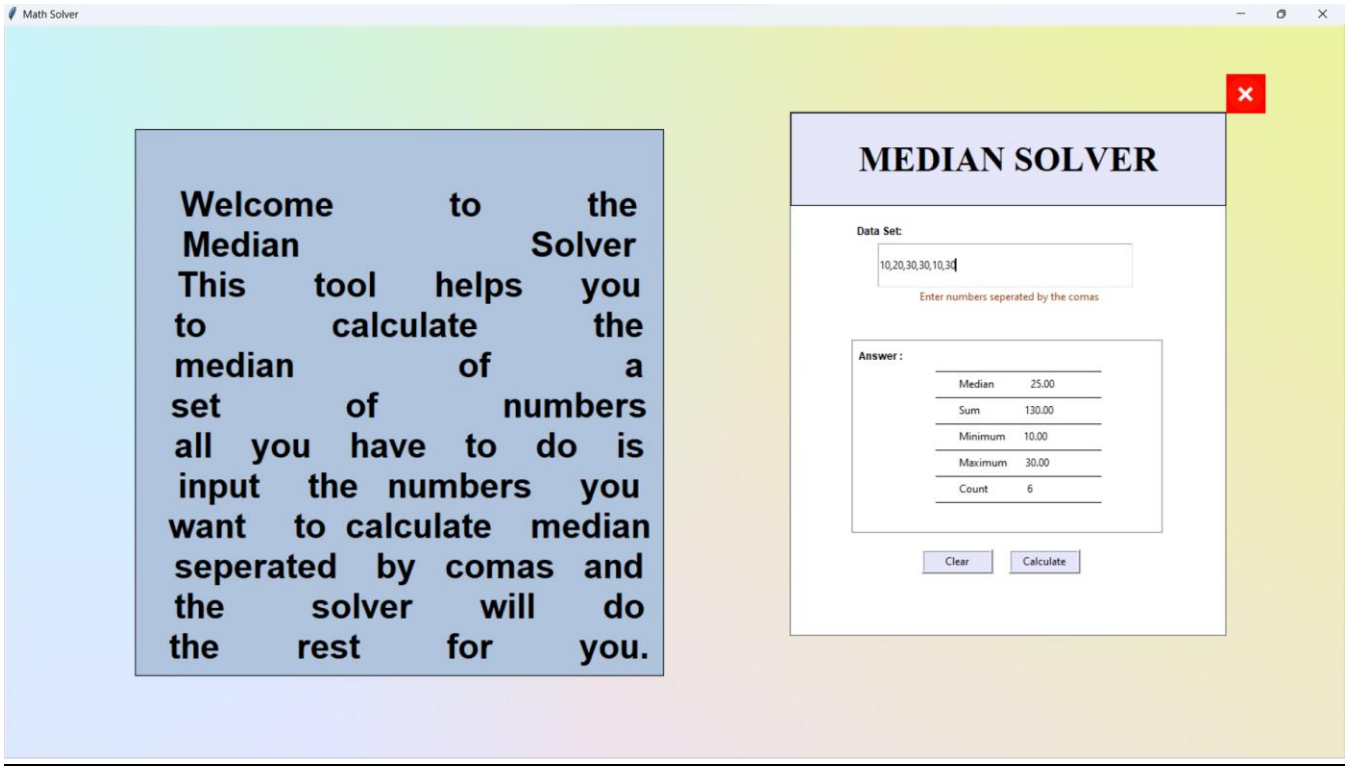
```
my_pic9 =
Image.open("C:\\Users\\prath\\PycharmProjects\\pythonProject2\\logout_icon.png")
resized9 = my_pic9.resize((25,25),Image.ANTIALIAS)


signinIm9= ImageTk.PhotoImage(resized9)

img9Label = Label(login_window,image=signinIm9,bg = "#2B2B2B")
img9Label.place(x = 35,y = 303+10)

login_window.title("Math Solver")

login_window.mainloop()
```

