**A MINI-PROJECT**

**REPORT**

**On**

**“SCIENTIFIC CALCULATOR APPLICATION”**

Submitted in partial fulfillment towards the award of degree in

B.TECH in Computer Science and Engineering

SESSION 2023-24



ODD SEMESTER

**NITRA TECHNICAL CAMPUS, GHAZIABAD**

**(College Code-802)**

**Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

&

CSE (AI & ML)

|  |  |
| --- | --- |
|  | **SUBMITTED BY:**  **Alok Kumar**  **ROLL NO:2208020109004**  **CSE/3rd Year** |

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S No.** | **CONTENT** | **PAGE** |
| 1. | ACKNOWLEDGEMENT | V |
| 2. | DECLARATION | VI |
| 3. | ABSTRACT | VII |
| 4. | LIST OF TABLES | VIII |
| 5. | LIST OF FIGURES | X |
| 4.  5.  6.  7.  8.  9.  10.  11.  12. | CHAPTER’S   1. Introduction 2. Problem Formulation 3. Proposed Solution / Methodology 4. System requirements 5. Technology used 6. Code Organization and Structure 7. Results 8. Conclusion | 1  3  5  6  7  10  15  17 |

**ACKNOWLEDGEMENT**

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior, and acts during the course of study.

I am also thankful to my friends those give me support to helpful to complete this project and solve my all difficulties.

I am taking this opportunity to express out gratefulness to the Management, Teachers and staff.

**(Sign of Student)**

|  |  |
| --- | --- |
|  | **Alok Kumar**  **ROLL NO:2208020109004**  **CSE/2nd Year** |

**DECLARATION**

I **ALOK KUMAR (ROLL NO.2208020109004)** hereby declare that the report of **Mini-Project** titled “**Scientific Calculator** ” is uniquely prepared by me and does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

I also confirm that the report is only prepared for my academic requirement, not for any other purpose. It might not be used with the interest of the opposite party of the any organization.

|  |  |
| --- | --- |
|  | **Alok Kumar**  **ROLL NO:2208020109004**  **CSE/2nd Year** |

**ABSTRACT**

The Scientific Calculator Application is a comprehensive and user-friendly calculator designed to perform both basic arithmetic operations and advanced mathematical functions. Developed using **Python** and **Tkinter**, the application offers a seamless and visually appealing interface for users to perform a wide range of calculations.

A scientific calculator is a type of electronic calculator, usually but not always handheld, designed to calculate problems in science, engineering, and mathematics. They have completely replaced slide rules in traditional applications, and are widely used in both education and professional. The **python** calculator was implemented using **tkinter** to make the calculation of mathematical functions easier. The application consists of scientific and standard functions. The standard is used to solve scientific notation type math functions like sin, cos, tan, log etc.

**LIST OF TABLES**

**1.Button layout Table:**

* + Describes the layout of buttons on the calculator, showing the arrangement of digits, arithmetic operations, and scientific functions.

| **Button** | **Description** |
| --- | --- |
| 1-9 | Digits 1-9 |
| 0 | Digit 0 |
| . | Decimal Point |
| = | Equals |
| + | Addition |
| - | Subtraction |
| x | Multiplication |
| / | Division |
| C | Clear |
| AC | All Clear |
| √ | Square Root |
| ^ | Exponentiation |
| x! | Factorial |
| toRad | Convert to Radians |
| toDeg | Convert to Degrees |
| sinθ | Sine Function |
| cosθ | Cosine Function |
| tanθ | Tangent Function |

**2.Functionality Table:**

* + Describes the functionality associated with each button press or action.

| **Button** | **Functionality** |
| --- | --- |
| 1-9 | Append respective digit to the input |
| 0 | Append '0' to the current input |
| . | Append '.' to the current input |
| = | Evaluate the expression and display the result |
| + | Append '+' to the current input |
| - | Append '-' to the current input |
| x | Append '\*' to the current input |
| / | Append '/' to the current input |
| C | Clear the last character in the input |
| AC | Clear all input |
| √ | Calculate the square root of the current input |
| ^ | Calculate the power/exponentiation |
| x! | Calculate the factorial |
| toRad | Convert the value to radians |
| toDeg | Convert the value to degrees |
| sinθ | Calculate the sine of the value in degrees |
| cosθ | Calculate the cosine of the value in degrees |
| tanθ | Calculate the tangent of the value in degrees |

**3.Error Handling Table:**

Describes how errors are handled, such as displaying an error message for invalid input.

| **Error Type** | **Description** |
| --- | --- |
| Evaluation | Display "Error" if the expression is invalid |

This is a general breakdown of the functionality and layout of the calculator. You may customize it further based on your needs or add additional features.

