Laboratory Activity 6 - GUI Design: Layout and Styling

Name: Delavin, Katarina Nicole R.	Date : 11/08/24
Section: CPE21S1	Instructor: Engr. Maria Rizette Sayo

Grid Layout

Code:

```
import tkinter as tk
from tkinter import messagebox
Fat


def login():
   username = entry_username.get()
   password = entry_password.get()


   correct_username = "katarina"
   correct_password = "pink"

   if username == correct_username and password == correct_password:
       messagebox.showinfo("Login Successful", "Welcome!")
   else:
       messagebox.showerror("Login Failed", "Invalid username or password.")


def create_login_gui(width=300, height=200, bg_color="#F4C2C2"):

   window = tk.Tk()
   window.title("Login System")
   window.geometry(f"{width}x{height}")
   window.configure(bg=bg_color)


   lbl_username = tk.Label(window, text="Username:", bg=bg_color)
   lbl_username.pack(pady=10)
   global entry_username
   entry_username = tk.Entry(window)
   entry_username.pack(pady=5)


   lbl_password = tk.Label(window, text="Password:", bg=bg_color)
```

```
    lbl_password.pack(pady=5)
    global entry_password
    entry_password = tk.Entry(window, show="*")
    entry_password.pack(pady=5)


    btn_login = tk.Button(window, text="Login", command=login)
    btn_login.pack(pady=20)


    window.mainloop()


create_login_gui(width=400, height=300, bg_color="#F4C2C2")
```

Output:

Observation: When you run the application, you'll see a neatly organized layout with a "Text: " label and input field aligned horizontally at the top, followed by a "Password: " label and input field directly beneath it. The "Register" button is positioned to the right of the password input field, creating a clean and intuitive user interface for entering credentials. The components should be well-spaced, providing a clear and user-friendly experience.

Grid Layout using Loops

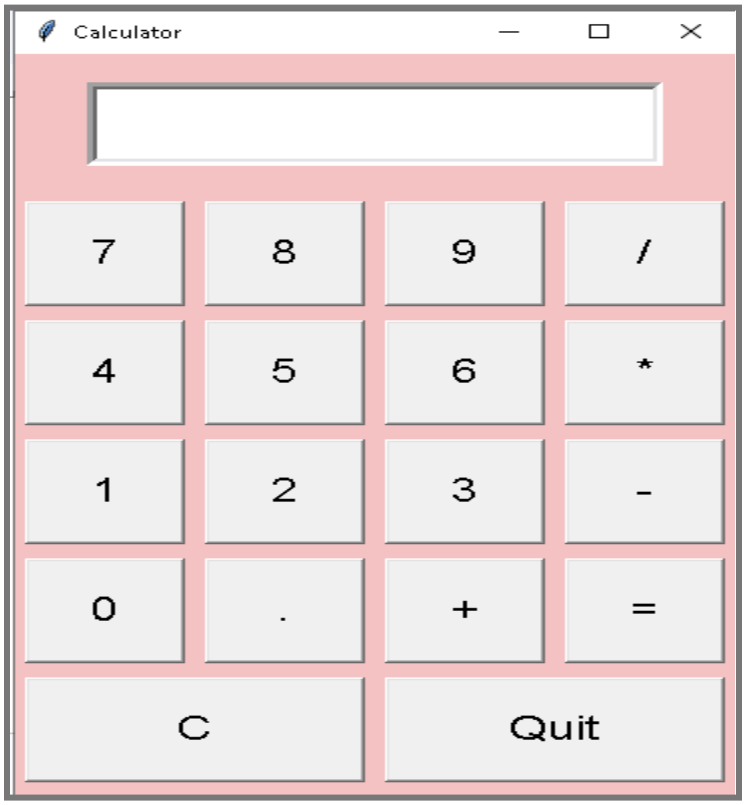
Code:

```
import tkinter as tk


def press(key):
    expression = entry.get()
    entry.delete(0, tk.END)
    entry.insert(tk.END, expression + key)
```

```
def calculate():
    try:
        result = str(eval(entry.get()))
        entry.delete(0, tk.END)
        entry.insert(tk.END, result)
    except Exception as e:
        entry.delete(0, tk.END)
        entry.insert(tk.END, "Error")


def clear():
    entry.delete(0, tk.END)


window = tk.Tk()
window.title("Calculator")
window.geometry("400x500")
window.configure(bg="#F4C2C2")


entry = tk.Entry(window, font=("Arial", 18), bd=5, relief=tk.SUNKEN,
justify="right")
entry.grid(row=0, column=0, columnspan=4, ipadx=8, ipady=10, pady=20)


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


for (text, row, col) in buttons:
    if text == "=":
        btn = tk.Button(window, text=text, font=("Arial", 18),
command=calculate, width=5, height=2)
    else:
        btn = tk.Button(window, text=text, font=("Arial", 18), command=lambda
key=text: press(key), width=5, height=2)
    btn.grid(row=row, column=col, padx=5, pady=5)


btn_clear = tk.Button(window, text="C", font=("Arial", 18), command=clear,
width=5, height=2)
btn_clear.grid(row=5, column=0, columnspan=2, padx=5, pady=5, sticky="we")
```

```
btn_quit = tk.Button(window, text="Quit", font=("Arial", 18),
command=window.quit, width=5, height=2)
btn_quit.grid(row=5, column=2, columnspan=2, padx=5, pady=5, sticky="we")


window.mainloop()
```