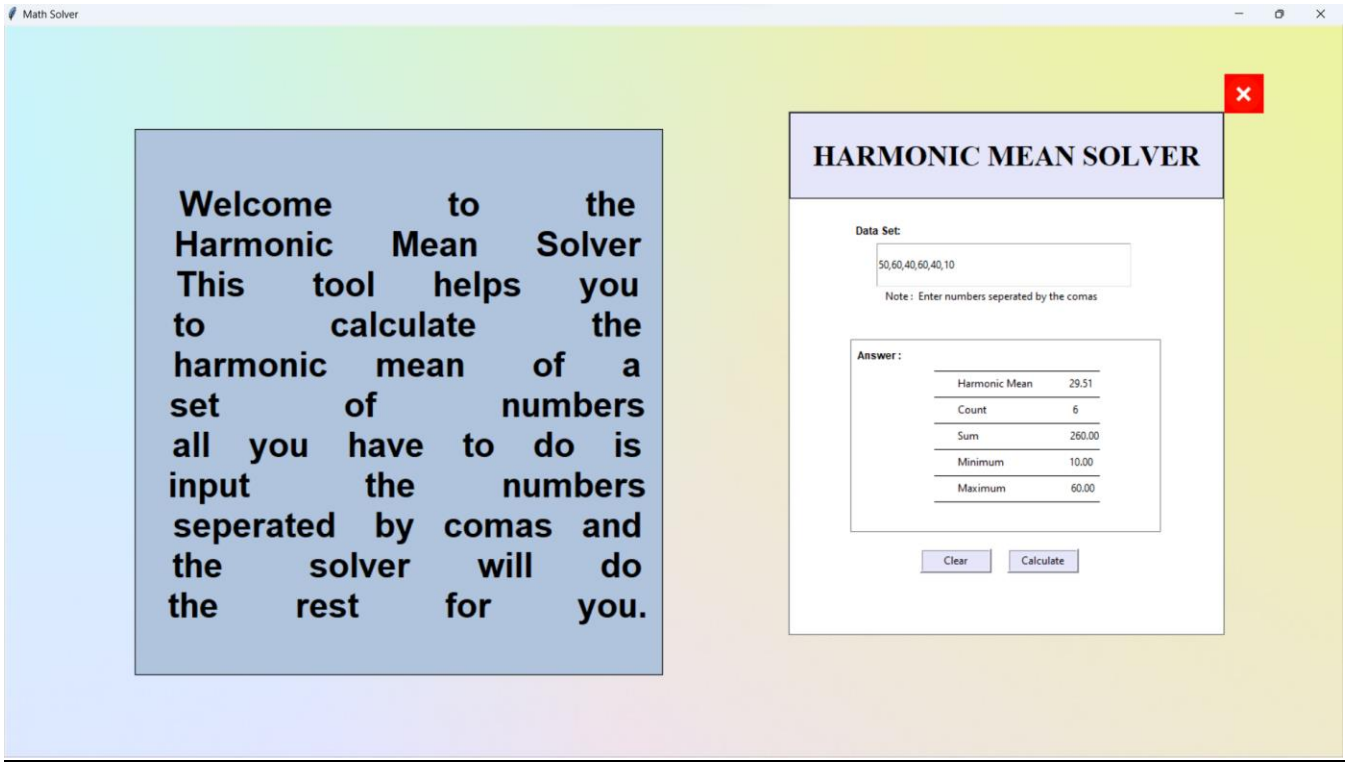
6.3) Input Screen and Output Screen

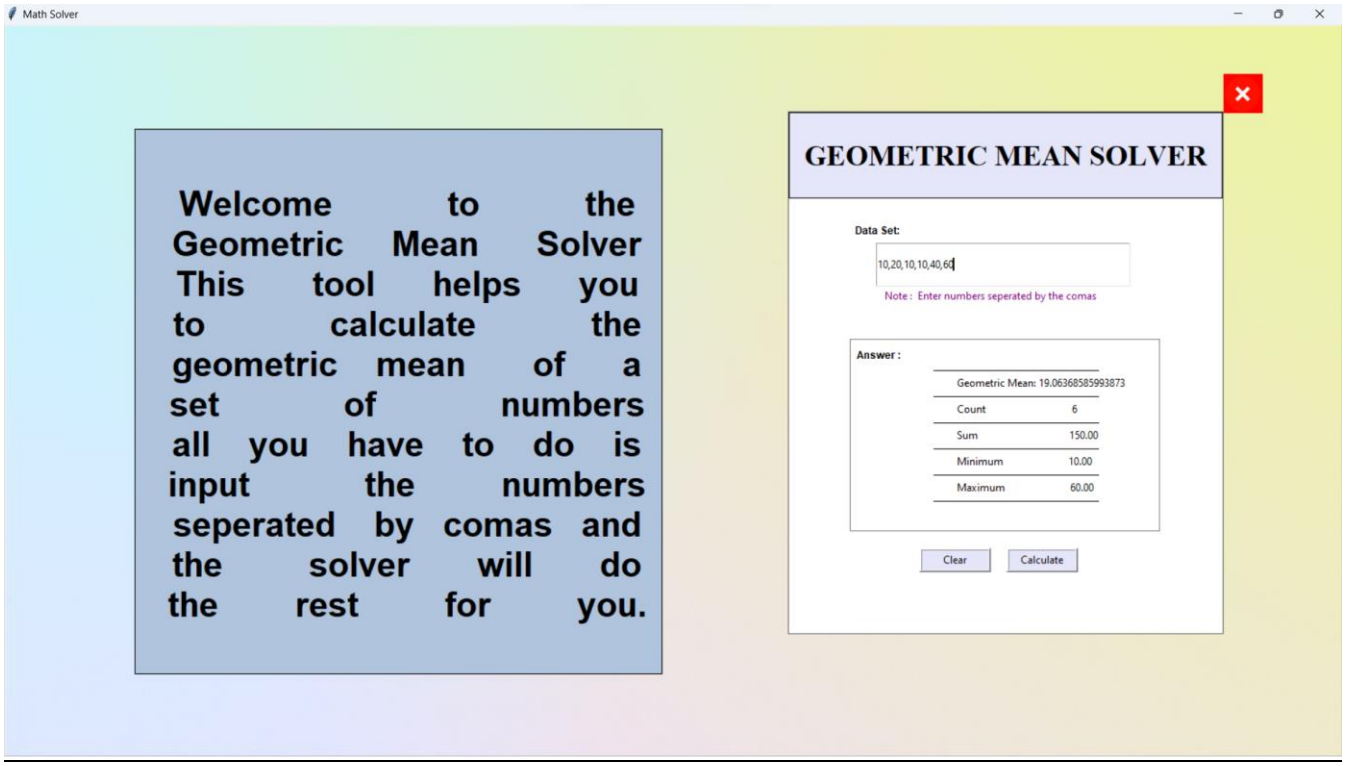


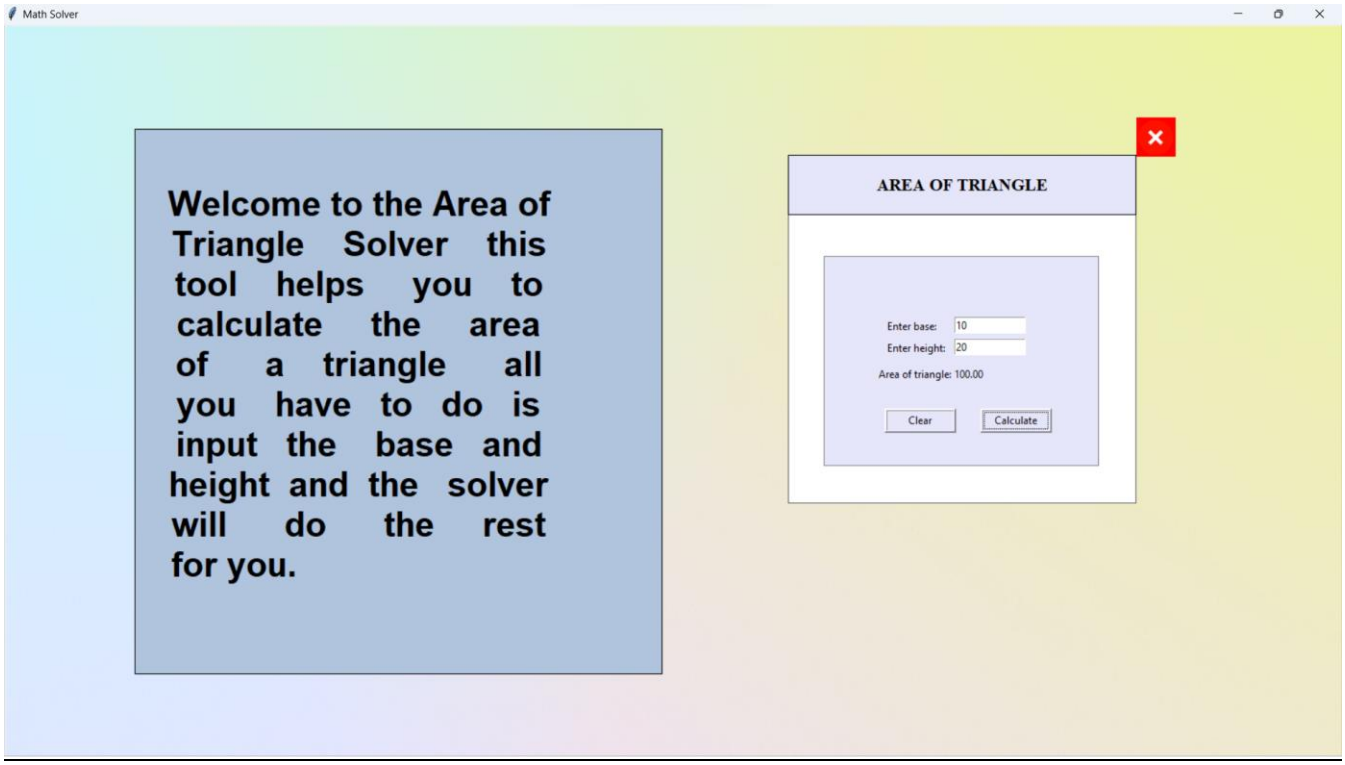


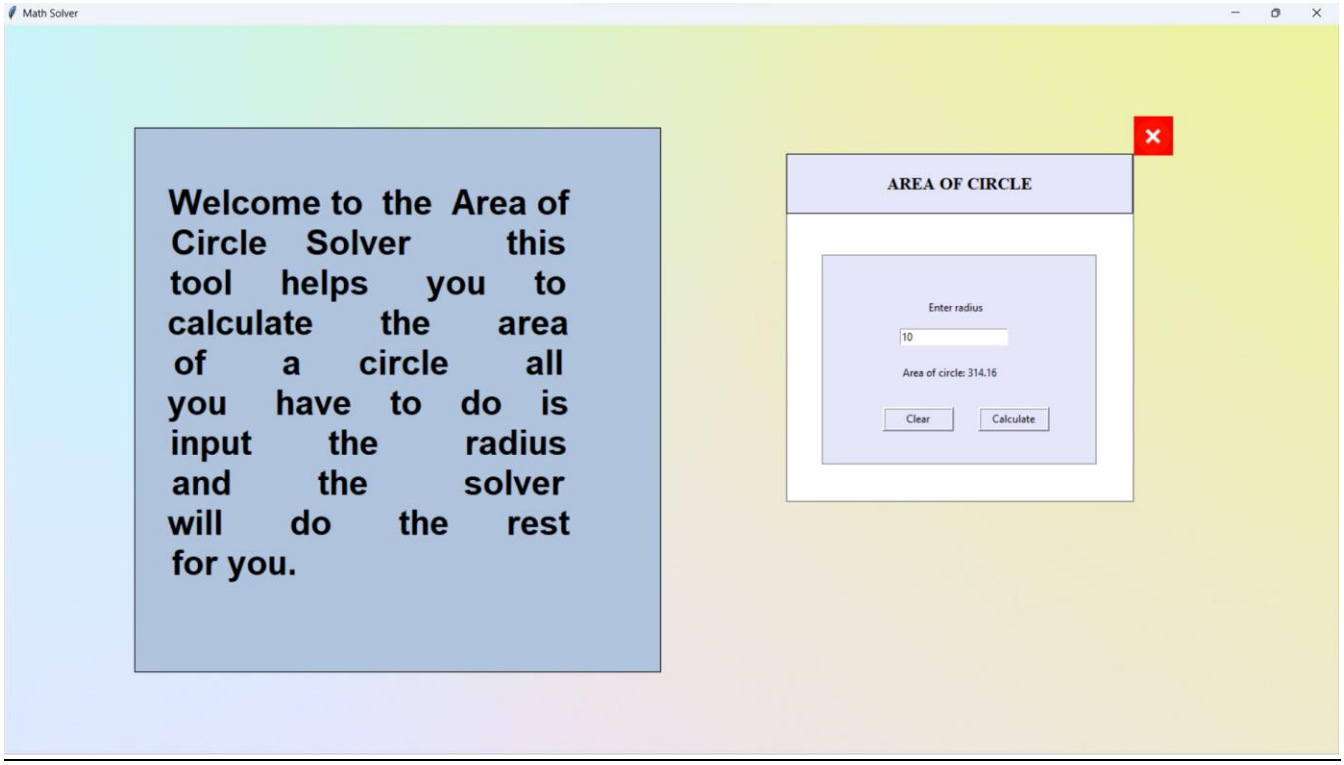

Math Solver
Welcome to the Mode Solver This tool helps you to calculate the mode of a set of numbers all you have to do is input the numbers you want to calculate mode seperated by comas and the solver will do the rest for you.
MODE SOLVER
Data Set:
10,20,30,30,30
Note : Enter numbers seperated by the comas
Answer :
