Final project

Github link: <https://github.com/DurinAnash/Final-project->

This Python program creates a graphical user interface (GUI) using tkinter for solving algebraic equations with a single variable (x). It leverages the Pillow library to parse, manipulate, and solve equations, displaying the solution and solving steps in a separate window. For now, I have been able to execute a version of the calculator that is able to solve equations from simpler to for more complex.

"""

Advanced Calculator Application

This module implements a graphical calculator using Tkinter, supporting basic arithmetic,

mathematical functions (e.g., sqrt, log, sin, cos, tan), and a history feature to track calculations.

"""

import tkinter as tk

from tkinter import messagebox, Toplevel

import re

import math

from PIL import Image, ImageTk # For loading and displaying images

class CalculatorApp:

"""A Tkinter-based advanced calculator application with history and error handling.

The calculator supports arithmetic operations, mathematical functions, and a history window

to review past calculations. It includes accessibility features and image-based buttons.

"""

def \_\_init\_\_(self, root: tk.Tk):

"""Initialize the calculator application.

Args:

root (tk.Tk): The root Tkinter window.

Attributes:

root (tk.Tk): The main application window.

history (list): List of strings storing past calculations.

calc\_icon (ImageTk.PhotoImage): Icon for the calculate button.

history\_icon (ImageTk.PhotoImage): Icon for the history button.

entry\_equation (tk.Entry): Entry widget for user input.

label\_result (tk.Label): Label widget to display calculation results.

Raises:

Exception: If image files (calculator.png, history.png) cannot be loaded.

"""

self.root = root

self.root.title("Advanced Calculator")

self.root.geometry("400x600")

self.history = []

# Load button icons with error handling

try:

self.calc\_icon = ImageTk.PhotoImage(Image.open("calculator.png").resize((50, 50)))

self.history\_icon = ImageTk.PhotoImage(Image.open("history.png").resize((50, 50)))

except Exception as e:

messagebox.showerror("Error", f"Failed to load images: {e}")

self.calc\_icon = None

self.history\_icon = None

self.create\_main\_window()

def create\_main\_window(self):

"""Create and configure the main calculator window.

Sets up the main frame, labels, entry widget, result display, and buttons for

calculation, history, and exit. Includes accessibility features for button images.

"""

# Main frame for layout

main\_frame = tk.Frame(self.root, padx=10, pady=10)

main\_frame.pack(expand=True, fill="both")

# Title and input/result labels

tk.Label(main\_frame, text="Advanced Calculator", font=("Arial", 16, "bold")).pack(pady=5)

tk.Label(main\_frame, text="Enter Equation:", font=("Arial", 12)).pack(anchor="w")

tk.Label(main\_frame, text="Result:", font=("Arial", 12)).pack(anchor="w")

# Entry widget for user equation input

self.entry\_equation = tk.Entry(main\_frame, width=30, font=("Arial", 12))

self.entry\_equation.pack(pady=5)

# Label to display calculation result

self.label\_result = tk.Label(main\_frame, text="0", font=("Arial", 12), relief="sunken", width=30)

self.label\_result.pack(pady=5)

# Frame for action buttons

button\_frame = tk.Frame(main\_frame)

button\_frame.pack(pady=10)

# Calculate, History, and Exit buttons with icons

tk.Button(

button\_frame,

text="Calculate",

image=self.calc\_icon,

compound="left",

command=self.calculate,

font=("Arial", 10)

).pack(side="left", padx=5)

tk.Button(

button\_frame,

text="History",

image=self.history\_icon,

compound="left",

command=self.show\_history,

font=("Arial", 10)

).pack(side="left", padx=5)

tk.Button(

button\_frame,

text="Exit",

command=self.exit\_app,

font=("Arial", 10)

).pack(side="left", padx=5)

# Accessibility: Add alternate text for image buttons

self.root.option\_add("\*Button\*image\*calc\_icon\*alt", "Calculator Icon")

self.root.option\_add("\*Button\*image\*history\_icon\*alt", "History Icon")

def create\_history\_window(self):

"""Create and display the calculation history window.

Opens a new Toplevel window showing past calculations and a button to return

to the main calculator.

"""

history\_window = Toplevel(self.root)

history\_window.title("Calculation History")

history\_window.geometry("300x400")

tk.Label(history\_window, text="Calculation History", font=("Arial", 14, "bold")).pack(pady=10)

# Text widget to display history (read-only)

history\_text = tk.Text(history\_window, height=15, width=35, font=("Arial", 10))

history\_text.pack(pady=5)

# Populate history entries

for entry in self.history:

history\_text.insert(tk.END, f"{entry}\n")

history\_text.config(state="disabled")

# Button to close history window

tk.Button(history\_window, text="Back to Calculator", command=history\_window.destroy, font=("Arial", 10)).pack(pady=10)

def validate\_input(self, equation: str) -> bool:

"""Validate the user-provided equation.

Checks if the equation is non-empty and contains only allowed characters

(digits, operators, parentheses, and math functions).

Args:

equation (str): The user-entered equation string.

Returns:

bool: True if the equation is valid, False otherwise.

Displays an error messagebox if validation fails.

"""

if not equation.strip():

messagebox.showerror("Input Error", "Equation cannot be empty.")

return False

# Regular expression to allow digits, operators, parentheses, and math functions

allowed\_pattern = r'^[\d\s+\-\*/().^%sqrtlogsincoStanpi]+$'

if not re.match(allowed\_pattern, equation.replace(" ", "")):

messagebox.showerror("Input Error", "Invalid characters in equation.")

return False

return True

def calculate(self):

"""Evaluate the user-entered equation and display the result.

Replaces math function names with Python equivalents, evaluates the equation,

and updates the result label. Stores valid calculations in history.

Raises:

Exception: If the equation is invalid or evaluation fails.

"""

equation = self.entry\_equation.get()

if not self.validate\_input(equation):

return

try:

# Replace math function names with Python math module equivalents

equation = equation.replace("sqrt", "math.sqrt")

equation = equation.replace("log", "math.log")

equation = equation.replace("sin", "math.sin")

equation = equation.replace("cos", "math.cos")

equation = equation.replace("tan", "math.tan")

equation = equation.replace("pi", str(math.pi))

# Evaluate the equation safely

result = eval(equation, {"math": math})

self.label\_result.config(text=f"{result:.4f}")

self.history.append(f"{equation} = {result:.4f}")

except Exception as e:

messagebox.showerror("Calculation Error", f"Invalid equation: {e}")

self.label\_result.config(text="Error")

def show\_history(self):

"""Display the calculation history window.

Calls create\_history\_window to open a new Toplevel window with past calculations.

"""

self.create\_history\_window()

def exit\_app(self):

"""Exit the application after user confirmation.

Prompts the user to confirm exit and destroys the root window if confirmed.

"""

if messagebox.askyesno("Exit", "Are you sure you want to exit?"):

self.root.destroy()

def main():

"""Entry point for the calculator application.

Initializes the Tkinter root window and starts the main event loop.

"""

root = tk.Tk()

app = CalculatorApp(root)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()