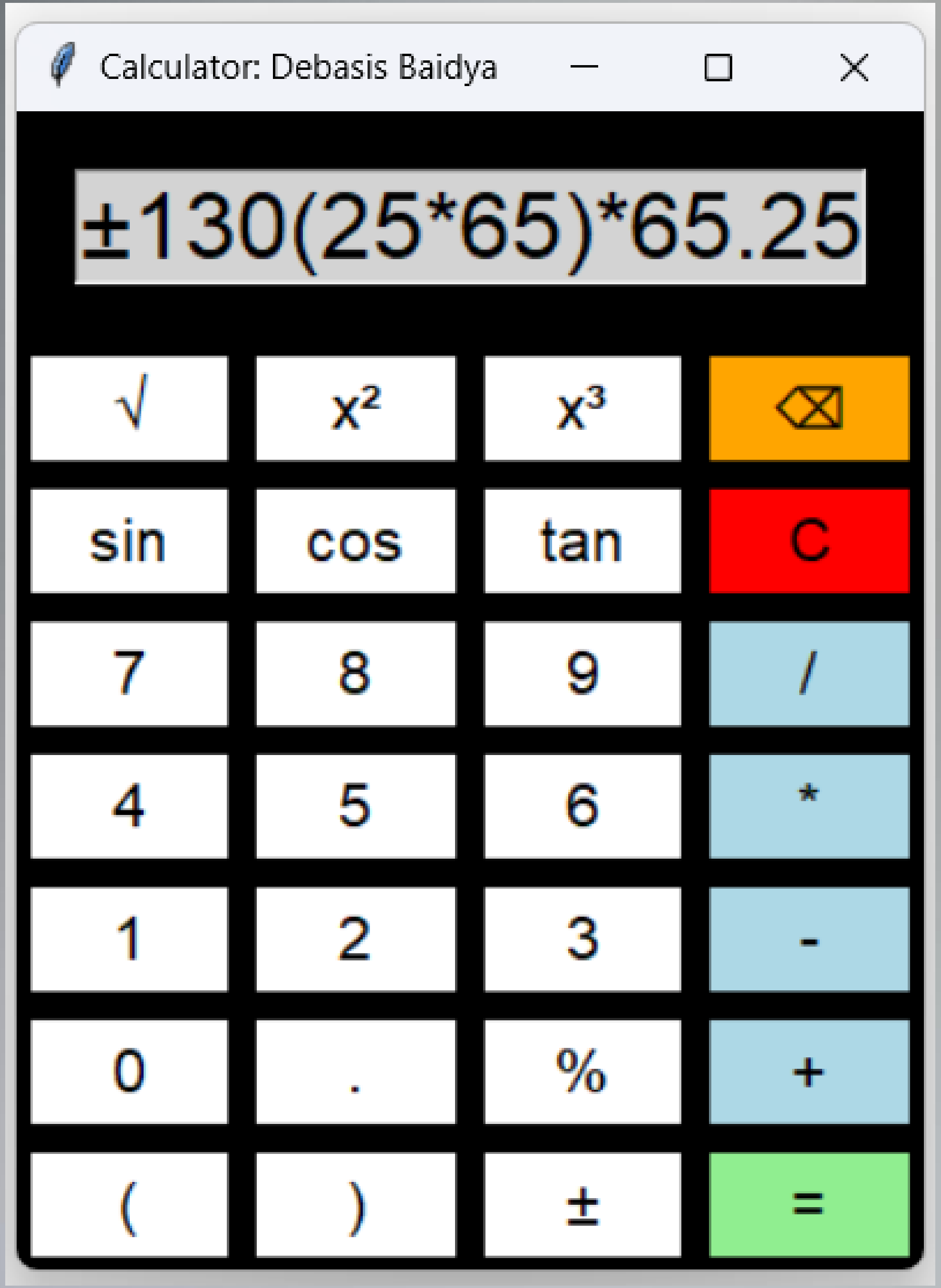
GUI Calculator Project Overview

This presentation will walk through **a GUI Calculator Project** built using **Python** and **Tkinter**, highlighting its key features, code structure, and functionality to demonstrate how the calculator works.