Mode 30.00
Sum 120.00
Minimum 10.00
Maximum 30.00
Count 5
Clear
Calculate

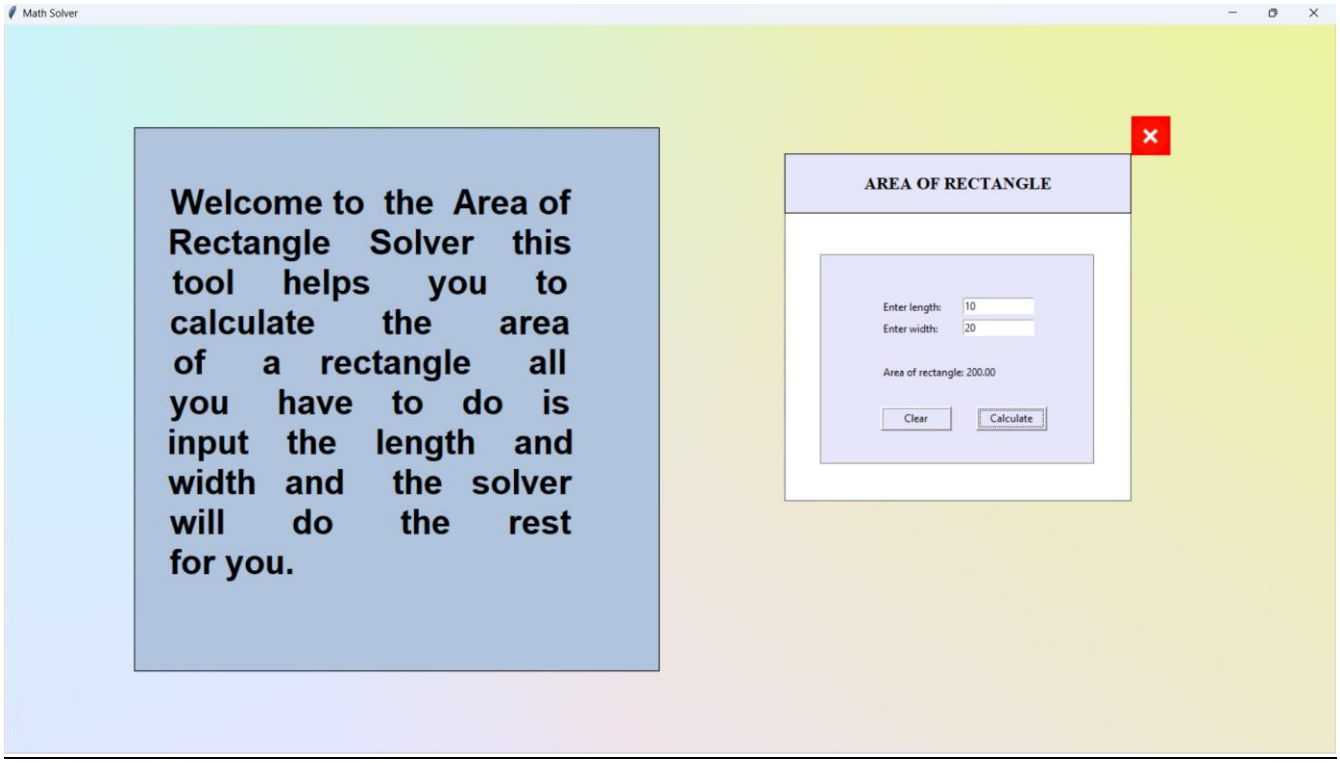

Math Solver
Welcome to the Median Solver This tool helps you to calculate the median of a set of numbers all you have to do is input the numbers you want to calculate median seperated by comas and the solver will do the rest for you.
MEDIAN SOLVER
Data Set:
10,20,30,30,10,30
Enter numbers seperated by the comas
Answer :
Median 25.00
Sum 130.00
Minimum 10.00
Maximum 30.00
Count 6
Clear
Calculate

