

CST1500

Computer Systems Architecture and Operating Systems

Bash Project

By

Shayan Ahmed(M00983522), Fawaaz Shaikh(M00909151)

Code

*import* tkinter *as* tk

*import* math

*import* threading

*class* Calculator:

*def* \_\_init\_\_(self, master):

*self*.master *=* master

        master.title("Scientific Calculator")

*# Configure background color of the root window*

        master.config(bg*=*"black")

*# Create and configure the display entry widget*

*self*.display *=* tk.Entry(master, width*=*30, borderwidth*=*5, bg*=*"black", fg*=*"white")

*self*.display.grid(row*=*0, column*=*0, columnspan*=*5, padx*=*10, pady*=*10)

*# Define buttons and their layout (text, row, column)*

*self*.buttons *=* [

            ('7', 1, 0), ('8', 1, 1), ('9', 1, 2), ('/', 1, 3), ('C', 1, 4),

            ('4', 2, 0), ('5', 2, 1), ('6', 2, 2), ('\*', 2, 3), ('(', 2, 4),

            ('1', 3, 0), ('2', 3, 1), ('3', 3, 2), ('-', 3, 3), (')', 3, 4),

            ('0', 4, 0), ('.', 4, 1), ('=', 4, 2), ('+', 4, 3), ('π', 4, 4),

            ('sin', 5, 0), ('cos', 5, 1), ('tan', 5, 2), ('^', 5, 3), ('√', 5, 4)

        ]

*# Create buttons and assign commands*

*for* (text, row, col) *in* *self*.buttons:

            tk.Button(master, text*=*text, command*=lambda* t*=*text: *self*.on\_button\_click(t), borderwidth*=*5, bg*=*"black", fg*=*"white").grid(row*=*row, column*=*col, sticky*=*"we")

*# Create the "Clear all" button*

        tk.Button(master, text*=*"Clear all", command*=lambda* t*=*"Clear all": *self*.on\_button\_click(t), borderwidth*=*5, bg*=*"black", fg*=*"white").grid(row*=*6, column*=*0, sticky*=*"we", columnspan*=*5)

*def* *on\_button\_click*(self, text):

*# Function to handle button clicks*

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

*# Evaluate the expression when '=' is clicked*

*try*:

                expression *=* *self*.display.get()

                result *=* eval(expression)

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, str(result))

*except*:

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, "Error")

*elif* text *==* 'C':

*# Clear the last entered value when 'C' is clicked*

            e*=self*.display.get()

            e *=* e[0:len(e) *-* 1]  *# deleting the last entered value*

*self*.display.delete(0, "end")

*self*.display.insert(0, e)

*elif* text *==* 'Clear all':

*# Clear the display when 'Clear all' is clicked*

*self*.display.delete(0, tk.END)

*elif* text *==* 'π':

*# Insert pi value when 'π' is clicked*

*self*.display.insert(tk.END, str(math.pi))

*elif* text *==* '√':

*# Calculate square root when '√' is clicked*

*try*:

                expression *=* *self*.display.get()

                result *=* math.sqrt(float(expression))

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, str(result))

*except*:

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, "Error")

*elif* text *in* ['sin', 'cos', 'tan']:

*# Calculate trigonometric functions when 'sin', 'cos', or 'tan' is clicked*

*try*:

                expression *=* *f*'math.{text}(math.radians({*self*.display.get()}))'

                result *=* eval(expression)

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, str(result))

*except*:

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, "Error")

*elif* text *==* '^':

*# Calculate power when '^' is clicked*

*try*:

                expression *=* *self*.display.get()

                result *=* int(expression[0]) *\*\** int(expression[1])

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, str(result))

*except*:

*self*.display.delete(0, tk.END)

*self*.display.insert(tk.END, "Error")

*else*:

*# Insert the clicked button's text into the display*

*self*.display.insert(tk.END, text)

*def* *run\_calculator*():

    root *=* tk.Tk()

    calculator *=* Calculator(root)

    root.mainloop()

*# Create a thread for running the calculator*

calculator\_thread *=* threading.Thread(target*=*run\_calculator)

calculator\_thread.start()

*#calculator\_thread1 = threading.Thread(target=run\_calculator)*

*#calculator\_thread1.start()*

Output

A screenshot of a computer

Description automatically generated

Self-report

Report by Shayan: Through building this scientific calculator project, I attained practical knowledge in GUI development using Tkinter, comprehending how to design and configure widgets for user interaction. I deepened my understanding of event driven programming by employing callback functions to handle button clicks and other events, facilitating smooth interaction with the interface. Additionally, I honed my skills in error handling, ensuring the calculator provides informative feedback in case of wrong inputs or calculations. Exploring threading for running the calculator in a separate thread enhanced the responsiveness of the GUI, improving the overall user experience. Overall, this project provided a comprehensive learning experience encompassing GUI development, event-driven programming, error handling, and threading, equipping me with valuable skills.

Report by Fawaz: I completed my Python calculator project recently and wanted to share my experience. First off, I outlined what I wanted the calculator to do: add, subtract, multiply, and divide. While coding, I faced some challenges, like handling non-numeric inputs. I solved them by searching online and asking seniors for help. I also tested the calculator thoroughly to ensure it worked with different numbers and operations. Overall, this project taught me a lot about coding, problem-solving, and perseverance. It was a good challenge, and I'm happy with how it turned out. Thanks for giving me this project.