DEPT. Of Computer Science Engineering

SRM IST, Kattankulathur – 603 203

MINI PROJECT REPORT

DIGITAL CALCULATOR USING GUI

Advance Programming and Practice

[18CSC207J]

Submitted by

Pranay Jha [RA2111003011092]

Under the Guidance of

Dr. Vidhya M

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that **18CSC207J – ADVANCED PROGRAMMING PRACTICE project report** titled **" Digital Calculator Using GUI -Tkinter"** is the bonafide work of **Pranay Jha(RA2111003011092)** who undertook the task of completing the project within the allotted time.

Signature	**Signature**
LAB INCHARGE	**HEAD OF DEPARTMENT**
Dr. Vidhya M	Dr. M. Pushpalatha
Assistant Professor	Professor and Head
Department of Computing Technologies,	Computing Technologies
SRM Institute of Science and Technology	SRM Institute of Science and Technology

INDEX

ABSTRACT

When creating a calculator using the GUI Tkinter in Python, developers can design a wide range of user interfaces to suit different user preferences. For instance, the calculator's layout can be customized to suit the user's screen resolution, and the color scheme can be changed to match the user's preferences.

Additionally, developers can integrate different features into the calculator, such as memory functions, exponentials, trigonometric functions, and logarithmic functions.

Furthermore, the GUI Tkinter calculator can be used in education to help students learn math concepts and solve complex problems. Teachers can use this calculator as a tool for teaching different mathematical concepts, such as arithmetic, algebra, geometry, and calculus. It can help students learn how to use different mathematical functions, solve equations, and evaluate expressions quickly and efficiently. Moreover, the calculator can help students to visualize mathematical concepts, which can be challenging to understand through traditional teaching methods.

The GUI Tkinter calculator can also be used in finance and accounting to perform different financial calculations, such as loan payments, interest rates, and compound interest. The calculator can help accountants and financial analysts to perform complex calculations accurately and efficiently, reducing the likelihood of errors in their work.

Hence, the creation of a calculator using the GUI Tkinter in Python has revolutionized the way we perform mathematical calculations. With its userfriendly interface and powerful functions, it has become an essential tool in many fields, such as education, finance, and engineering. Its ability to simplify complex calculations and improve efficiency has made it a valuable asset to professionals and students alike.

INTRODUCTION

In today's world, technology has made our lives easier and more efficient in many ways. One such example is the use of calculators, which have become an essential tool for performing mathematical calculations quickly and accurately. In recent years, graphical user interface (GUI) technology has made significant progress, making it easier for developers to create user-friendly software applications.

A digital calculator is a tool that performs mathematical operations. It is an essential tool for any computer user, and it can also be used as a learning tool for students. In this tutorial, we will be using the GUI tkinter module in Python to create a simple digital calculator. The tkinter module provides a set of tools for creating graphical user interfaces (GUIs) in Python.

To create a digital calculator using tkinter, we will first create a basic GUI window. This window will contain a set of buttons that the user can click on to input numbers and perform mathematical operations. We will use the grid() method to position the buttons in the window.

Next, we will create a text box in which the user can see the input numbers and the result of the operations. We will use the Entry() widget to create the text box. Finally, we will write the code that performs the mathematical operations. We will define functions for each operation (addition, subtraction, multiplication, and division), and then call these functions based on the user input.

Once we have completed these steps, we will have a functional digital calculator with a simple GUI interface that can be used for basic mathematical operations.

One such development is the creation of a calculator using the GUI Tkinter in Python. This calculator offers a user-friendly interface that is easy to use and provides a wide range of mathematical functions. Using the Tkinter library in Python, developers can design and create a calculator with different themes and styles, allowing users to personalize their experience.

Moreover, the use of a GUI calculator made using Tkinter can be beneficial in many fields, such as education, finance, and engineering. It can be used by students, teachers, accountants, and engineers to perform complex calculations quickly and efficiently. This technology has made it possible to have a powerful calculator that is easy to use, with a wide range of functions and a modern, userfriendly interface.