**LIST OF FIGURES**

In the context of a Tkinter-based calculator code, figures usually refer to graphical elements such as buttons, labels, and frames. Here's a list of figures, or graphical components, that are part of your calculator GUI:

1. **Calculator Image:**



- The image displayed at the top of the calculator GUI, specified by the `headinglabel`.

2. **Heading Label:**- The label displaying the text "My Calculator."



**3. Textfield:**

- The entry widget used for input and display of calculations.



**4. Numeric Buttons (0-9):**

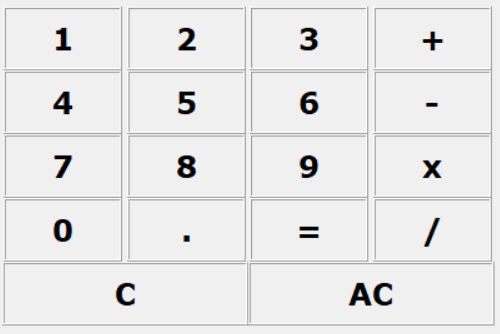
- Buttons representing the digits from 0 to 9.

**5. Decimal Point Button (.):**

- Button for the decimal point.

**6. Arithmetic Operation Buttons (+, -, x, /):**

- Buttons for addition, subtraction, multiplication, and division.



**7. Equals Button (=):**

- Button for performing the calculation.

**8. Clear Button (C):**

- Button to clear the last character entered.

**9.All Clear Button (AC):**

- Button to clear all input.

**10. Scientific Calculator Buttons:**

- Buttons for additional scientific functions, such as square root (√), exponentiation (^), factorial (x!), conversion between radians and degrees, and trigonometric functions (sin, cos, tan).

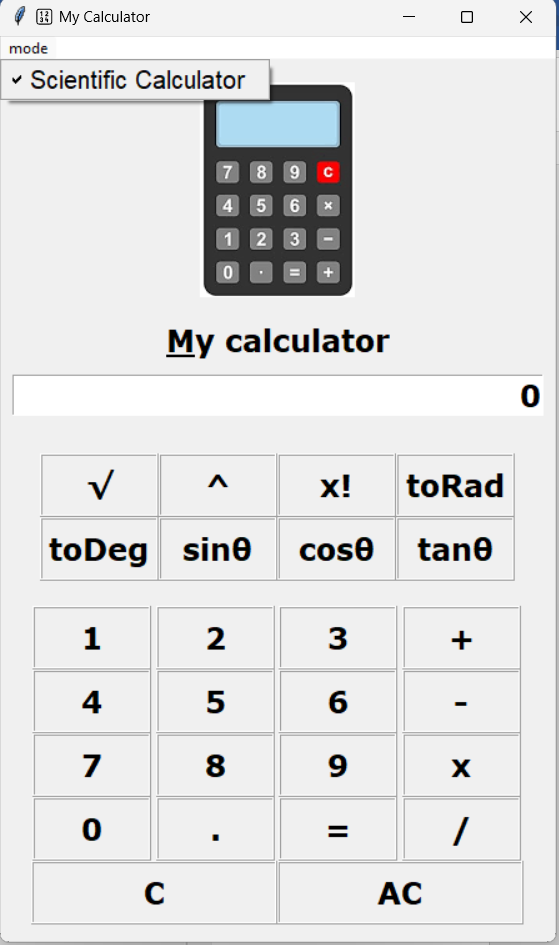
**11. Mode Menu:**

- Menu option for switching between standard and scientific calculator modes.

**12. Frames:**

- Frames used to organize and group different sections of the calculator, such as `buttonframe` for standard buttons and `scframe` for scientific buttons.

These are the main graphical elements/components in your calculator GUI. If you have used any external images or icons, you might want to include those in your list as well.



**C****hapter:1 Introduction**

**Python** offers multiple options for developing a GUI (Graphical User Interface). Out of all the GUI methods, **Tkinter** is the most commonly used method. It is a standard **Python** interface to the Tk GUI toolkit shipped with **Python**. **Python** with **Tkinter** outputs the fastest and easiest way to create GUI applications. Creating a GUI using **Tkinter** is an easy task.

To create a **Tkinter**:

1. Importing the module – **tkinter**
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets

**1.1 Background:**

The project aims to develop a user-friendly Scientific Calculator using the **Tkinter** library in **Python**. Calculators are essential tools for performing various mathematical operations, and this project enhances the basic calculator functionalities by incorporating advanced scientific features.

**1.2 Motivation:**

The motivation behind this project is to create a versatile calculator that caters to both basic and scientific calculation needs. The user interface is designed to be intuitive, making it accessible to a wide range of users.

**scientific calculator** implemented in **Python** using the **Tkinter** library for the graphical user interface. The calculator offers both standard and scientific functionality, allowing users to perform a variety of mathematical operations.