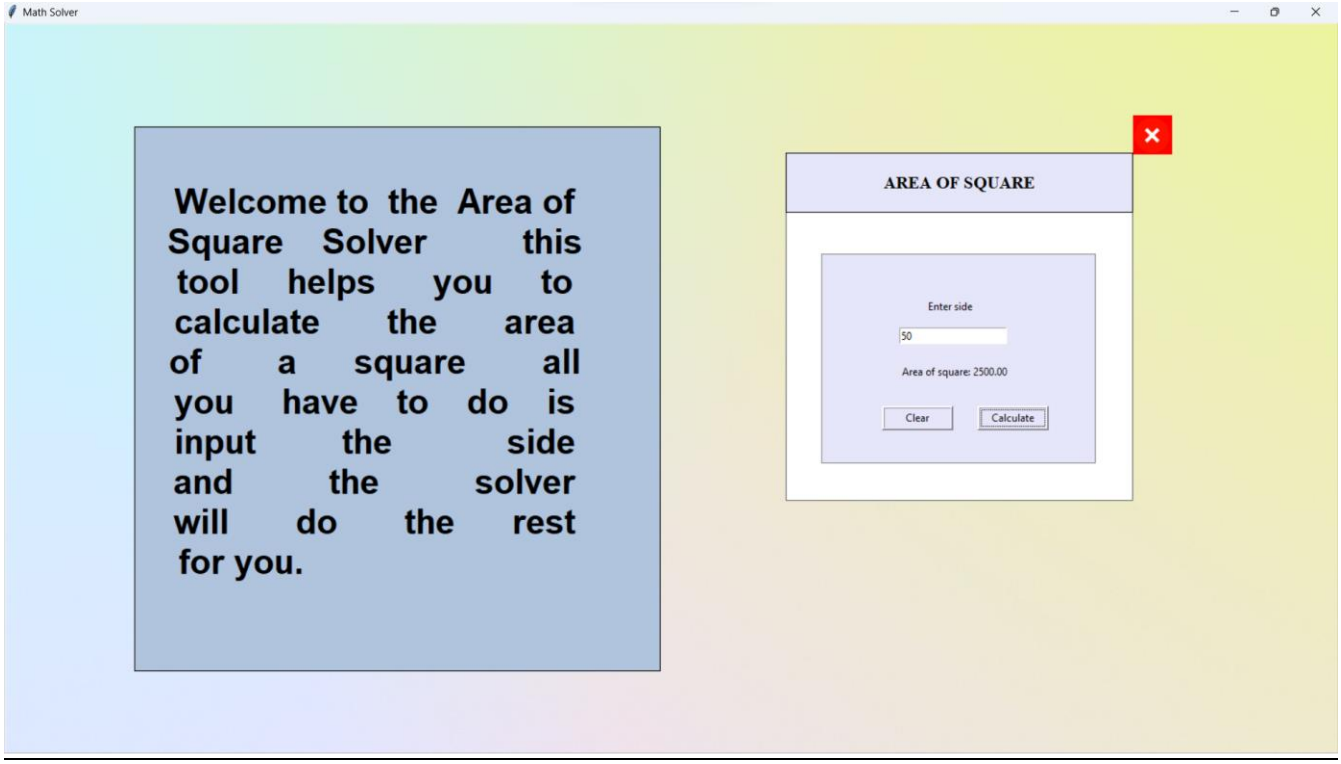

Math Solver
Welcome to the Harmonic Mean Solver This tool helps you to calculate the harmonic mean of a set of numbers all you have to do is input the numbers seperated by comas and the solver will do the rest for you.
HARMONIC MEAN SOLVER
Data Set:
50,60,40,60,40,10
Note : Enter numbers seperated by the comas
Answer :
Harmonic Mean 29.51
Count 6
Sum 260.00
Minimum 10.00
Maximum 60.00
Clear
Calculate

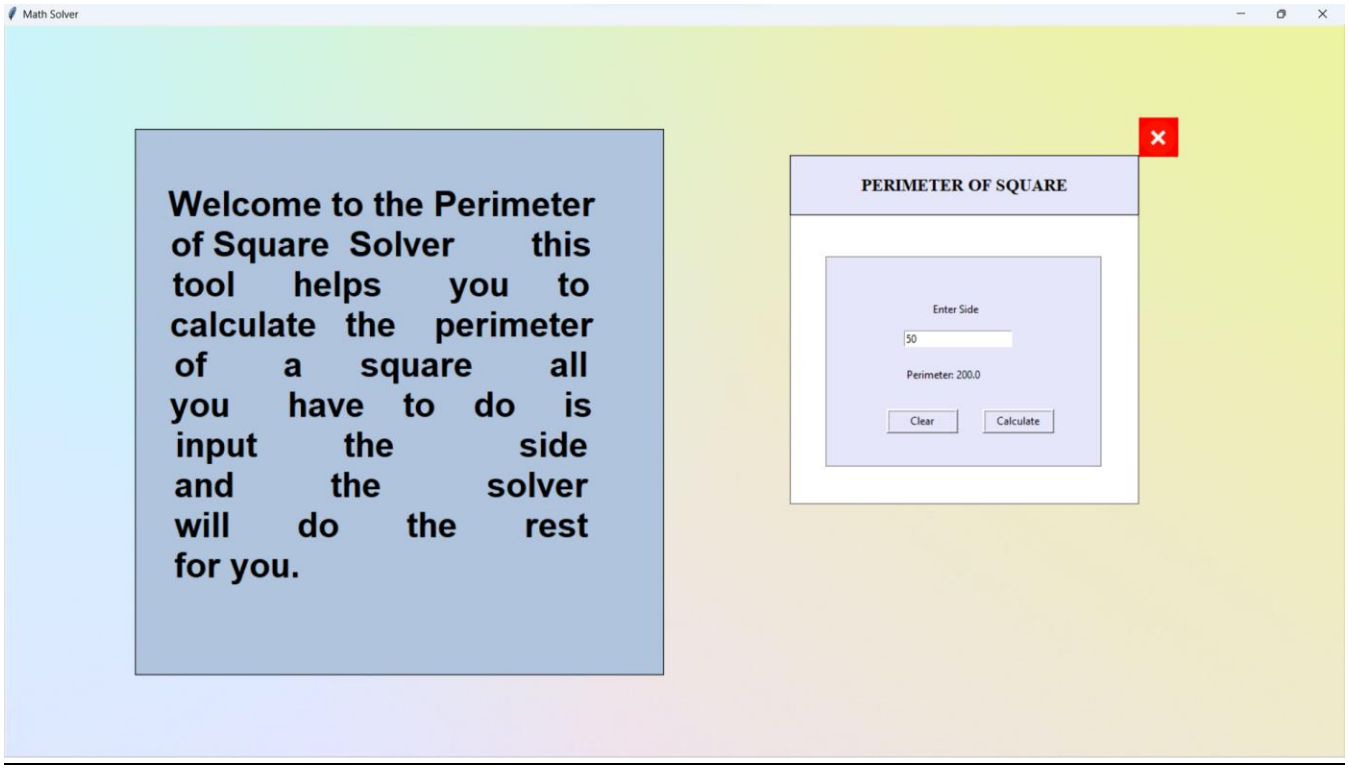

Math Solver
Welcome to the Geometric Mean Solver This tool helps you to calculate the geometric mean of a set of numbers all you have to do is input the numbers seperated by comas and the solver will do the rest for you.
GEOMETRIC MEAN SOLVER
Data Set:
10,20,10,10,40,60
Note : Enter numbers seperated by the comas
Answer :
Geometric Mean: 19.06368585993873
Count 6
Sum 150.00
Minimum 10.00
Maximum 60.00
Clear
Calculate

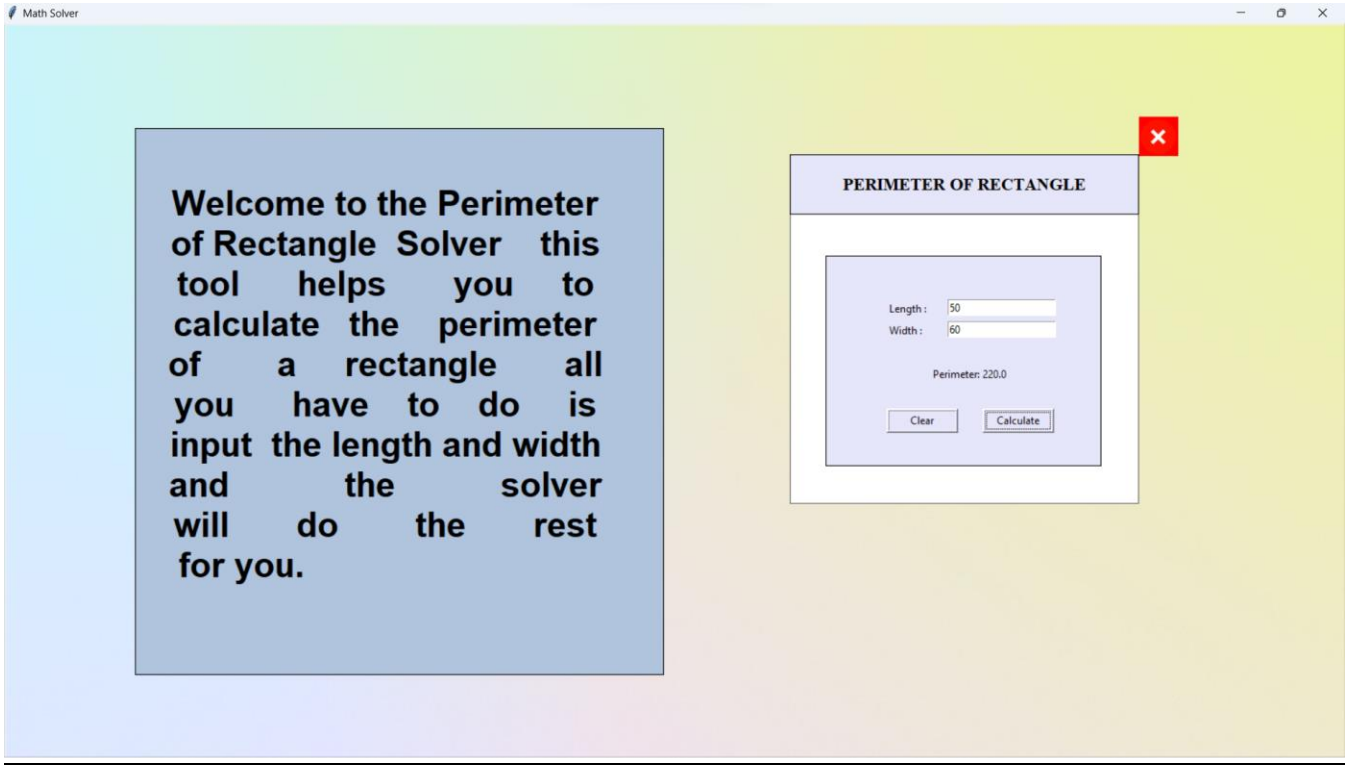

Math Solver
Welcome to the Area of Triangle Solver this tool helps you to calculate the area of a triangle all you have to do is input the base and height and the solver will do the rest for you.
AREA OF TRIANGLE
Enter base: 10
Enter height: 20
Area of triangle: 100.00
Clear
Calculate

