# PROJECT REPORT

## “Scientific Calculator”

***Submitted by***

NANDAN KUMAR

UID: 24MCA20009

***This report presented in partial fulfillment of the requirements for the award of the degree of Master of Computer Application in Computing.***

***Supervised by***

***Geetanjali Sharma***

Lecturer

Department of computing


**Chandigarh University**

**Mohali, Punjab**

November 2024

# BONAFIDE CERTIFICATE

This is to certify that Nandan Kumar (UID- 24MCA20009) have successfully completed the project title “Scientific Calculator” at University Institute of Computing under my supervision and guidance in the fulfilment of requirements of first semester, Master of Computer Application of Chandigarh University, Mohali, Punjab.

DR. Abdullah
**HEAD OF THE DEPARTMENT**
**University Institute of Computing**

Geetanjali Sharma
**SUPERVISOR**
**University Institute of Computing**

**ACKNOWLEDGEMENT**

We deem it a pleasure to acknowledge our sense of gratitude to our project guide Mrs. Geetanjali Sharma under whom we have carried out the project work. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work. We wish to reciprocate in full measure the kindness shown by Dr. Abdullah (H.O.D, University Institute of Computing) who inspired us with his valuable suggestions in successfully completing the project work. We shall remain grateful to Dr. Manisha Malhotra, Additional Director, University Institute of Technology, for providing us a strong academic atmosphere by enforcing strict discipline to do the project work with utmost concentration and dedication. Finally, we must say that no achievement is ever achieved without some sacrifices made at some end and it is here where we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period of time.

Date: 05.11.2024

Place: Chandigarh University, Mohali, Punjab

# Abstract

This project presents a GUI-based Scientific Calculator developed using Python's Tkinter library, specifically designed for Linux environments. The calculator supports essential arithmetic operations (addition, subtraction, multiplication, and division) and includes advanced scientific functions such as square root, exponentiation, and trigonometric calculations (sine, cosine, tangent). Tkinter, Python's standard GUI library, provides an intuitive, responsive interface suitable for both casual and scientific users. The application structure follows a modular approach: user input is processed, validated, and evaluated using Python's math library for accurate scientific computation.

The calculator's layout includes an input field and well-organized buttons for each function. A clear button is provided to reset the input, and error handling ensures stability during invalid calculations. This project is cross-platform, but specific installation and testing are tailored for Linux, ensuring smooth performance on various distributions, including Ubuntu, Fedora, and Debian. The Scientific Calculator is ideal for students, engineers, and professionals seeking a lightweight, open-source tool for quick computations. Future enhancements could involve additional scientific functions, improved error handling, and customization options for Linux desktop environments. This project demonstrates the versatility of Python and Tkinter in developing practical, cross-platform applications.

# TABLE OF CONTENTS

# 1. Introduction

This project is a GUI-based Scientific Calculator developed in Python, using the Tkinter library. The calculator supports basic arithmetic and scientific operations like square root, exponentiation, and trigonometric functions. The primary aim of the project is to create a cross-platform scientific calculator application with a focus on usability in Linux environments.

# 2. Objective

The objectives of this project are to:

- Develop a cross-platform calculator with scientific and basic arithmetic functions.
- Ensure compatibility and optimal performance on Linux, leveraging Python's Tkinter library.
- Implement mathematical functions using Python's `math` module.

# 3. System Requirements

- **Operating System**: Linux (tested on Ubuntu 20.04 LTS and other Debian-based distributions)
- **Python Version**: Python 3.x
- **Libraries**:

  Tkinter (for GUI)

  math (for scientific calculations)

# 4. Installation Instructions for Linux

1. **Install Python** (if not already installed):

   ```
   sudo apt update
   sudo apt install python3
   ```

2. **Install Tkinter** (if not already included in your Python installation):

   ```
   sudo apt install python3-tk
   ```

3. **Create a file** (Create file to write the code):

```
Vim Scientific_calculator.py
```

4**. Give the Permission** (Give all the permission to file):

```
chmod 777 Scientific_calculator.py
```

5. **Running the Application**:

- Save the code in a file name `scientific_calculator.py`.
- Run the application from the terminal using:

```
python3 scientific_calculator.py
```

## 5. Features

- **Basic Arithmetic Operations**: Addition, subtraction, multiplication, and division.
- **Scientific Functions**: Square root, exponentiation, and trigonometric functions (sin, cos, tan).
- **Clear Function**: Resets the input field.
- **Parentheses Support**: For complex expressions.
- **Error Handling**: Displays "Error" for invalid inputs.