The project utilizes the **Math** module to extend its capabilities beyond basic arithmetic, allowing for operations like square roots and trigonometric functions. Additionally, the **Messagebox** library plays a crucial role in error handling and user feedback, alerting users when they attempt to divide by zero or enter invalid expressions, thus ensuring a smooth user experience.

Overall, this project combines these elements to create a powerful and user-friendly calculator application.

by Debasis Baidya



Project Structure

The project is structured around a Tkinter main window, with an entry widget for input and buttons that trigger calculations through the on_button_click function.

01

Tkinter Framework:

Tkinter is used to build the GUI of the calculator, handling the layout and user interactions.

02

Core Functionality:

on_button_click(ch)

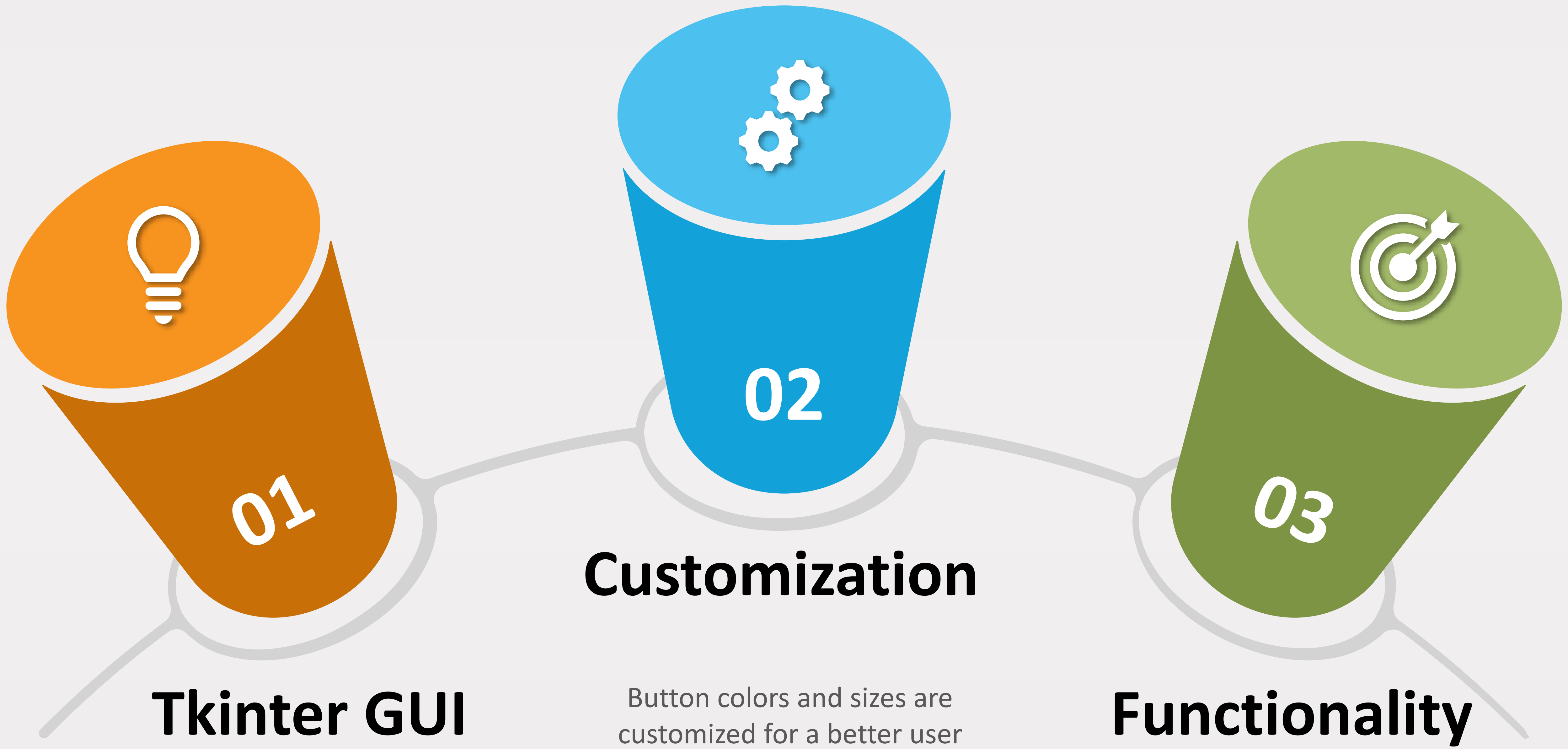
This function updates the display based on user input, performs calculations, and shows results or errors.