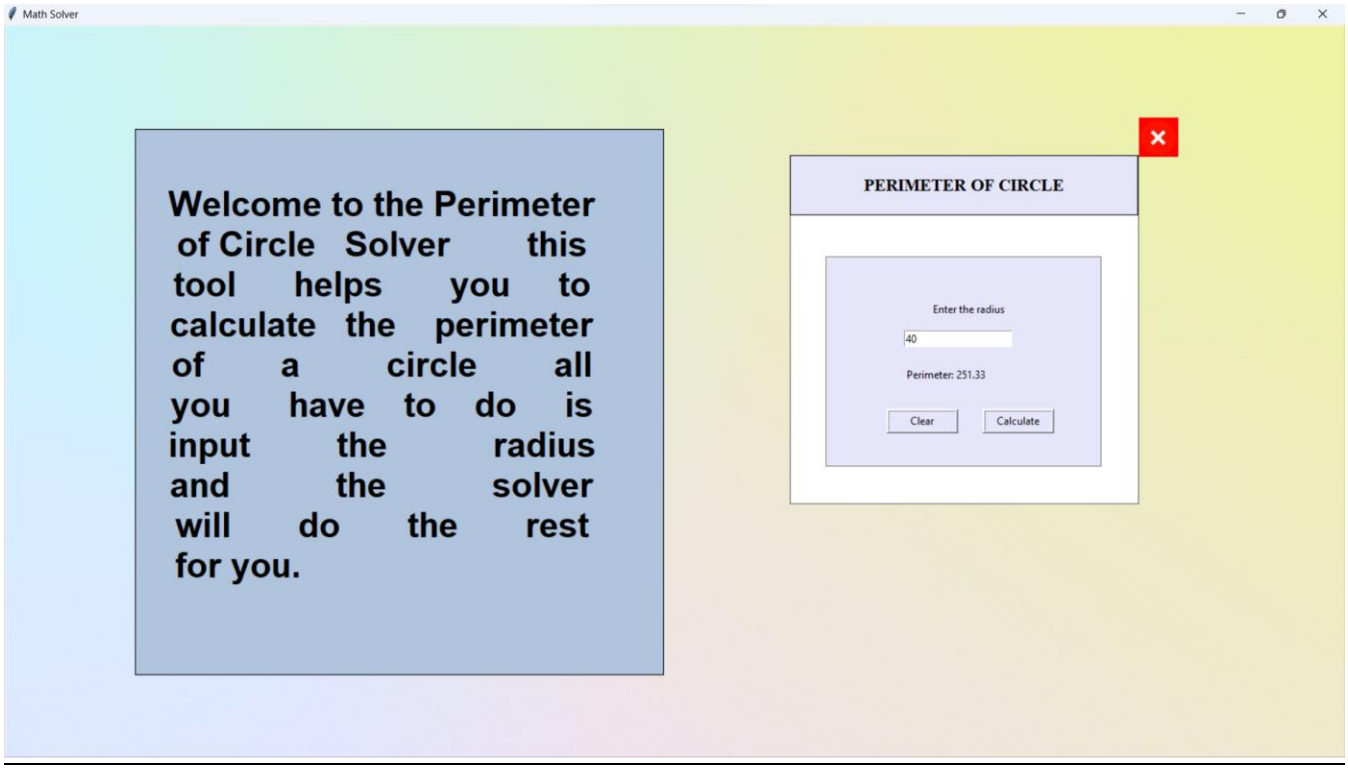

Math Solver
Welcome to the Area of Circle Solver this tool helps you to calculate the area of a circle all you have to do is input the radius and the solver will do the rest for you.
AREA OF CIRCLE
Enter radius
10
Area of circle: 314.16
Clear
Calculate

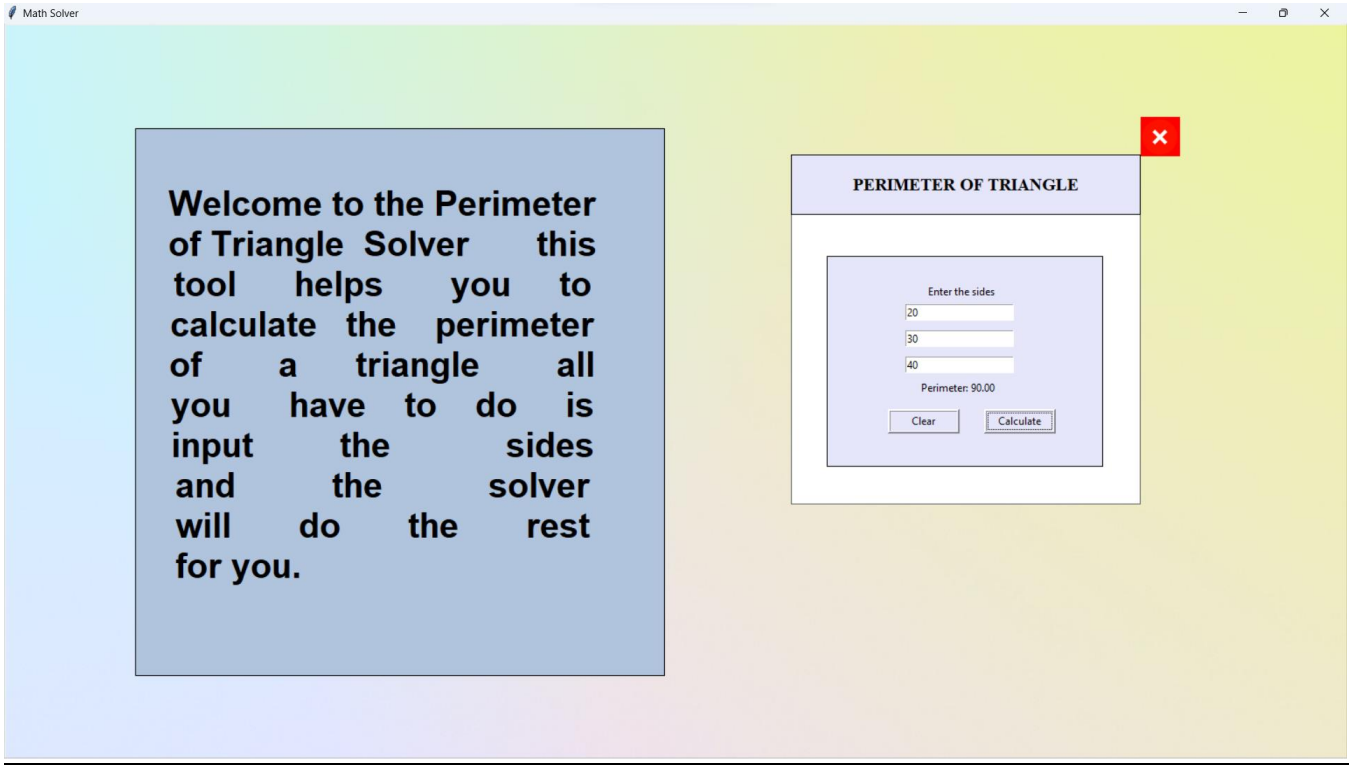

Math Solver
Welcome to the Area of Rectangle Solver this tool helps you to calculate the area of a rectangle all you have to do is input the length and width and the solver will do the rest for you.
AREA OF RECTANGLE
Enter length:
10
Enter width:
20
Area of rectangle: 200.00
Clear
Calculate