PROBLEM EXPLANATION

While calculators are essential tools for performing mathematical calculations, some users may find them challenging to use, especially when dealing with complex equations. Traditional calculators may have limited functionality, which can make it difficult for users to perform advanced mathematical operations. Additionally, the user interfaces of traditional calculators can be difficult to navigate, which can slow down the calculation process.

Moreover, traditional calculators may not be suitable for different fields, such as finance, engineering, and science. These fields may require specialized calculations, which traditional calculators may not be able to perform.

Furthermore, traditional calculators may be expensive, and not all users may have access to them. This can limit the accessibility of this essential tool to students and professionals in different fields.

Thus, there is a need for a calculator that is user-friendly, provides a wide range of functions, is accessible, and can be customized to suit different fields. The creation of a calculator using the GUI Tkinter in Python provides a solution to these problems by offering a user-friendly interface, a wide range of functions, and customizable feature.

IMPLEMENTATION

Creating a digital calculator using GUI (Graphical User Interface) with tkinter in volves designing and programming a graphical user interface that mimics the loo k and feel of a traditional handheld calculator. The goal is to enable users to perf orm basic arithmetic operations such as addition, subtraction, multiplication, and division using an easy-to-use interface.

Here is a detailed explanation of how to create a digital calculator using tkinter:

1. Importing the tkinter module:

The first step is to import the tkinter module, which is used to create graphical u ser interfaces in Python. This is done by adding the following line of code at the beginning of the script:

`` import tkinter as

tk

```

2. Creating the main window:

The next step is to create the main window for the calculator. This is done using the `Tk()` function in tkinter:

``` root = tk.Tk()

root.title("Digital Calculator")

```
```

This creates a window with the title "Digital Calculator".

3. Adding the display widget:
The display widget is the area where the numbers and operators are displayed. In this case, we will use an Entry widget to create the display:

```
display = tk.Entry(root, width=30, borderwidth=5)
display.grid(row=0, column=0, columnspan=4, padx=10, pady=10)
```

This creates an Entry widget with a width of 30 characters and a border width of 5. The `grid()` function is used to position the widget in the window. It is placed in the first row and first column of the grid layout, spanning all four columns.

4. Creating the buttons:

The next step is to create the buttons for the calculator. We will create the button s for the digits and operators using a for loop:

```
button_texts = ['7', '8', '9', '/',
                '4', '5', '6', '*',
                '1', '2', '3', '-',
                '0', '.', '=', '+']

row_num = 1 col_num
= 0
```

for text in button_texts:

button = tk.Button(root, text=text, padx=10, pady=10, command=lambda text =text: button_click(text))

button.grid(row=row_num, column=col_num)

col_num += 1
if col_num > 3:

col_num = 0

row_num += 1

```

This creates 16 buttons, arranged in a 4x4 grid. The `button_texts` list contains t he text for each button. The `Button()` function is used to create each button. Th e `padx` and `pady` arguments are used to set the padding around the text. The ` command` argument is used to set the function that is called when the button is c licked. In this case, the `button_click()` function is called with the text of the but ton as an argument.

5. Defining the button_click() function:

The `button_click()` function is called when a button is clicked. It is responsible for updating the display with the clicked button's text. The function can be defin ed as follows:

``` def button_click(text):

current = display.get()

display.delete(0, tk.END)

display.insert(0, current + text)

```

This function gets the current contents of the display using the `get()` method. It then deletes the contents of the display using the `delete()` method and inserts th e new text using the `insert()` method.
```