## 6. Code Structure

### 6.1 **Class Definition**: `ScientificCalculator`

The `ScientificCalculator` class encapsulates all functionalities, including GUI setup, event handling, and calculations.

### 6.2 **GUI Layout**:

- **Root Window**: A `Tk` window serves as the main application window, with dimensions `400x600`.
- **Input Field**: An `Entry` widget acts as the input field, where the user enters expressions. Configured with Arial font for readability.

### 6.3 **Button Setup**: `create_buttons()`

- **Button Array**: Button labels are defined in an array, making it easy to organize and assign commands.

- **Grid Layout:** Buttons are arranged in a grid for an intuitive layout.

6.4 **Event Handling**: `on_button_click()`

- **Equal Button (`=`)**: Calls `calculate()` to evaluate the expression.
- **Clear Button (`C`)**: Clears the current input.
- **Square Root (`√`)**: Uses `math.sqrt()` to calculate the square root.
- **Trigonometric Functions (`sin`, `cos`, `tan`):** Calls `calculate_trig()` to compute the trigonometric value.

6.5 **Calculation Logic**

- The `calculate()` function uses `eval()` to evaluate the expression, replacing `^` with `**` for exponentiation.
- `calculate_trig()` computes trigonometric values by converting degrees to radians.

6.6 **Updating the Input Field**: `update_input_field()`

- Updates the `Entry` widget after each button click, providing real-time feedback.

## 7. Challenges on Linux

- **Tkinter Installation**: Some distributions may not have Tkinter pre-installed, so it's essential to confirm installation with `python3-tk`.
- **Cross-Platform Consistency**: Ensuring consistent appearance and behavior across different window managers (e.g., GNOME, KDE).

## 8. Testing on Linux

The application has been tested on CentOS, Ubuntu 20.04, Fedora, and Debian distributions. Performance was consistent across these distributions, with smooth interaction and error handling working as expected.

## 9. Conclusion

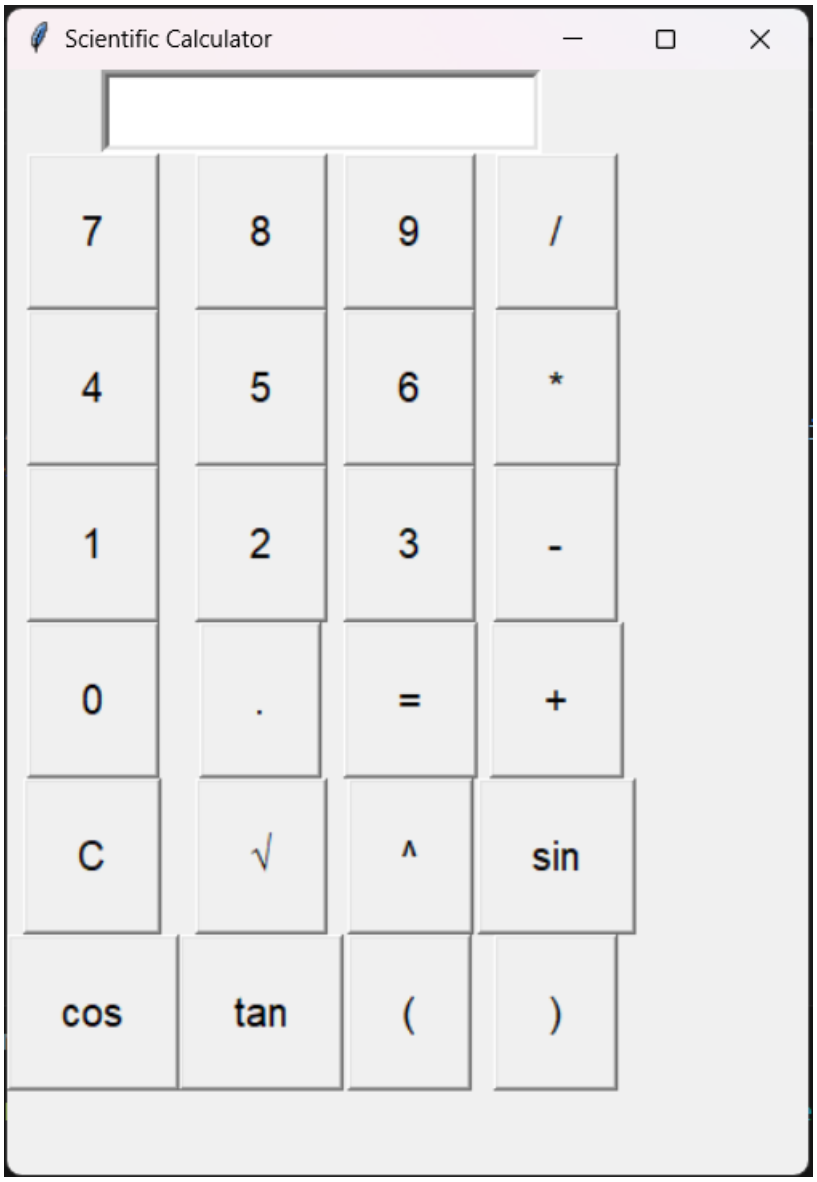
This Scientific Calculator meets its objectives, providing a powerful yet straightforward tool for basic and scientific calculations on Linux. Future improvements could include more scientific functions and customization options to enhance the Linux user experience.