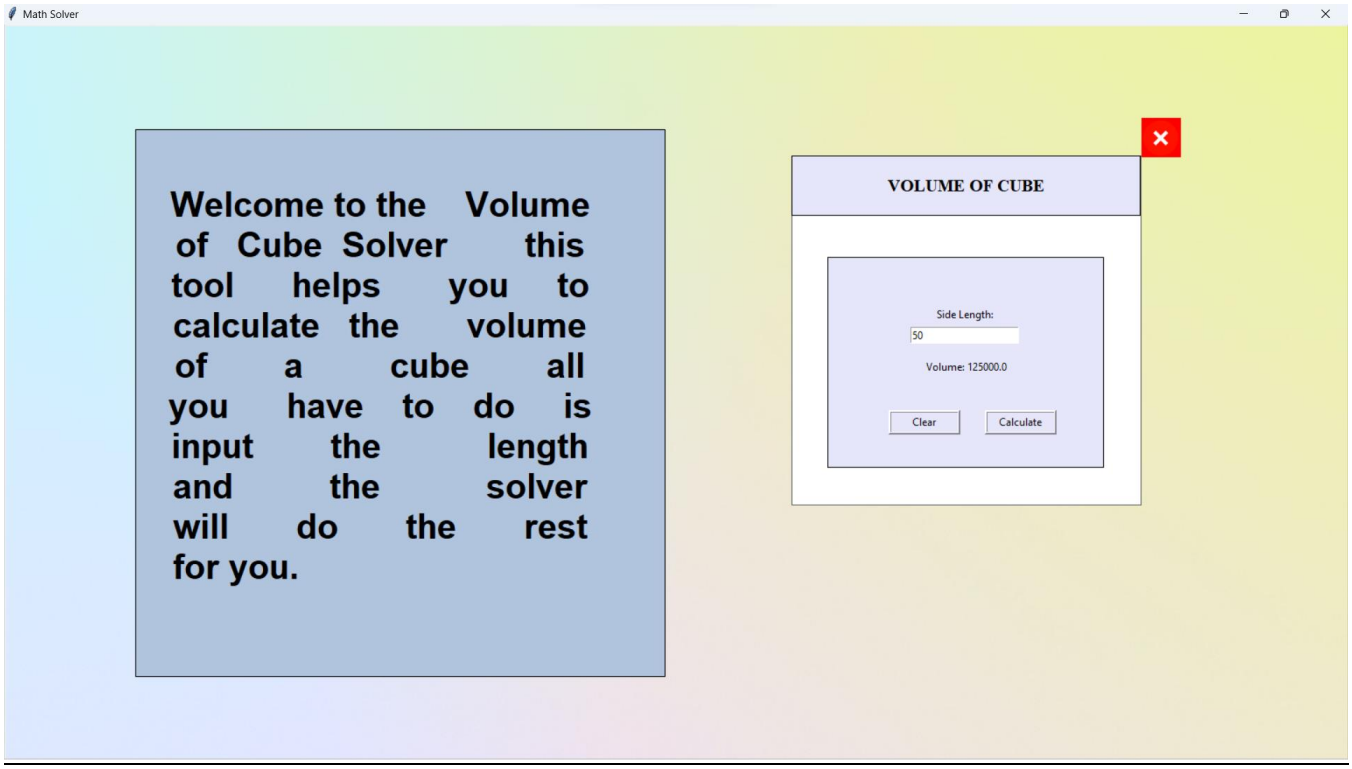

Math Solver
Welcome to the Area of Square Solver this tool helps you to calculate the area of a square all you have to do is input the side and the solver will do the rest for you.
AREA OF SQUARE
Enter side
50
Area of square: 2500.00
Clear
Calculate

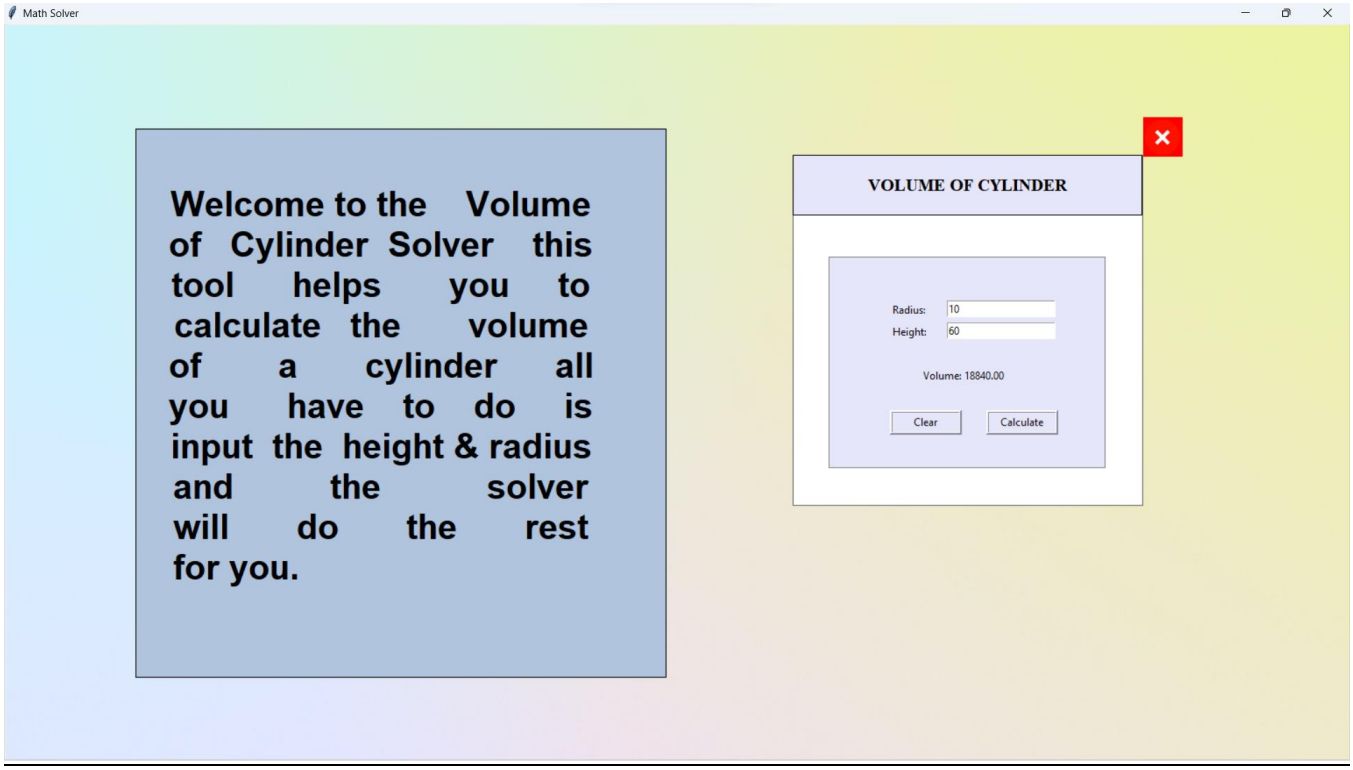

Math Solver
Welcome to the Perimeter of Square Solver this tool helps you to calculate the perimeter of a square all you have to do is input the side and the solver will do the rest for you.
PERIMETER OF SQUARE
Enter Side
50
Perimeter: 200.0
Clear
Calculate