6. Defining the calculate() function:

The `calculate()` function is responsible for performing the arithmetic operation when the "=" button is clicked. It can be defined as follows:

``` def calculate():

expression = display.get()

result =

## CODE

import tkinter as tk from

functools import partial

import math import re

normalcalc = True

window = tk.Tk()

window.title("Calculator")

window.geometry("300x400")

window.minsize(300, 400)

def button_click(expression, button):

if button == "C":

expression.set("") elif button ==

"CE":
```

```
        expression.set(expression.get()[:-1])
    elif button == "√x":
        expression.set(str(math.sqrt(eval(expression.get()))))
    elif button == "x^2":
        expression.set(str(eval(expression.get()) ** 2))
    elif button == "1/x":
        expression.set(str(1 / eval(expression.get())))
    elif button == "±":         if expression.get()[0] ==
    "-":
            expression.set(expression.get()[1:])
        else:
            expression.set("-" + expression.get())
    elif button == "sin":
        expression.set(str(math.sin(math.radians(float(expression.get())))))
    elif button == "cos":
        expression.set(str(math.cos(math.radians((float(expression.get()))))))
    elif button == "tan":
        expression.set(str(math.tan(math.radians((float(expression.get()))))))
    elif button == "sec":
        expression.set(str(1 / math.cos(math.radians((float(expression.get()))))))
    elif button == "csc":
```

```
        expression.set(str(1 / math.sin(math.radians((float(expression.get()))))))
    elif button == "cot":
        expression.set(str(1 / math.tan(math.radians((float(expression.get()))))))
    elif button == "x^3":
        expression.set((float(expression.get()) ** 3))
    elif button == "e^x":
    expression.set(str(math.exp(float(expression.get()))))
    elif button == "log":
        expression.set(str(math.log10(float(expression.get()))))
    elif button == "ln":
        expression.set(str(math.log(float(expression.get()))))
    elif button == "π":
        expression.set(str(math.pi))
    elif button == "!":
        expression.set(str(math.factorial(float(expression.get()))))
    elif button == "e":
        expression.set(str(math.e))
    elif button == "10^x":
        expression.set(str(10 ** float(expression.get())))
    elif button == "=" :
```

```
        express = expression.get()         express =
re.sub(r"\b0+(\d)", r"\1", express)
        try:
expression.set(str(eval(express)))
except:
            expression.set("Error")
else:
        expression.set(expression.get() + str(button))

    if len(expression.get()) > 15:
        display.config(font=('Arial', 15))
else:
        display.config(font=('Arial', 25))


    display.icursor(tk.END)




def on_key_press(event):     if
len(expression.get()) > 15:
```

```
        display.config(font=('Arial', 15))
    else:
        display.config(font=('Arial', 25))


def on_resize(event):     width = event.width     if
len(expression.get()) > 15 and width <= 300:
        display.config(font=('Arial', 15))
else:
        display.config(font=('Arial', 25)) window.bind("<Configure>", on_resize)



def validate_input(key):
    allowed_keys = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '.', '+', '-', '*', '/']
    if key in allowed_keys:
        return True
elif key == '\b':
return True     else:
        return False


expression = tk.StringVar()
expression.set("")
```

```
display = tk.Entry(window, textvariable=expression, font=('Arial', 25), justify="
right", validate="key", validatecommand=(window.register(validate_input), '%S
'))

display.pack(fill=tk.BOTH, expand=True)

display.bind('<Key>', on_key_press) buttonFrame =

tk.Frame(window, bg="#F7F5E3")

buttonFrame.pack(fill=tk.BOTH, expand=True)


scFrame = tk.Frame(window, bg="#F7F5E3")


buttons = ["1/x", "x^2", "√x", "/",
    "7", "8", "9", "*",
    "4", "5", "6", "-",
    "1", "2", "3", "+",
    "0", ".", "±", "=",
    "C", "%", "<-"
]


scbutton = ["sin", "cos", "tan", "sec", "csc", "cot", "x^3", "e^x", "log", "ln", "π",
"!", "(", ")", "e", "10^x"]
```