**Features of the Calculator:**

1. **Basic Arithmetic Operations:**

Addition, subtraction, multiplication, and division.

Clear and All Clear buttons for convenient input management.

**2.Scientific Calculator Functions:**

Square root (√), exponentiation (^), and factorial (x!) operations.

Conversion between degrees and radians (toDeg, toRad).

Trigonometric functions: sine (sinθ), cosine (cosθ), and tangent (tanθ).

**3.User-Friendly Interface:**

Clear and organized layout with numeric buttons, operation buttons, and a text field to display input and results.

Integration of an image at the top for aesthetic appeal.

**4.Error Handling:**

The calculator incorporates error handling to manage exceptions during expression evaluation, providing users with informative error messages.

**5.Switchable Modes:**

Users can switch between a standard calculator mode and a scientific calculator mode, enhancing flexibility.

Identified Issues and Proposed Improvements:

The code review revealed several areas for enhancement, including improvements in the user interface, error handling, code organization, and adherence to best coding practices. The proposed solutions aim to address these issues and elevate the overall quality of the calculator program.

**Objectives:**

**Improve User Experience:**

Enhance the graphical layout for improved aesthetics.

Provide more informative error messages for a better user understanding.

**Code Quality and Readability:**

Organize the code into functions or classes to enhance readability and maintainability.

Adopt consistent coding styles, avoid magic numbers, and use meaningful variable names.

**Flexibility and Safety:**

Replace the use of eval for expression evaluation with safer alternatives.

Implement constant values for window dimensions and avoid hard-coded paths.

**Documentation:**

Add comments to clarify complex sections of code.

Provide docstrings for functions to document their purpose and usage.

Through the proposed solutions and methodologies, the objective is to transform the existing calculator into a more user-friendly, robust, and maintainable application. The ensuing sections of the document will delve into each identified issue and outline the steps taken to implement the proposed solutions.

**Chapter:2 Problem Formulation**

The problem formulation for this code involves identifying and describing the issues, challenges, or improvements that could be addressed in the existing scientific calculator program. Here are some points to consider:

1. **User Interface Improvements:**
   * The calculator interface could be enhanced for better user experience.
   * Consider adding labels to indicate the current mode (normal or scientific).
   * Improve the layout and alignment of buttons for a cleaner look.
2. **Error Handling:**
   * The code currently catches exceptions during evaluation and shows an error message box. Consider providing more specific error messages or handling different types of errors more gracefully.
3. **Code Structure:**
   * The code could be organized into functions and classes for better readability and maintainability.
   * Consider separating the GUI setup, normal calculator logic, and scientific calculator logic into different functions or classes.
4. **Magic Numbers:**
   * Replace magic numbers in the code with named constants or variables for better code readability. For example, the window dimensions (445x570, 445x705) could be defined as constants.
5. **Comments and Documentation:**
   * Add comments to explain complex sections of code, especially in the scientific calculator logic.
   * Consider providing docstrings for functions to document their purpose and usage.
6. **Variable Naming:**
   * Choose more descriptive variable names to improve code readability.
   * Avoid using single-letter variable names unless they are conventional (e.g., **e** for an exception).
7. **Redundant Code:**
   * There are redundant print statements used for debugging. Remove or comment out these statements in the final version of the code.
8. **Separation of Concerns:**
   * Consider separating the calculator logic from the GUI code. This will make it easier to modify or extend the calculator in the future.
9. **Code Optimization:**
   * The use of **eval** for calculation might be risky. Consider using safer alternatives to evaluate expressions.
10. **File Paths:**
    * The path to the image file is currently hard-coded. Make sure the image file is included with the code or use a relative path.
11. **Consistent Style:**
    * Ensure consistent coding style throughout the program, including indentation, spacing, and naming conventions.
12. **Testing:**
    * Test the calculator thoroughly with various inputs to identify and fix any unexpected behavior.

Addressing these points will contribute to better code quality, maintainability, and user experience.

**Chapter:3 Proposed Solution / Methodology**

To address the identified issues and improve the code, you can follow the proposed solutions and methodologies outlined below:

1. **User Interface Improvements:**
   * Consider using a grid layout manager to organize buttons systematically.
   * Add labels to indicate the current mode and improve the overall aesthetics of the GUI.
2. **Error Handling:**
   * Provide more specific error messages for different types of exceptions.
   * Enhance the error handling to give users a clear understanding of what went wrong.
3. **Code Structure:**
   * Organize the code into functions or classes.
   * Use separate functions for GUI setup, normal calculator logic, and scientific calculator logic.
4. **Magic Numbers:**
   * Define constants for magic numbers, such as window dimensions.
5. **Comments and Documentation:**
   * Add comments to explain complex sections of code.
   * Provide docstrings for functions to document their purpose and usage.
6. **Variable Naming:**
   * Use descriptive variable names for better readability.
7. **Redundant Code:**
   * Remove or comment out redundant print statements used for debugging.
8. **Separation of Concerns:**
   * Separate the calculator logic from the GUI code to improve maintainability.
9. **Code Optimization:**
   * Avoid using **eval** for calculations. Instead, use safer alternatives, such as parsing and evaluating expressions.
10. **File Paths:**
    * Use relative paths or include the image file with the code.