03

Math Operations:

Includes basic operations and functions like sin, cos, tan, square root, square, and cube, all using Python's math module.

User Interface Design



Tkinter GUI

The interface uses a grid layout with buttons for operations and digits

Customization

Button colors and sizes are customized for a better user experience.

Functionality

The calculator supports standard operations, square roots, and trigonometric functions.

Percentage Calculation

1

Expression Parsing

The calculate_percentage function splits the expression based on operators and identifies percentage values.

2

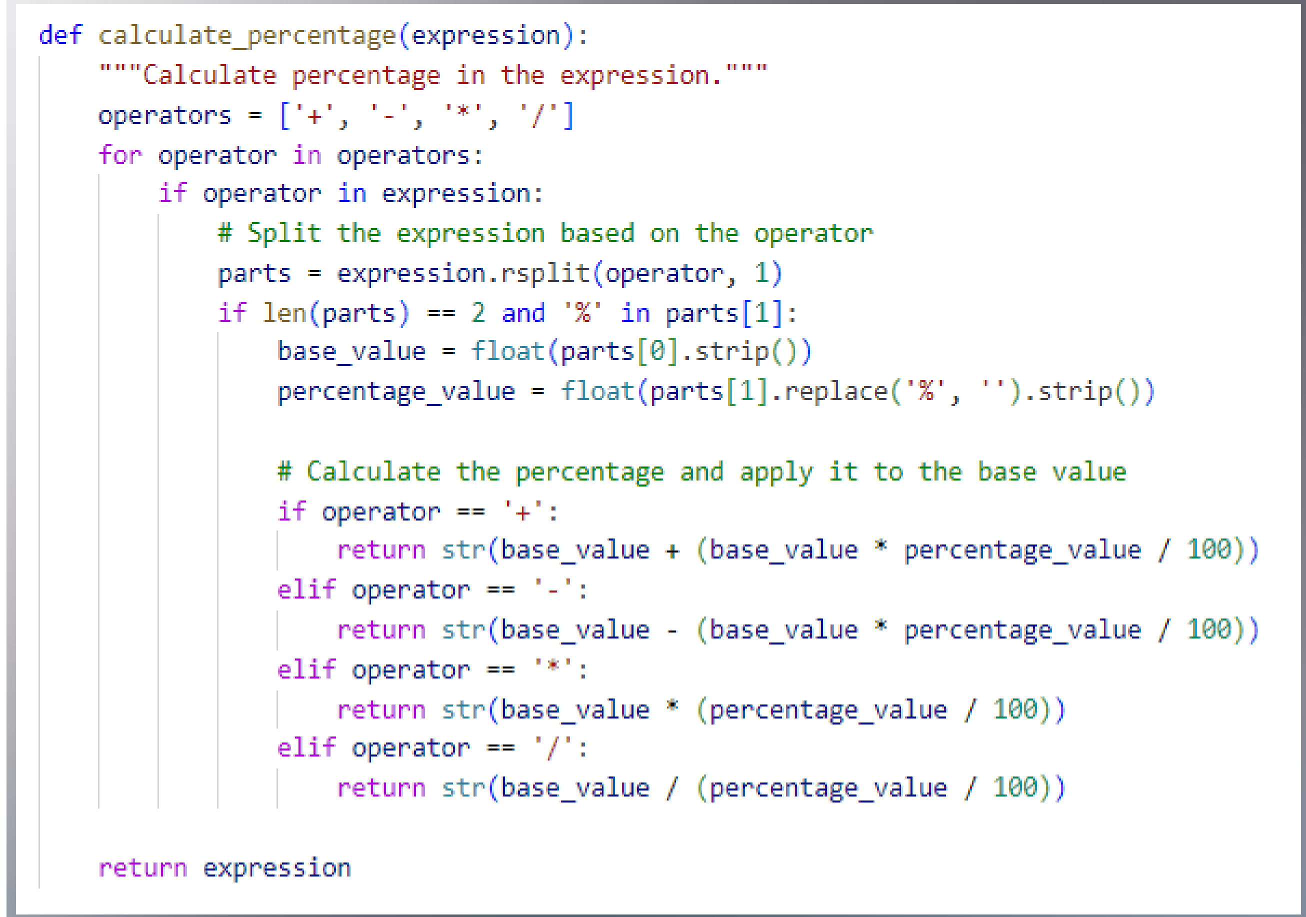
Percentage Computation

The function applies the percentage to the base value to compute the final result.