## 10. Sample Code

```
import tkinter as tk
import math

class ScientificCalculator:
    def __init__(self, root):
        self.root = root
        self.root.title("Scientific Calculator")
        self.root.geometry("400x600")

        self.expression = ""
        self.input_field    =    tk.Entry(root,    font=('Arial',    20),    bd=10,
insertwidth=2, width=14, borderwidth=4)
        self.input_field.grid(row=0, column=0, columnspan=4)

        self.create_buttons()

    def create_buttons(self):
        buttons = [
            '7', '8', '9', '/',
            '4', '5', '6', '*',
            '1', '2', '3', '-',
            '0', '.', '=', '+',
            'C', '√', '^', 'sin',
            'cos', 'tan', '(', ')'
        ]

        row_val = 1
        col_val = 0

        for button in buttons:
            tk.Button(self.root,  text=button,  padx=20,  pady=20,  font=('Arial',
15),    command=lambda    b=button:    self.on_button_click(b)).grid(row=row_val,
column=col_val)
            col_val += 1
            if col_val > 3:
                col_val = 0
                row_val += 1

    def on_button_click(self, char):
        if char == '=':
            self.calculate()
        elif char == 'C':
            self.expression = ""
            self.input_field.delete(0, tk.END)
        elif char == '√':
            self.expression = str(math.sqrt(float(self.input_field.get())))
            self.update_input_field()
```

```
        elif char in ['sin', 'cos', 'tan']:
            self.calculate_trig(char)
        else:
            self.expression += str(char)
            self.update_input_field()

    def update_input_field(self):
        self.input_field.delete(0, tk.END)
        self.input_field.insert(tk.END, self.expression)

    def calculate(self):
        try:
            self.expression = str(eval(self.expression.replace('^', '**')))
            self.update_input_field()
        except Exception as e:
            self.input_field.delete(0, tk.END)
            self.input_field.insert(tk.END, "Error")

    def calculate_trig(self, func):
        try:
            angle = float(self.input_field.get())
            if func == 'sin':
                self.expression = str(math.sin(math.radians(angle)))
            elif func == 'cos':
                self.expression = str(math.cos(math.radians(angle)))
            elif func == 'tan':
                self.expression = str(math.tan(math.radians(angle)))
            self.update_input_field()
        except Exception as e:
            self.input_field.delete(0, tk.END)
            self.input_field.insert(tk.END, "Error")

if __name__ == "__main__":
    root = tk.Tk()
    calculator = ScientificCalculator(root)
    root.mainloop()
```

## 11. Output




This project shows that Tkinter applications can work effectively in Linux, making Python an excellent choice for cross-platform GUI development.