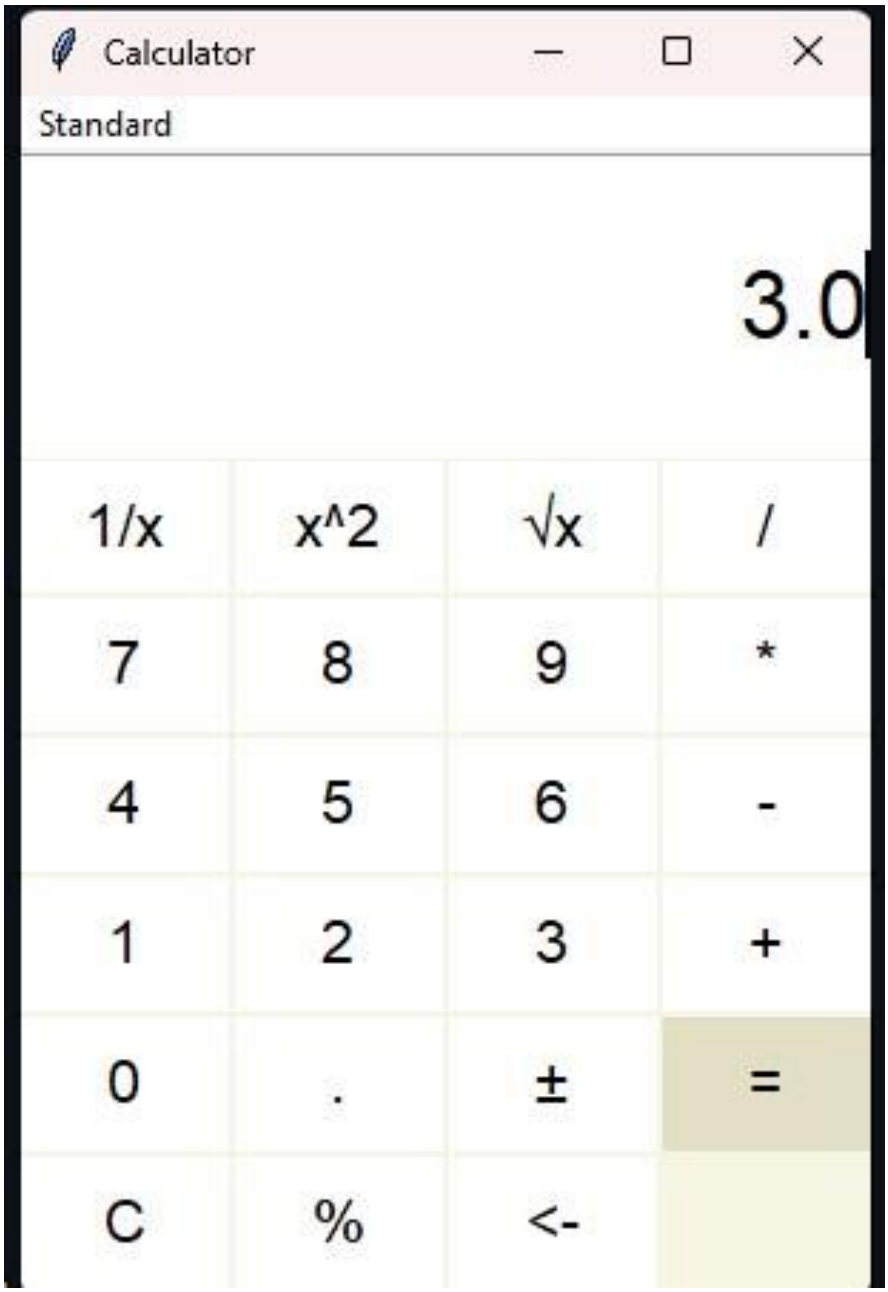
```
def on_enter(e):
    e.widget['bg'] = '#E2DFC5'

def on_leave(e):
    e.widget['bg'] = 'white'

def create_buttons(frame, buttons, mode):
    row, col = 0, 0     for
button in buttons:
        command = partial(mode, expression, button)        btn = tk.Button(frame,
text=button, font=('Arial', 16), width=5, height=0, re lief='ridge', bg="white",
borderwidth=0, highlightthickness=0, command=comm and)
btn.grid(row=row, column=col, padx=1, pady=1, sticky=tk.NSEW)

        btn.bind("<Enter>", on_enter)
btn.bind("<Leave>", on_leave)

        if button.isdigit():          btn.bind("<Key-{}>".format(button), lambda event:
mode(expression, bu tton))
```

```
        display.bind("<Return>", lambda event: mode(expression, "="))
display.focus_set()


        col += 1
if col > 3:
col = 0
row += 1
create_buttons(b
uttonFrame,
buttons,
button_click)


def sc_click():
   global normalcalc     if normalcalc:         print("sc")
buttonFrame.pack_forget()
create_buttons(scFrame, scbutton, button_click)
scFrame.pack(fill=tk.BOTH, expand=True)
buttonFrame.pack(fill=tk.BOTH, expand=True)
create_buttons(buttonFrame, buttons, button_click)
normalcalc = False     else:
```

```
        scFrame.pack_forget()
buttonFrame.pack(fill=tk.BOTH, expand=True)
create_buttons(buttonFrame, buttons, button_click)
normalcalc = True

for i in range(6):
    buttonFrame.rowconfigure(i, weight=1) for
i in range(5):
    scFrame.rowconfigure(i, weight=1)

for i in range(4):
    scFrame.columnconfigure(i, weight=1)
buttonFrame.columnconfigure(i, weight=1)


menubar = tk.Menu(window, font=('Arial', 55)) mode =
tk.Menu(menubar, tearoff=0)
mode.add_checkbutton(label="Scientific", command=sc_click)
menubar.add_cascade(label="Standard", menu=mode)
window.config(menu=menubar)
```

```
window.mainloop()
```