11.**Consistent Style:**

* + Ensure consistent coding style, including indentation and spacing.

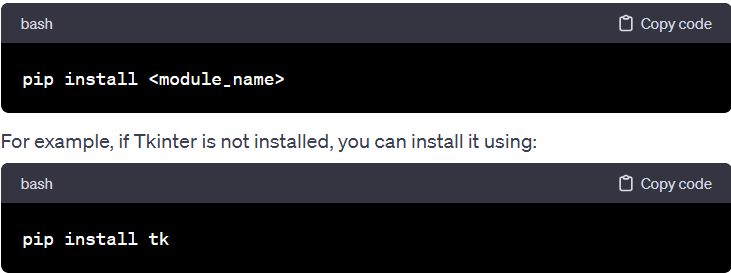
1. **Testing:**
   * Thoroughly test the calculator with various inputs to identify and fix any unexpected behavior.

By implementing these proposed solutions, you can enhance the code quality, maintainability, and user experience of the scientific calculator.Top of Form

**Chapter:4 System Requirements**

To run the provided **Python** code for the scientific calculator, you'll need to ensure that your system meets certain requirements. Here are the system requirements for running the code:

1. **Python Installation:**
   * Ensure that **Python** is installed on your system. The code is written in **Python** 3, and it's recommended to have **Python** 3.6 or later.
2. **Tkinter Library:**
   * **Tkinter** is the standard GUI (Graphical User Interface) library for **Python** and is used in the code for creating the calculator interface. **Tkinter** is included with most **Python** installations, so you typically don't need to install it separately.
3. **Image File:**
   * The code includes an image file ("img2.png") for the calculator's picture label. Make sure that the image file is present in the specified path or update the code with the correct path to the image file.
4. **Operating System:**
   * The code is platform-independent and should run on Windows, macOS, and Linux systems.
5. **Additional Notes:**
   * If you encounter any issues related to missing modules, you can install them using the following command:



* + Ensure that the code is saved with the ".py" extension, and run it using a **Python** interpreter.
  + The code uses the **“math”** module for mathematical functions, and this module is part of the standard library.

By meeting these requirements, you should be able to run the scientific calculator code successfully on your system. If you encounter any issues or have specific platform-related concerns, please refer to the documentation for the respective software and libraries used in the code.

Top of Form

**Chapter:5 Technology Used**

Here are the key technologies used in the development of the calculator:

1. **Python:**
   * **Python** is a high-level, versatile programming language known for its readability and simplicity. It is widely used for various applications, including GUI development, web development, data science, and more.
2. **Tkinter:**

**Tkinter** is the standard GUI toolkit that comes bundled with **Python**. It provides tools for creating graphical interfaces and is commonly used for developing desktop applications. **Tkinter** is based on the Tk GUI toolkit and is used for creating windows, buttons, text fields, and other GUI elements.

1. **Visual Studio Code:**

Visual Studio Code is a free source code editor, made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code

completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. The **python** extension in Visual Studio Code makes it an excellent video editor.

1. **math Module:**
   * The code utilizes the **math** module, which is part of the **Python** standard library. The **math** module provides mathematical functions such as square root (**sqrt**), exponentiation (**pow**), trigonometric functions (**sin**, **cos**, **tan**), and factorial (**factorial**).
2. **Integrated Development Environment (IDE):**
   * The code can be developed and executed using any **Python**-compatible IDE or text editor. Examples of popular IDEs include Visual Studio Code, PyCharm, Jupyter Notebooks, and others.
3. **Version Control:**
   * While not explicitly mentioned in the code, version control systems like Git can be used to manage and track changes to the codebase, collaborate with others, and maintain a history of modifications.
4. **Operating System:**
   * The code is designed to be platform-independent and should run on various operating systems, including Windows, macOS, and Linux.
5. **Image Processing:**
   * The code includes the use of an image file ("img2.png") for a picture label. This is not a technology per se, but it involves loading and displaying an image in the GUI. The image processing aspect is handled by **Tkinter**.

These technologies collectively enable the creation of a functional and interactive scientific calculator with a graphical user interface. **Python**, with its extensive standard library and third-party modules, provides a robust foundation for developing diverse applications, including desktop utilities like the scientific calculator in this code. **Tkinter**, being a part of the standard library, simplifies the process of building graphical interfaces for **Python** applications.

