A Report on

**SCIENTIFIC CALCULATOR**

PYTHON PROJECT

By – SAHIL PATHAN

CONTENT

|  |
| --- |
| Page no. |
| Introduction 3 |
| Objective 3 |
| Background 3 |
| Hardware Requirements 4 |
| Software Requirements 4 |
| Coding 5 |
| Output Screenshot 7 |
| Future Scope 7 |
| Conclusion 8 |
| Reference and Bibliography 8 |

**Introduction:**

Imagine a digital tool that not only performs everyday math but also tackles complex calculations like a pro – that's a scientific calculator. It's like having a math-savvy companion at your fingertips! This report is all about creating such a nifty calculator using a special library called Tkinter in Python. Tkinter turns coding magic into user-friendly buttons and displays, making it super easy to interact with the computer.

Think about those times you needed to crunch numbers quickly, from figuring out tips to solving equations. A scientific calculator takes these tasks to the next level, handling fancy math like trigonometry and logarithms effortlessly. This report is like a behind-the-scenes tour, showing how we put together a smart calculator using Tkinter's trickery. You'll see how computer code and visual design come together, making math not just powerful but also surprisingly enjoyable.

**Objective:**

The main objective of this project is to create a user-friendly and functional scientific calculator that can handle various mathematical operations. The calculator aims to provide a convenient tool for users to perform calculations quickly and accurately. By utilizing Tkinter's GUI capabilities, the calculator offers an intuitive interface with buttons for numbers, operators, and scientific functions.

**Background:**

In a world driven by numbers and computations, calculators have been an indispensable tool for centuries. From mechanical contraptions to the rise of electronic calculators, these devices have been a constant companion for mathematicians, scientists, engineers, and students. The advent of digital technology led to the development of more sophisticated calculators, capable of performing complex mathematical operations with ease. Among these, the scientific calculator emerged as a key player, catering to a diverse range of mathematical needs. From scientific research to daily life calculations, these calculators provide a bridge between complex mathematics and practical applications.

However, the evolution of calculators didn't stop at physical devices. The digital era paved the way for software-based calculators, offering the same capabilities in a virtual form. These digital calculators are not only more accessible but also customizable, allowing users to tailor the experience to their needs. This project dives into the digital landscape by harnessing the power of Python, a versatile programming language, and Tkinter, a GUI library, to craft a scientific calculator that marries computation with a user-friendly interface.

Tkinter, short for "Tk interface," is a standard library in Python that facilitates the creation of graphical user interfaces (GUIs). It's a toolkit that provides the building blocks to design and develop interactive and visually appealing applications. Tkinter's simplicity and ease of use make it an ideal choice for both beginners and experienced programmers to create desktop applications without delving into complex UI frameworks. The library includes a variety of widgets – buttons, labels, entry fields, etc. – that can be arranged and customized to form a user-friendly interface. Whether it's a calculator, like the one in this project, or a more elaborate application, Tkinter simplifies the process of turning code into interactive experiences, making it an indispensable tool for GUI development in Python.