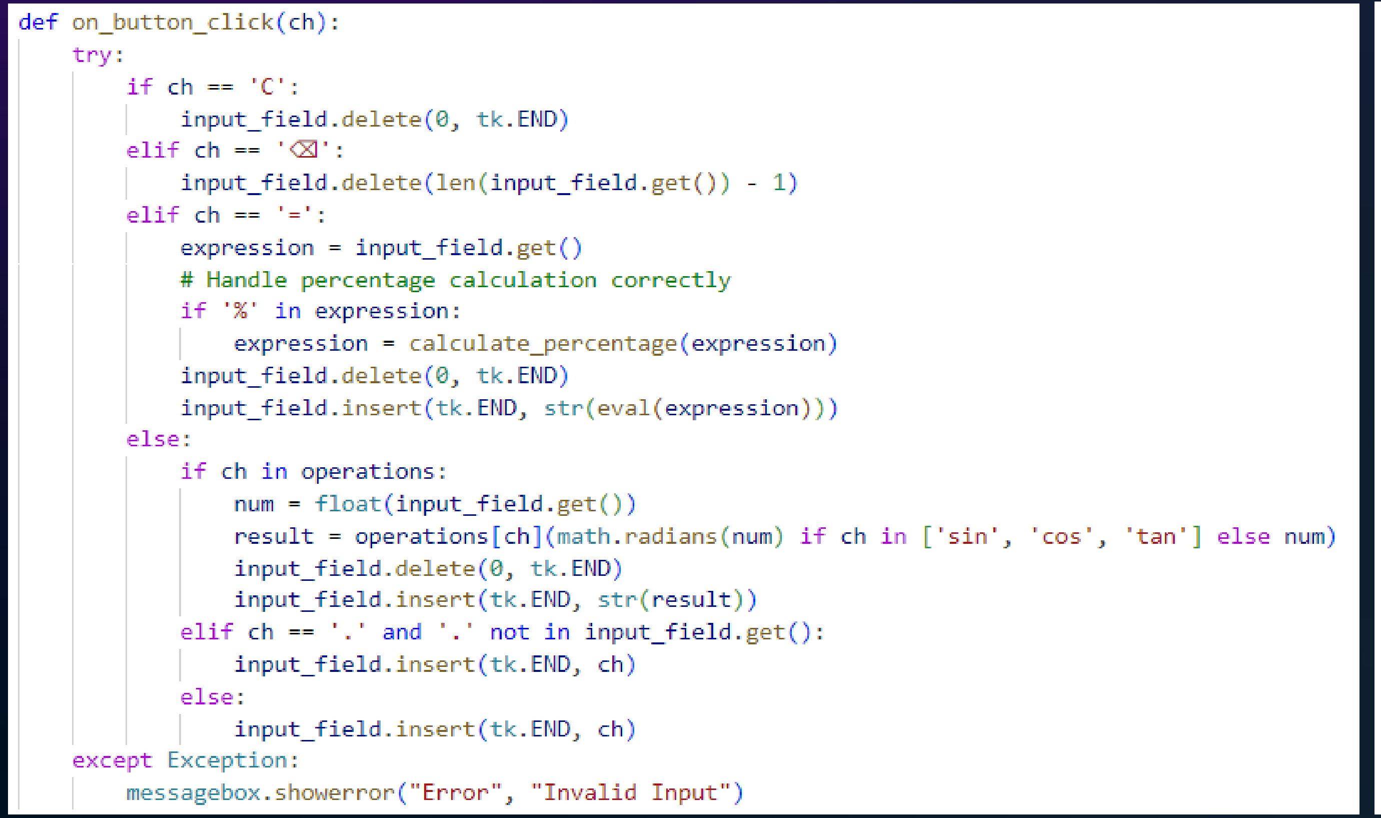
3

Handling Percentages

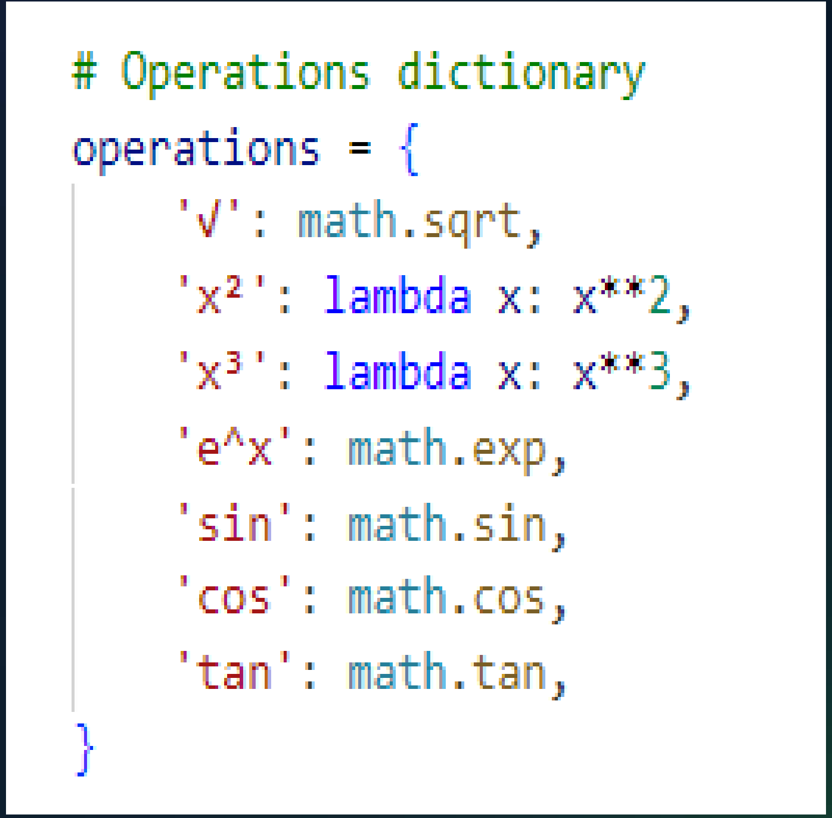
This allows the calculator to accurately process expressions like GST Calculation "100 + 18% = 118” & Discount Calculation “100-15% = 85”.