**1.Python**

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

• Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

• Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

• Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

**Python Features**

**Python's features include –**

• Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

• Easy-to-read − Python code is more clearly defined and visible to the eyes.

• Easy-to-maintain − Python's source code is fairly easy-tomaintain.

• A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

• Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

• Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

• Databases − Python provides interfaces to all major commercial databases.

• GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

• Scalable − Python provides a better structure and support for large programs than shell scripting.

**2.Tkinter Programming**

**Tkinter** is the standard GUI library for **Python**. **Python** when combined with **Tkinter**

provides a fast and easy way to create GUI applications. **Tkinter** provides a powerful

object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using **Tkinter** is an easy task. All you need to do is perform

the following steps −

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

user.



**This would create**

**a following window: –**

**Example:**

#!/usr/bin/**python**

import **Tkinter**

top = **Tkinter**.Tk()

# Code to add widgets will go here...

top.mainloop()

**Tkinter Widgets**

**Tkinter** provides various controls, such as buttons, labels and text boxes used in a

GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in **Tkinter**. We present these widgets as well

as a brief description in the following : −

**Operator & Description**

**1 Button -** The Button widget is used to display buttons in your application.

**2 Canvas -** The Canvas widget is used to draw shapes, such as lines, ovals, polygons and

rectangles, in your application.

**3 Checkbutton -** The Checkbutton widget is used to display a number of options as checkboxes.

The user can select multiple options at a time.

**4 Entry -** The Entry widget is used to display a single-line text field for accepting values from

a user.

**5 Frame -** The Frame widget is used as a container widget to organize other widgets.

**6 Label -** The Label widget is used to provide a single-line caption for other widgets. It can

also contain images.

**7 Listbox -** The Listbox widget is used to provide a list of options to a user.

**8 Menubutton -** The Menubutton widget is used to display menus in your application.

**9 Menu -** The Menu widget is used to provide various commands to a user. These commands

are contained inside Menubutton.

**10 Message -** The Message widget is used to display multiline text fields for accepting values from

a user.

**11 Radiobutton -** The Radiobutton widget is used to display a number of options as radio buttons.

The user can select only one option at a time.

**12 Scale -** The Scale widget is used to provide a slider widget.

**13 Scrollbar -** The Scrollbar widget is used to add scrolling capability to various widgets, such as

list boxes.

**14 Text -** The Text widget is used to display text in multiple lines.

**15 Toplevel -** The Toplevel widget is used to provide a separate window container.

**16 Spinbox -** The Spinbox widget is a variant of the standard **Tkinter** Entry widget, which can be

used to select from a fixed number of values.

**17 PanedWindow -** A PanedWindow is a container widget that may contain any number of panes,

arranged horizontally or vertically.

**18 LabelFrame -** A labelframe is a simple container widget. Its primary purpose is to act as a spacer

or container for complex window layouts.

**19 tkMessageBox -** This module is used to display message boxes in your applications.

**Code Structure and Organization**

**The source code for the, Scientific Calculator is as follows:-**

*#scientific calculator*

**from** tkinter **import\***

**from** tkinter**.**messagebox **import** **\***

**import** math **as** m

*#some useful variables*

font**=**("Verdana"**,**18**,**'bold ')

*#important function*

def clear**():**

        ex**=**textfield**.**get()

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

        textfield**.**delete(0**,**END)

        textfield**.**insert(0**,**ex)

def all\_clear**():**

    textfield**.**delete(0**,**END)

    textfield**.**insert(0**,** '0')

def click\_btn\_function**(**event**):**

**global** p

    print("Button Clicked")

    b**=**event**.**widget

    text**=**b['text']

    print(text)

**if** text**==**"x"**:**

        textfield**.**insert(END**,**"\*")

**return**

**if** text**==**"="**:**

**try:**

            ex**=**textfield**.**get()

            answer**=**eval(ex)

            textfield**.**delete(0**,**END)

            textfield**.**insert(0**,**answer)

**except** Exception **as** e**:**

            print("Error.."**,**e)

            showerror("Error"**,**e)

**return**

**if** textfield**.**get() **==** '0'**:**

        textfield**.**delete(0**,** END)

    textfield**.**insert(END**,**text)

*#windows*

window**=**Tk()

window**.**title("🔢 My Calculator")

window**.**geometry("445x570")

*#picture label*

pic**=**PhotoImage(file**=**"D:\Project\img2.png")

headinglabel**=**Label(window**,**image**=**pic)

headinglabel**.**pack(side**=**TOP**,**pady**=**15)

*#heading label*

heading**=**Label(window**,**text**=**"My calculator"**,**font**=**font**,**underline**=**0)

heading**.**pack(side**=**TOP)