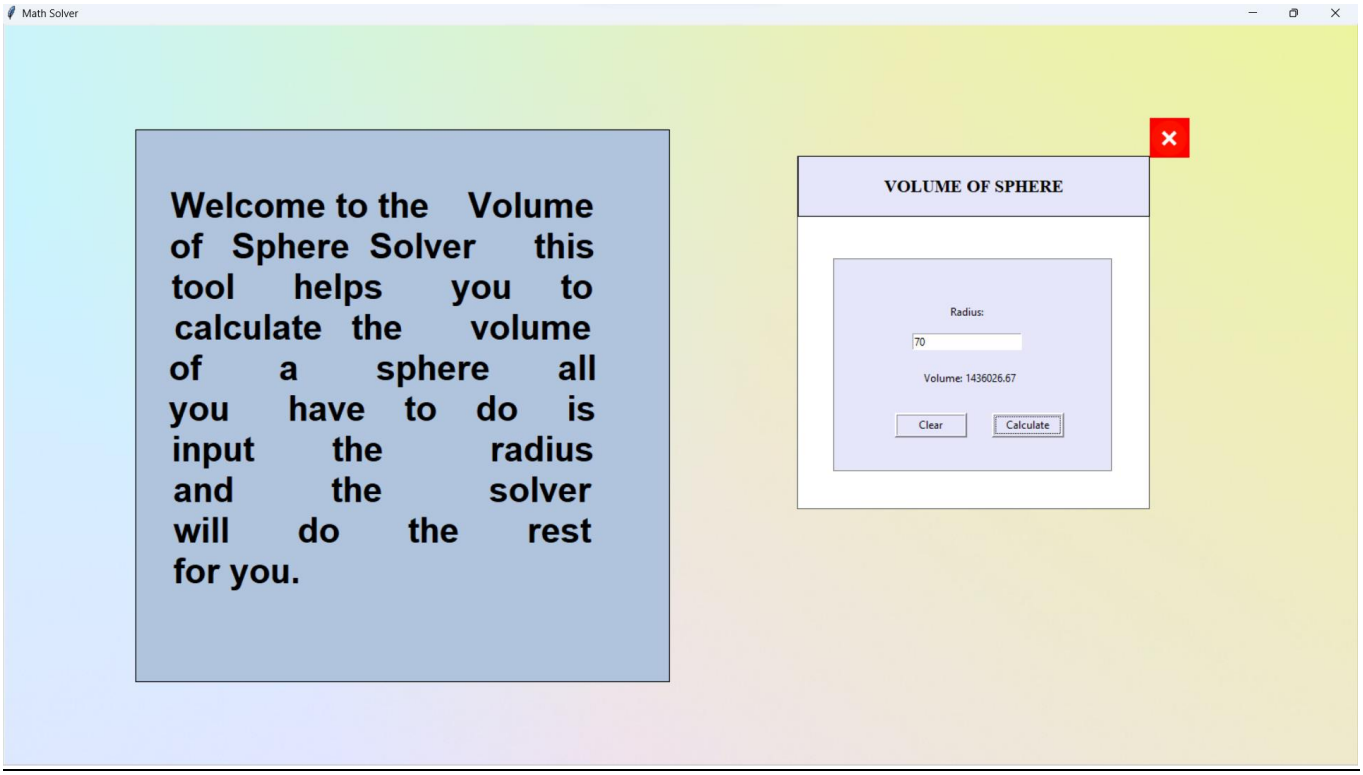

Math Solver
Welcome to the Perimeter of Rectangle Solver this tool helps you to calculate the perimeter of a rectangle all you have to do is input the length and width and the solver will do the rest for you.
PERIMETER OF RECTANGLE
Length :
50
Width :
60
Perimeter: 220.0
Clear
Calculate

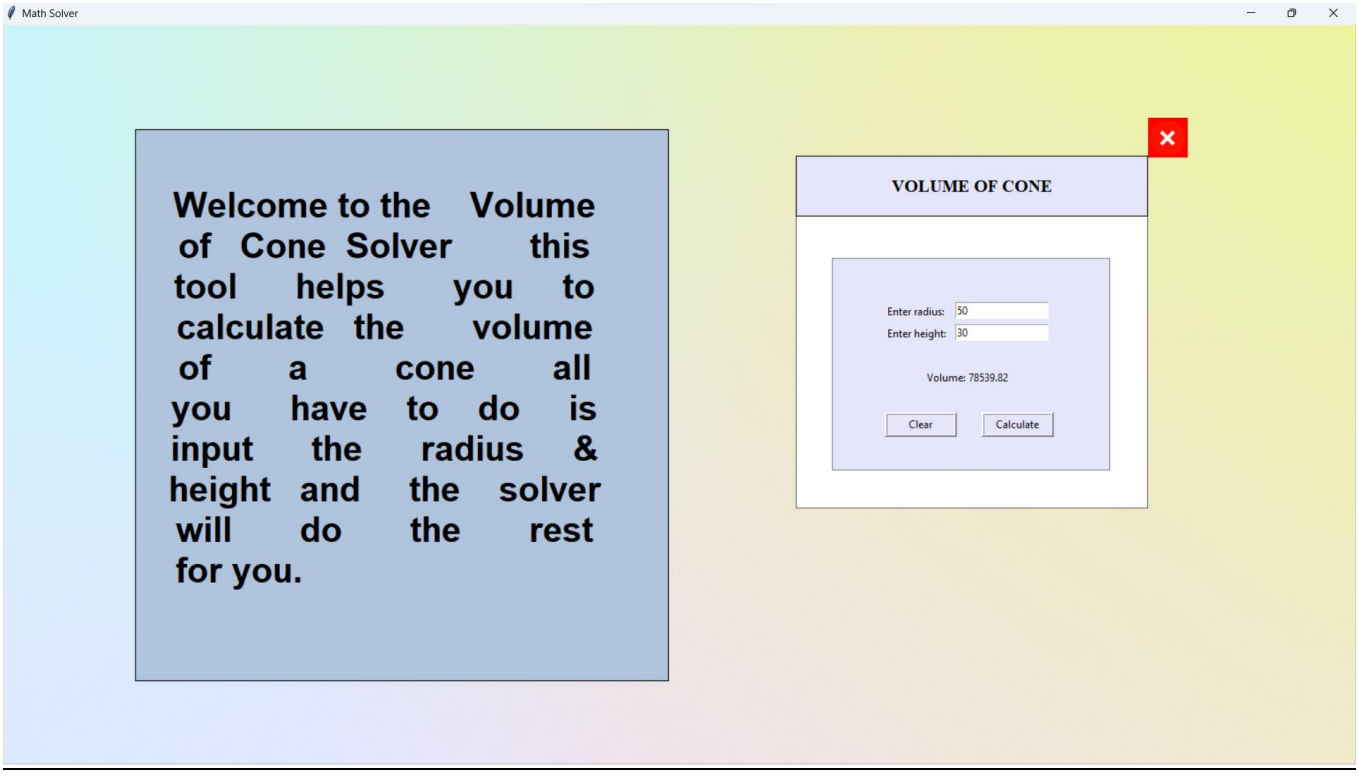

Math Solver
Welcome to the Perimeter of Circle Solver this tool helps you to calculate the perimeter of a circle all you have to do is input the radius and the solver will do the rest for you.
PERIMETER OF CIRCLE
Enter the radius
40
Perimeter: 251.33
Clear
Calculate


Math Solver
Welcome to the Perimeter of Triangle Solver this tool helps you to calculate the perimeter of a triangle all you have to do is input the sides and the solver will do the rest for you.
PERIMETER OF TRIANGLE
Enter the sides
20
30
40
Perimeter: 90.00
Clear
Calculate


Math Solver
Welcome to the Volume of Cube Solver this tool helps you to calculate the volume of a cube all you have to do is input the length and the solver will do the rest for you.
VOLUME OF CUBE
Side Length:
50
Volume: 125000.0
Clear
Calculate


Math Solver
Welcome to the Volume of Cylinder Solver this tool helps you to calculate the volume of a cylinder all you have to do is input the height & radius and the solver will do the rest for you.
VOLUME OF CYLINDER
Radius:
10
Height:
60
Volume: 18840.00
Clear
Calculate


Math Solver
Welcome to the Volume of Sphere Solver this tool helps you to calculate the volume of a sphere all you have to do is input the radius and the solver will do the rest for you.
VOLUME OF SPHERE
Radius:
70
Volume: 1436026.67
Clear
Calculate


Math Solver
Welcome to the Volume of Cone Solver this tool helps you to calculate the volume of a cone all you have to do is input the radius & height and the solver will do the rest for you.
VOLUME OF CONE
Enter radius: 50
Enter height: 30
Volume: 78539.82
Clear
Calculate

Testing and Validation Check

7) Testing and Validation Check

Validation is nothing but the security measures taken at the time of the execution of any program. It is necessary for the analyst to take the validation in their project as it provides more accuracy and systematic flow to project. Validation not only stops input of the false data but also provides the information in the form of message to the user clearly warn the user to input correct data type. Hence it plays an important role of a guide during input of data.