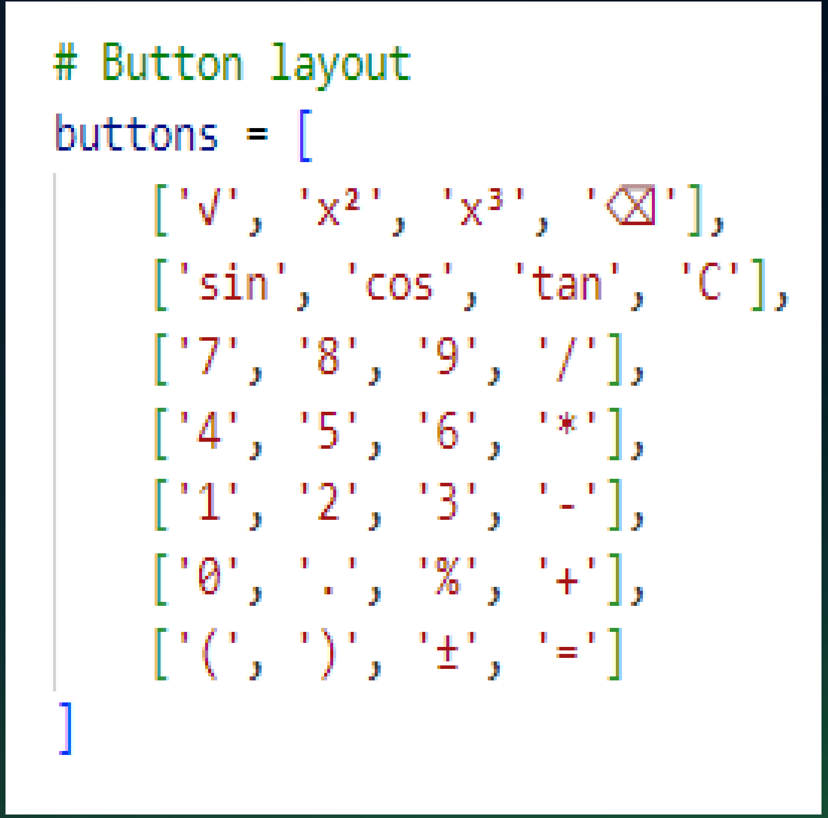
```python
def calculate_percentage(expression):
    """Calculate percentage in the expression."""
    operators = ['+', '-', '*', '/']
    for operator in operators:
        if operator in expression:
            # Split the expression based on the operator
            parts = expression.rsplit(operator, 1)
            if len(parts) == 2 and '%' in parts[1]:
                base_value = float(parts[0].strip())
                percentage_value = float(parts[1].replace('%', '').strip())

                # Calculate the percentage and apply it to the base value
                if operator == '+':
                    return str(base_value + (base_value * percentage_value / 100))
                elif operator == '-':
                    return str(base_value - (base_value * percentage_value / 100))
                elif operator == '*':
                    return str(base_value * (percentage_value / 100))
                elif operator == '/':
                    return str(base_value / (percentage_value / 100))

    return expression
```



```python
def on_button_click(ch):
    try:
        if ch == 'C':
            input_field.delete(0, tk.END)
        elif ch == '⌫':
            input_field.delete(len(input_field.get()) - 1)
        elif ch == '=':
            expression = input_field.get()
            # Handle percentage calculation correctly
            if '%' in expression:
                expression = calculate_percentage(expression)
            input_field.delete(0, tk.END)
            input_field.insert(tk.END, str(eval(expression)))
        else:
            if ch in operations:
                num = float(input_field.get())
                result = operations[ch](math.radians(num) if ch in ['sin', 'cos', 'tan'] else num)
                input_field.delete(0, tk.END)
                input_field.insert(tk.END, str(result))
            elif ch == '.' and '.' not in input_field.get():
                input_field.insert(tk.END, ch)
            else:
                input_field.insert(tk.END, ch)
    except Exception:
        messagebox.showerror("Error", "Invalid Input")
```



```python
# Operations dictionary
operations = {
    '√': math.sqrt,
    'x²': lambda x: x**2,
    'x³': lambda x: x**3,
    'e^x': math.exp,
    'sin': math.sin,
    'cos': math.cos,
    'tan': math.tan,
}
```



```python
# Button layout
buttons = [
    ['√', 'x²', 'x³', '⌫'],
    ['sin', 'cos', 'tan', 'C'],
    ['7', '8', '9', '/'],
    ['4', '5', '6', '*'],
    ['1', '2', '3', '-'],
    ['0', '.', '%', '+'],
    ['(', ')', '±', '=']
]
```