*#textfield*

textfield**=**Entry(window**,**font**=**font**,**justify**=**RIGHT)

textfield**.**pack(side**=**TOP**,**pady**=**10**,**fill**=**X**,**padx**=**10)

textfield**.**insert(0**,** '0')

*#buttons*

buttonframe**=**Frame(window)

buttonframe**.**pack(side**=**TOP)

*#adding buttons*

temp**=**1

**for** i **in** range(0**,**3)**:**

**for** j **in** range(0**,**3)**:**

        btn**=** Button(buttonframe**,**text**=**str(temp)**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

        btn**.**grid(row**=**i**,**column**=**j)

        temp**=**temp**+**1

        btn**.**bind("<Button-1>"**,**click\_btn\_function)

zerobtn**=** Button(buttonframe**,**text**=**"0"**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

zerobtn**.**grid(row**=**3**,**column**=**0)

dotbtn**=** Button(buttonframe**,**text**=**"."**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

dotbtn**.**grid(row**=**3**,**column**=**1)

equalbtn**=** Button(buttonframe**,**text**=**"="**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

equalbtn**.**grid(row**=**3**,**column**=**2)

plusbtn**=** Button(buttonframe**,**text**=**"+"**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

plusbtn**.**grid(row**=**0**,**column**=**3)

minusbtn**=** Button(buttonframe**,**text**=**"-"**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

minusbtn**.**grid(row**=**1**,**column**=**3)

multbtn**=** Button(buttonframe**,**text**=**"x"**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

multbtn**.**grid(row**=**2**,**column**=**3)

divbtn**=** Button(buttonframe**,**text**=**"/"**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

divbtn**.**grid(row**=**3**,**column**=**3)

clearbtn**=** Button(buttonframe**,**text**=**"C"**,**font**=**font**,**width**=**11**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white"**,**command**=**clear)

clearbtn**.**grid(row**=**4**,**column**=**0**,**columnspan**=**2)

allclearbtn**=** Button(buttonframe**,**text**=**"AC"**,**font**=**font**,**width**=**11**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white"**,**command**=**all\_clear)

allclearbtn**.**grid(row**=**4**,**column**=**2**,**columnspan**=**2)

*#binding buttons*

plusbtn**.**bind("<Button-1>"**,**click\_btn\_function)

minusbtn**.**bind("<Button-1>"**,**click\_btn\_function)

multbtn**.**bind("<Button-1>"**,**click\_btn\_function)

divbtn**.**bind("<Button-1>"**,**click\_btn\_function)

zerobtn**.**bind("<Button-1>"**,**click\_btn\_function)

dotbtn**.**bind("<Button-1>"**,**click\_btn\_function)

equalbtn**.**bind("<Button-1>"**,**click\_btn\_function)

def enterClick**(**event**):**

    print("hi")

    e**=**Event()

    e**.**widget**=**equalbtn

    click\_btn\_function(e)

textfield**.**bind("<Return>"**,**enterClick)

*#######################################################################*

*#functions...*

scframe**=**Frame(window)

*#Buttons....*

sqrtbtn**=**Button(scframe**,**text**=**"√"**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

sqrtbtn**.**grid(row**=**0**,**column**=**0)

powbtn**=**Button(scframe**,**text**=**"^"**,**font**=**font**,**width**=**5**,**relief**=**RIDGE**,**activebackground**=**"orange"**,**activeforeground**=**"white")

powbtn**.**grid(row**=**0**,**column**=**1)

factbtn **=** Button(scframe**,** text**=**'x!'**,** font**=**font**,** width**=**5**,** relief**=**'ridge'**,** activebackground**=**'orange'**,**activeforeground**=**'white')

factbtn**.**grid(row**=**0**,** column**=**2)

radbtn **=** Button(scframe**,** text**=**'toRad'**,** font**=**font**,** width**=**5**,** relief**=**'ridge'**,** activebackground**=**'orange'**,**activeforeground**=**'white')

radbtn**.**grid(row**=**0**,** column**=**3)

degbtn **=** Button(scframe**,** text**=**'toDeg'**,** font**=**font**,** width**=**5**,** relief**=**'ridge'**,** activebackground**=**'orange'**,**activeforeground**=**'white')

degbtn**.**grid(row**=**1**,** column**=**0)

sinbtn **=** Button(scframe**,** text**=**'sinθ'**,** font**=**font**,** width**=**5**,** relief**=**'ridge'**,** activebackground**=**'orange'**,**activeforeground**=**'white')

sinbtn**.**grid(row**=**1**,** column**=**1)

cosbtn **=** Button(scframe**,** text**=**'cosθ'**,** font**=**font**,** width**=**5**,** relief**=**'ridge'**,** activebackground**=**'orange'**,**activeforeground**=**'white')

cosbtn**.**grid(row**=**1**,** column**=**2)

tanbtn **=** Button(scframe**,** text**=**'tanθ'**,** font**=**font**,** width**=**5**,** relief**=**'ridge'**,** activebackground**=**'orange'**,**activeforeground**=**'white')

tanbtn**.**grid(row**=**1**,** column**=**3)

normalcalc**=**TRUE

def calculate\_sc**(**event**):**

    print("btn..")