**Hardware and Software Requirements:**

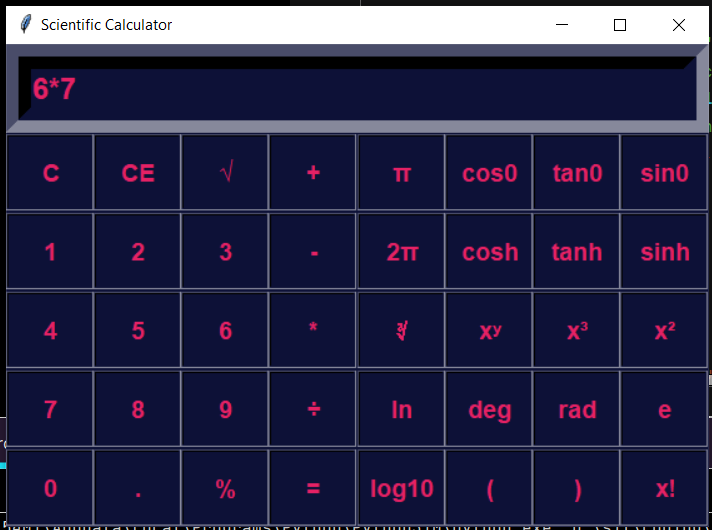
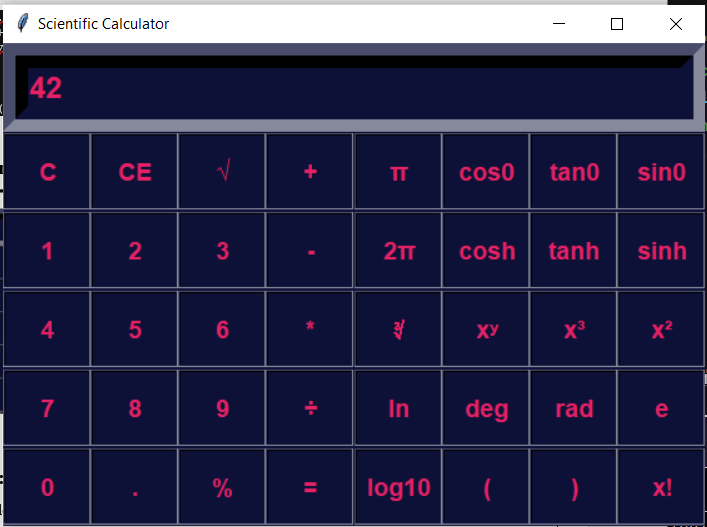
|  |  |
| --- | --- |
| **Hardware Tools** | **Minimum Requirements** |
| Processor | i3 or above |
| Hardware | 2 GB |
| RAM | 4 GB |
| Monitor | 11” – 17” Coloured |
| Input Devices | Keyboard, Mouse |

|  |  |
| --- | --- |
| **Software Tools** | **Minimum Requirements** |
| Platform | Windows, Linux or MacOS |
| Operating System | Windows, Linux or MacOS |
| Technology | Python |
| Scripting Language | Python (version 3.x) |
| Libraries | Tkinter libraries |
| IDE` | VSCode, PyCharm |

**Coding:**

from tkinter import \*  
import math  
  
def click(value):  
 ex = display.get()  
 ans=""  
  
 try:  
  
 if value == 'C':  
 ex = ex[**0**:len(ex)-**1**]  
 display.delete(**0,** END)  
 display.insert(**0,** ex)  
 return  
  
 elif value == 'CE':  
 display.delete(**0,** END)  
  
 elif value == "√":  
 ans = math.sqrt(eval(ex))  
  
 elif value == "π":  
 ans = math.pi  
  
 elif value == "cos0":  
 ans = math.cos(math.radians(eval(ex)))  
  
 elif value == "tan0":  
 ans = math.tan(math.radians(eval(ex)))  
  
 elif value == "sin0":  
 ans = math.sin(math.radians(eval(ex)))  
  
 elif value == "2π":  
 ans = **2**\*math.pi  
  
 elif value == "cosh":  
 ans = math.cosh(eval(ex))  
  
 elif value == "tanh":  
 ans = math.tanh(eval(ex))  
  
 elif value == "sinh":  
 ans = math.sinh(eval(ex))  
  
 elif value == chr(**8731**):  
 ans = eval(ex)\*\*(**1**/**3**)  
  
 elif value == "x\u02b8":  
 display.insert(END**,** "\*\*")  
 return  
  
 elif value == "x\u00B3":  
 ans = eval(ex) \*\* **3** elif value == "x\u00B2":  
 ans = eval(ex) \*\* **2** elif value == "ln":  
 ans = math.log2(eval(ex))  
  
 elif value == "deg":  
 ans = math.degrees(eval(ex))  
  
 elif value == "rad":  
 ans = math.radians(eval(ex))  
  