Handling User Input

1 **on_button_click Function**

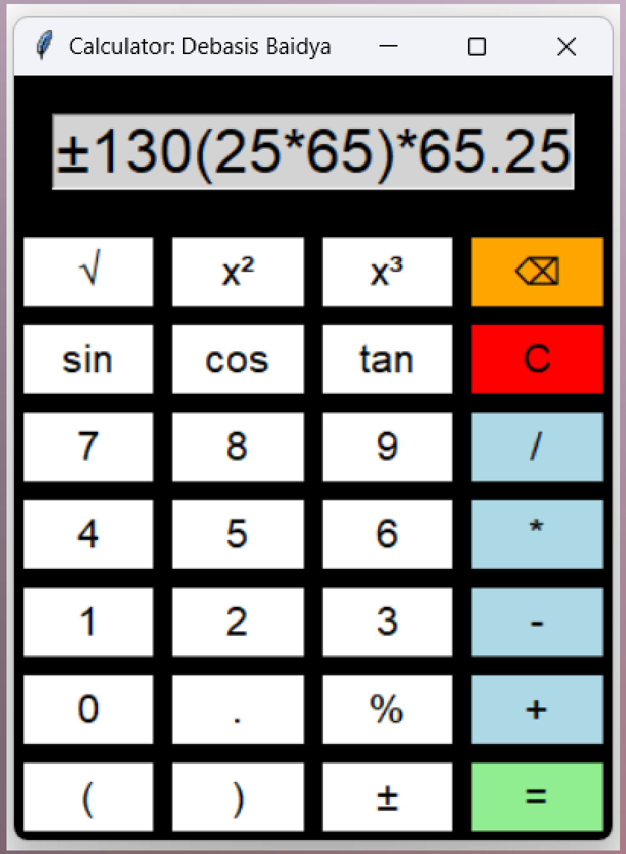
Manages all user inputs, from clearing the display to computing results.
Backspace & Clear Buttons work to handle typos.

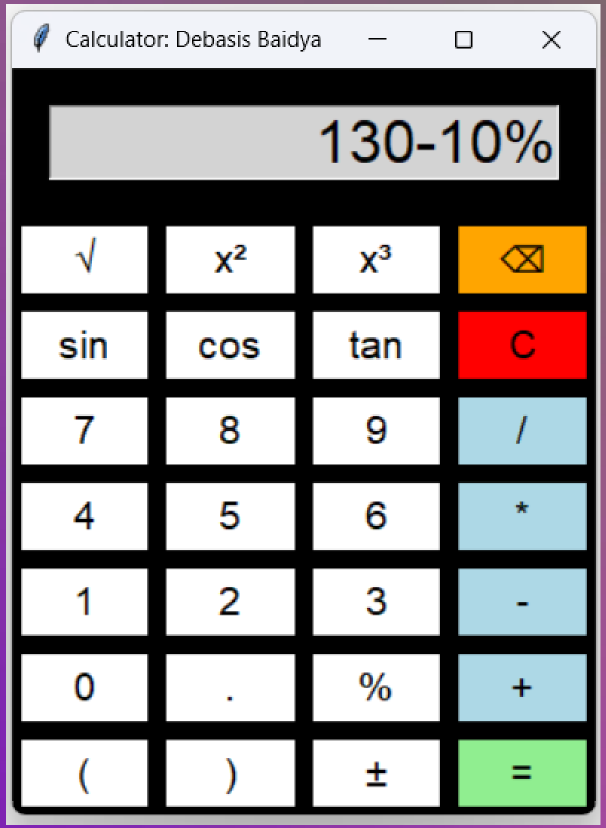
2 **Operations Dictionary**

Advanced operations are mapped in a dictionary for easy reference and execution.

3 **Error Handling & Button Layout**

Invalid inputs trigger an error message.
Button Layout positions buttons in the Calculator.