    btn**=**event**.**widget

    text**=**btn['text']

    print(text)

    ex**=**textfield**.**get()

    answer**=**""

**if** text**==**'toDeg'**:**

        print("cal degree")

        answer**=**str(m**.**degrees(float(ex)))

**elif** text**==**"toRad"**:**

        print('radian')

        answer**=**str(m**.**radians(float(ex)))

**elif** text**==**"x!"**:**

        print("cal factoria")

        answer**=**str(m**.**factorial(int(ex)))

**elif** text**==**"sinθ"**:**

        print("cal sin")

        answer**=**str(m**.**sin(m**.**radians(int(ex))))

**elif** text**==**"cosθ"**:**

        print("cal cos")

        answer**=**str(m**.**cos(m**.**radians(int(ex))))

**elif** text**==** "tanθ"**:**

        print("cal tag")

        answer**=**str(m**.**tan(m**.**radians(int(ex))))

**if** text**==**"√"**:**

        print("SQRT")

        answer**=**m**.**sqrt(int(ex))

**elif** text**==**"^"**:**

        print("pow")

        base**,**pow**=**ex**.**split(",")

        print(base)

        print(pow)

        answer**=**m**.**pow(int(base)**,**int(pow))

    textfield**.**delete(0**,**END)

    textfield**.**insert(0**,**answer)

def sc\_click**():**

**global** normalcalc

**if** normalcalc**:**

        buttonframe**.**pack\_forget()

*#add sc frame*

        scframe**.**pack(side**=**TOP**,**pady**=**20)

        buttonframe**.**pack(side**=**TOP)

        window**.**geometry("445x705")

*#sc*

        print("show scientific")

        normalcalc**=False**

**else:**

        print("show normal")

        scframe**.**pack\_forget()

        window**.**geometry("445x570")

        normalcalc**=True**

*#binding sc buttons*

sqrtbtn**.**bind("<Button-1>"**,**calculate\_sc)

powbtn**.**bind("<Button-1>"**,**calculate\_sc)

factbtn**.**bind("<Button-1>"**,**calculate\_sc)

radbtn**.**bind("<Button-1>"**,**calculate\_sc)

degbtn**.**bind("<Button-1>"**,**calculate\_sc)

cosbtn**.**bind("<Button-1>"**,**calculate\_sc)

sinbtn**.**bind("<Button-1>"**,**calculate\_sc)

tanbtn**.**bind("<Button-1>"**,**calculate\_sc)

fontMenu**=**(""**,**15)

menubar**=**Menu(window)

mode**=**Menu(menubar**,**font**=**fontMenu**,**tearoff**=**0)

mode**.**add\_checkbutton(label**=**"Scientific Calculator"**,**command**=**sc\_click)

menubar**.**add\_cascade(label**=**"mode"**,**menu**=**mode)

window**.**config(menu**=**menubar)

window**.**mainloop()