OUTPUT (Digital Calculator)



CONCLUSION

In conclusion, the creation of a calculator using the GUI Tkinter in Python has revolutionized the way we perform mathematical calculations. This technology offers a wide range of functions, a user-friendly interface, and customizable features, making it suitable for different fields, such as education, finance, and engineering. The calculator's ability to simplify complex calculations and improve efficiency has made it a valuable asset to professionals and students alike.

The GUI Tkinter calculator provides an excellent example of how technology can simplify our daily tasks and improve our efficiency. It has made performing mathematical calculations more accessible, and its customizable features make it a versatile tool for different fields. Overall, the GUI Tkinter calculator is an essential tool that has helped users solve complex equations, reduce errors in their work, and improve their productivity.

REFRENCE

1. "How to Create a Calculator Using Tkinter in Python" by Alexander Gorman: This tutorial provides a step-by-step guide to building a simple calculator using tkinter. It covers topics such as creating buttons, using event handlers, and displaying results. Link: https://realpython.com/python-gui-tkinter/#how-to-create-a-calculatorusing-tkinter-in-python

2. "Python GUI Calculator Tutorial" by Tech With Tim: This YouTube video tutorial covers the creation of a more advanced calculator with a graphical user interface using tkinter. It covers topics such as creating a display, handling button presses, and performing calculations. Link: https://www.youtube.com/watch?v=QYUBz4MRyos

3. "Creating a GUI Calculator in Python" by CodersLegacy: This tutorial covers the creation of a simple calculator using tkinter. It includes code snippets and explanations for creating buttons, using event handlers, and performing calculations. Link: https://www.coderslegacy.com/python/guicalculator-in-python/

4. "How to Build a Calculator with Python and Tkinter" by The Python Jedi: This tutorial provides a step-by-step guide to building a basic calculator using tkinter. It covers topics such as creating buttons, using event handlers, and handling errors. Link: https://thepythonjedi.com/how-tobuild-a-calculator-with-python-and-tkinter/