 elif value == "e":  
 ans = math.e  
  
 elif value == "log10":  
 ans = math.log10(eval(ex))  
  
 elif value == "x!":  
 ans = math.factorial(eval(ex))  
  
 elif value == chr(**247**):  
 display.insert(END**,** "/")  
 return  
  
 elif value == "=":  
 ans = eval(ex)  
  
 else:  
 display.insert(END**,** value)  
 return  
  
 display.delete(**0,** END)  
 display.insert(**0,** ans)  
  
 except SyntaxError:  
 pass  
  
root = Tk()  
root.title("Scientific Calculator")  
root.geometry("562x386")  
root.config(bg="#0d1137")  
  
display = Entry(root**,** bg="#0d1137"**,** fg="#e52165"**,** bd=**20,** relief=SUNKEN**,** width=**40,** font=("arial"**, 18,** "bold"))  
display.grid(row=**0,** column=**0,** columnspan=**8**)  
  
but\_lst = ["C"**,** "CE"**,** "√"**,** "+"**,** "π"**,** "cos0"**,** "tan0"**,** "sin0"**,** "1"**,** "2"**,** "3"**,** "-"**,** "2π"**,** "cosh"**,** "tanh"**,** "sinh"**,** "4"**,** "5"**,** "6"**,** "\*"**,** chr(**8731**)**,** "x\u02b8"**,** "x\u00B3"**,** "x\u00B2"**,** "7"**,** "8"**,** "9"**,** chr(**247**)**,** "ln"**,** "deg"**,** "rad"**,** "e"**,** "0"**,** "."**,** "%"**,** "="**,** "log10"**,** "("**,** ")"**,** "x!"]  
  
rowval = **1**colval = **0**for i in but\_lst:  
 button=Button(root**,** text=i**,** font=("arial"**, 14,** "bold")**,** width=**5,** height=**2,** bd=**2,** relief=SUNKEN**,** bg="#0d1137"**,** fg="#e52165"**,** activebackground="#e52165"**,** command=lambda button=i: click(button))  
 button.grid(row=rowval**,** column=colval**,** pady=**1**)  
 colval+=**1** if colval>**7**:  
 rowval+=**1** colval=**0**root.mainloop()

**Output Screenshot:**

** **

** **

**Future Scope:**

The scientific calculator project can be expanded and improved in several ways:

* Adding more scientific functions and constants.
* Enhancing the user interface with themes and customization options.
* Implementing unit conversions and other utility features.
* Creating mobile versions of the calculator for increased accessibility.

**Conclusion:**

In wrapping up our journey through the creation of a scientific calculator using Tkinter in Python, we find ourselves at the crossroads of mathematics and technology. This project has demonstrated the incredible synergy between programming prowess and user interface design, showcasing how a simple codebase can transform into a versatile tool that empowers users to conquer complex calculations with ease. The harmonious marriage of mathematical functions and an intuitive GUI highlights the potential of digital applications to seamlessly integrate functionality and user experience.

As we bid adieu to our scientific calculator project, it's clear that the intersection of coding and design holds boundless possibilities. The journey doesn't end here – it extends into a realm of further exploration and innovation. With the foundation laid by this project, the future beckons with opportunities to enhance the calculator's features, expand its capabilities, and even delve into more intricate GUI designs. The scientific calculator, while a timeless tool, now stands as a testament to the dynamic interplay between mathematics, technology, and user-centric design, reminding us that with a bit of code and a touch of creativity, the ordinary can evolve into the extraordinary.

**Reference and Bibliography:**

* <https://docs.python.org/3/library/tkinter.html>
* <https://www.geeksforgeeks.org/python-tkinter-entry-widget/>
* <https://www.geeksforgeeks.org/python-add-style-to-tkinter-button/?ref=lbp>
* <https://www.youtube.com/playlist?list=PLPy7TevxmYAixB-9Q5p3HezYmrU6wZ7J9>