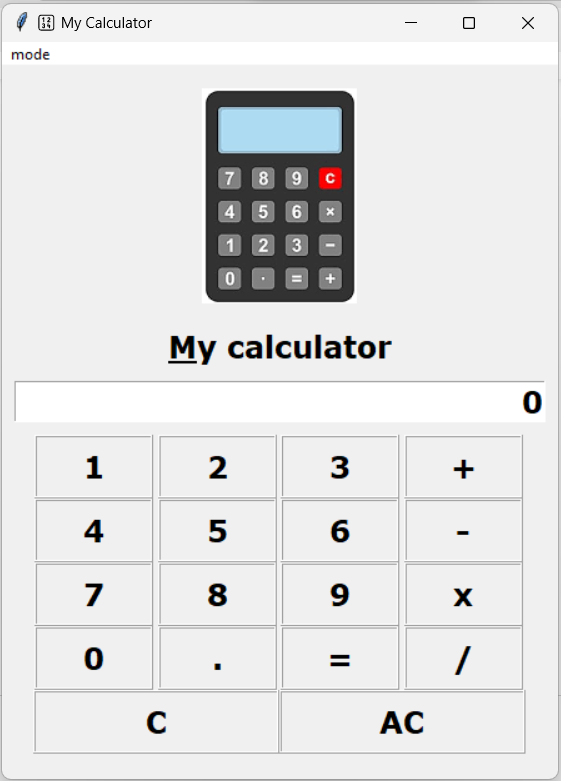
**RESULT**

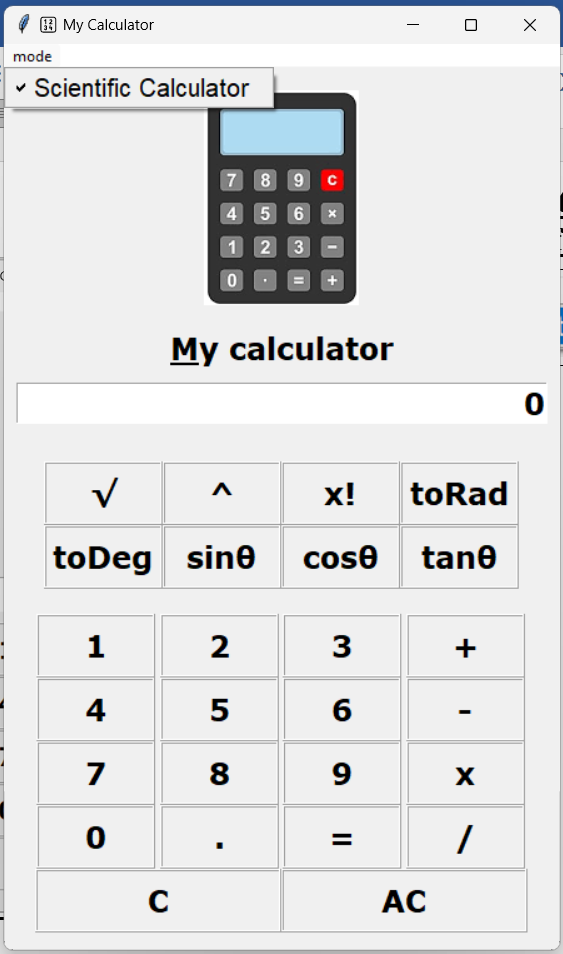
The provided code is a Python program that creates a graphical calculator using the Tkinter library. The calculator has two modes: a **basic calculator** and a **scientific calculator**. Here's a breakdown of the main components and functionalities:

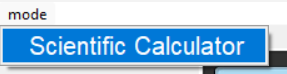
1. **Basic Calculator:**
   * The basic calculator includes buttons for digits 0-9, arithmetic operations (+, -, \*, /), a decimal point (.), and an equals (=) button.
   * The **click\_btn\_function** function handles button clicks and performs the corresponding actions. It updates the entry field with the pressed button's text.
   * The **clear** function removes the last character from the entry field.
   * The **all\_clear** function clears the entire entry field.
2. **Scientific Calculator:**
   * The scientific calculator mode can be activated by checking the "Scientific Calculator" option in the menu.
   * Additional scientific functions include square root (√), power (^), factorial (x!), conversion between degrees and radians (toDeg, toRad), and trigonometric functions (sinθ, cosθ, tanθ).
   * The **calculate\_sc** function handles the scientific calculator button clicks and performs the corresponding scientific calculations.
3. **Graphical User Interface (GUI):**
   * The Tkinter library is used for creating the graphical user interface.
   * The main window contains a title, a picture label, a heading label, and an entry field for input and display.
   * Calculator buttons are organized in a grid layout using the **Button** widget and are bound to the **click\_btn\_function** for basic calculator buttons and **calculate\_sc** for scientific calculator buttons.
   * There is a menu bar with an option to switch between basic and scientific calculator modes.
4. **Menu and Mode Switching:**
   * The menu bar includes an option to toggle between basic and scientific calculator modes.
   * Clicking on the "Scientific Calculator" option switches the calculator to the scientific mode, revealing additional buttons.
5. **Event Handling:**
   * The code handles events such as button clicks and Enter key presses.
   * The **enterClick** function is triggered when the Enter key is pressed, and it simulates a click on the equals (=) button.
6. **Code Organization:**
   * The code is organized into functions for better readability and maintainability.
   * Comments are added to explain the purpose and functionality of different sections of the code.
7. **User Interface Elements:**
   * The calculator's appearance is enhanced with a picture label at the top, providing a visual element to the GUI.
8. **Error Handling:**
   * Some basic error handling is implemented for the evaluation of expressions in the basic calculator, and more advanced error handling could be added based on specific use cases.

**Output**

**After clicking Mode then ScientificCalculator :-**

****



****

**6.Conclusion**

The proposed system is error free. Trivial concepts of Python language are implemented into the system. As, the usage of Python Tkinter as the GUI provided various controls, such as buttons, labels, and text boxes to build a user friendly application. The rapid expansion and use of the internet, confirms the splendid future and scope of the project.