Key Features

Advanced Mathematical Operations:

Perform complex calculations with ease, including trigonometric functions (sin, cos, tan), Root (√), Square $(x)^2$ & Cube (x^3).

Percentage Calculations:

Effortlessly calculate percentages with the dedicated '%' button, supporting addition, subtraction, multiplication, and division operations.

Intuitive User Interface:

- Black background and colorful buttons
  - Clear typography
  - Organized button layout
  - Resizable design

Comprehensive Functionality:

- Basic arithmetic (+, -, *, /)
- Parentheses and negation ( ), (±)
- Decimal input (.)
- Dedicated buttons for common operations

Conclusion

1

Efficient Calculations

The calculator handles a variety of mathematical operations.

2

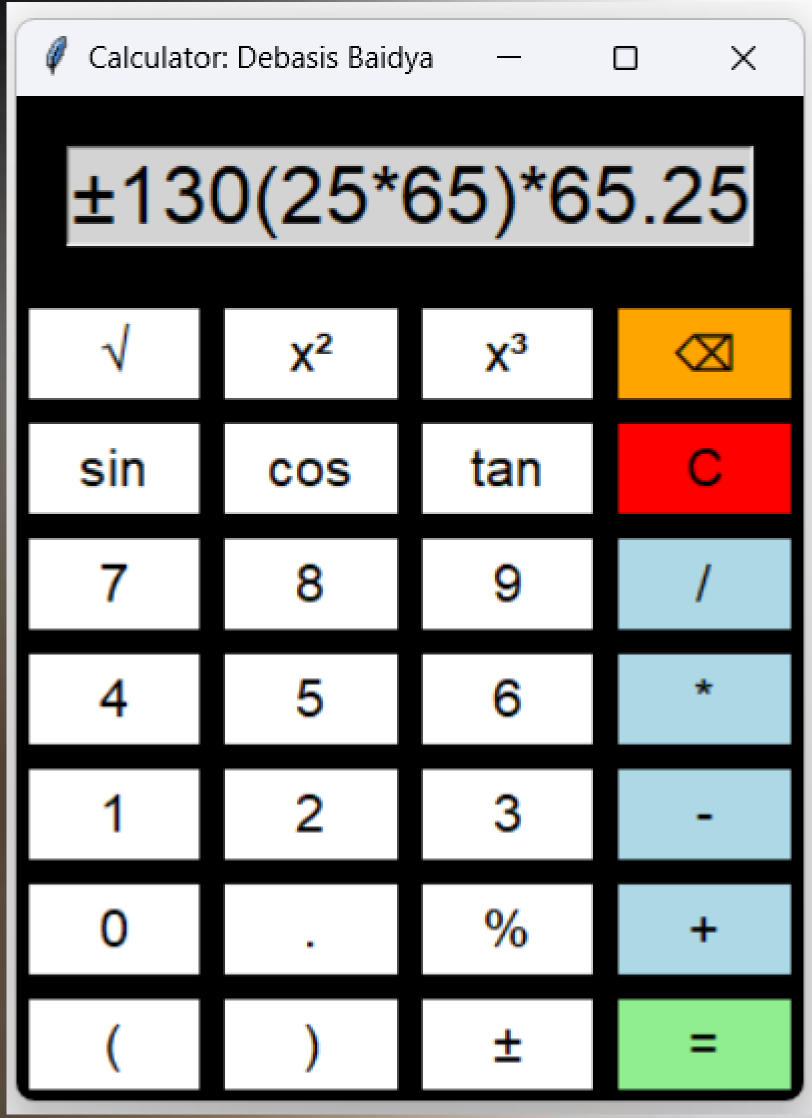
User-Friendly Design

The customized Tkinter interface provides a smooth user experience.

3

Robust Performance

The app is designed to be reliable and handle user errors.



Conclusion and Future Improvements

Summary

The key points of the calculator project were discussed.

Future Enhancements

Potential improvements include adding more advanced functions and refining the UI.

Q&A

The presentation is open for any questions & suggestions from the reviewer.