Validation put it controls over the data in both character as well as integer data type. Whenever wrong data or invalid data is stored by the user it frees the message immediately.

Validation input transaction:

Validation input data is largely done through software which is the programmer's responsibility but it is important that system analyst must know that common program might invalidate a transaction. Business committed to quality will include validation checks as a part of their routine software.

System Security Measures

Strong Passwords:

Enforce the use of strong, unique passwords for user accounts, with minimum length, complexity, and expiration requirements.

Limited User Privileges:

Limit user privileges to only what is necessary for their tasks, minimizing the risk of unauthorized access or misuse.

Secure Database Connectivity:

Use secure methods to connect to the database, such as encrypted connections, to protect data in transit.

Input Validation:

Validate and sanitize all user input to prevent common attacks such as SQL injection or cross-site scripting.

Error Handling:

Implement proper error handling to prevent leakage of sensitive information or revealing implementation details.

Regular Updates:

Keep your application and all dependencies up-to-date with the latest security patches and updates to address known vulnerabilities.

Implementation, Evolution and Maintenance

9) 9.1) Implementation:

Implementation phase is mainly concerned with the user training, site, and preparation and file conversion. It also involves final testing of the system.

During implementation the component built during development are put into optional use.

Following are the points should be considered while doing implementation of the application:

- Testing, debugging and documentation program.
- Converting data from old to new system.
- Giving training to the user about how to operate the system.
- Developing operating procedures for the computer operating staff.
- Establishing a maintenance procedure to repair and enhance system.
- Completing Documentation
- Operating system on the user location and solving all the issues
- occurred while operation.

9.2) Evaluation:

After the implementation phase, another stage in project development is evaluation. After keeping the project in the working condition for some time, all the errors that are shown in the computer program should be removed. The programmer needs to correct them so that same errors should not be repeated. We should also get the feedback from the user which are using it and ask them whether, it is user friendlyor not. After evaluating the program and satisfying the needs of the user the program is maintained

fully to give the same functionalities for what it was intendedto be. This stage should be implemented so as to regular check-up the errors with error handling techniques. This stage is the updating and correcting of the program to account for changing condition or field experience. Proper testing and Documentation significantly reduce the frequency and extent of the required maintenance.

9.3) Maintenance:

Maintenance is very crucial for success of any application; proper maintenance of the application makes it smooth working application. Maintenance is done basically, for two reasons i.e., to correct software errors which occurs after the testing and implementation of the application when one user it and other reason is to enhance the software capabilities in response to changing organizational needs. User often requires additional feature after he/she uses the application and becomes familiar with it. Some of the large companies gives AMC (Annual Maintenance Contract) to other companies for regular maintenance of the software/application. The cost of the maintenance increases the cost of the application/software. At a point of time, it becomes feasible to perform the tasks related to the maintenance of the software. Maintenance phase always occurs after the implementation of the application is done. It corrects all the previously undetected errors of the application and helps to do update in the application which is required by the user. Maintenance is one of thestages in the SDLC (System Development Life Cycle). It is basically done for the estimation, controlling, and making modification to the implemented system.

Future Scope of the Project

Future scope of Project:

- **Speech to Text and vice versa:** The users will be given the option of conversion of their own speech directly to text as it allows easy functioning and saves time for the users. This also allows people with disabilities to either listen to what they've input or easily input their own data without having the need to type.

- **Expand the math functions:** You can add more mathematical functions to the program, such as trigonometric functions, logarithmic functions, and exponential functions.

- **History log:** Add a history log to the program to keep track of the user's previous calculations, making it easier for them to review and reuse previous calculations.

- **Graphical representation of results:** Incorporate graphing capabilities into the program to provide a visual representation of the mathematical calculations.

- **Customization of interface:** Allow users to customize the interface to their preferences, such as font size, color scheme, and layout.

- **Support for different languages:** Add support for different languages, making the program accessible to a wider range of users.

- **Scientific notation:** Add support for scientific notation, which would be particularly useful when dealing with large or small numbers.

- **Integration with other software:** Integrate the program with other software such as spreadsheets or plotting software, allowing for more advanced data analysis.

- **Error messages improvement**: Improve error messages to provide more detailed information on the type of error and how to fix it.

- **Connection to internet shall allow access of data everywhere:** Connectivity of internet shall allow access to a user to access their key or their saved file on cloud from anywhere they want globally only if they have the app installed as making it into a website shall increase load to save it on the cloud and also make it susceptible to online threats.

Conclusion

Conclusion:

The Math Solver project created using Python Tkinter has proven to be a useful tool for anyone who needs to quickly solve mathematical problems. With its user-friendly interface and simple input methods, users can easily calculate the Pythagoras theorem, mean, mode, median, and many other mathematical functions.

The project utilizes the powerful capabilities of the Tkinter library to provide a smooth and responsive experience to the user. It is an excellent example of how Python can be used to create practical applications that are both efficient and user-friendly.

Overall, the Math Solver project provides a valuable resource to students, teachers, and professionals who need to solve mathematical problems quickly and accurately. With its versatility and ease of use, this application is sure to be a valuable addition to anyone's_toolkit.

Bibliography and References

12) **Bibliography and References** :

Bibliography:

Prathmesh Pradeep Gulhane. (2023).

References:

- Tkinter. https://docs.python.org/3/library/tkinter.html
- Pythagoras Theorem. (2021). Math is Fun. Retrieved from https://www.mathsisfun.com/pythagoras.html
- Mean, Mode, and Median. (2021). Khan Academy.
- Python In Wikipedia.
- Mathematics. In Wikipedia. https://en.wikipedia.org/wiki/Mathematics
- GeeksforGeeks. (2021). Python | Tkinter Widgets | GeeksforGeeks. https://www.geeksforgeeks.org/python-tkinter-widgets/
- Stack Overflow
- Youtube.

By utilizing these references, I was able to create a well-designed and functional application that can solve a wide range of mathematical problems. These references provided me a solid foundation of knowledgeand resources to create a quality software product.

A
PROJECT SYNOPSIS
ON

"Math Solver"

Submitted to

G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR
AUTONOMOUS
In the Partial Fulfillment of

B.Com. (Computer Application) Final Year

Synopsis Submitted by
Prathmesh Gulhane

Under the Guidance of

Pravin J. Yadao

Shiksha Mandal's
G. S. COLLEGE OF COMMERCE & ECONOMICS, NAGPUR
AUTONOMOUS
2022-2023

1. **Introduction: (Write 4 to 5 lines)**

As we know the time is very important factor in students life . Math Solver is the software which takes the input from the user to solve the mathematical and statistical problems. It takes the input from the user and gives the solution of that input to the user. It gives the answer accurately without any mistake and saves the time of the user to solve the lengthy problem.

2. Objectives of the project: (Write only 5 points)

1. Time saving of the user
2. To provide user friendly environment
3. Flexible to use
4. Accurate solutions
5. Speedy working

3. Project Category: Window Application

4. Tools/ Platform/ Languages to be used: Python

5. Scope of future application: (Write 4 to 5 points)

1. Time saver of the user
2. Accurate results provider
3. Speedy results provider
4. Busy schedule of people will lead to more users

Submitted by,

Prathmesh Pradeep Gulhane

Approved by,

Prof. Pravin Yadao
Project Guide