Output:



Observation:

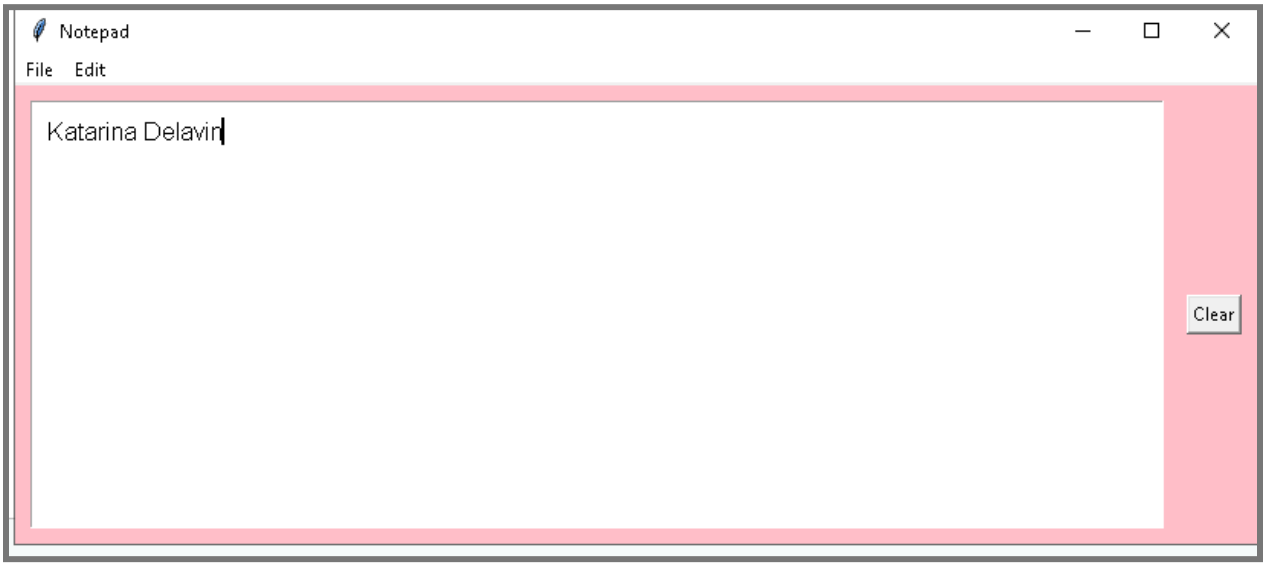
A "Grid Layout" window with a text input box at the top and a grid of buttons below it displays when you launch the application. Along with operation symbols like '+', ', ***, and T, the buttons display the numerals 0 through 9. It is easy to use as a basic calculator interface because each button is positioned properly. It is easy to use and has a clear layout that makes interaction simple.

Vbox and Hbox layout managers (Simple Notepad)

Code:

```
import tkinter as tk
from tkinter import Menu


def clear_text():
   text_area.delete(1.0, tk.END)


window = tk.Tk()
window.title("Notepad")
window.geometry("500x300")
window.configure(bg="#FFC0CB")  # Baby pink background for the window


menu_bar = Menu(window)
file_menu = Menu(menu_bar, tearoff=0)
file_menu.add_command(label="New")
file_menu.add_command(label="Open")
file_menu.add_command(label="Save")
file_menu.add_separator()
file_menu.add_command(label="Exit", command=window.quit)
menu_bar.add_cascade(label="File", menu=file_menu)

edit_menu = Menu(menu_bar, tearoff=0)
edit_menu.add_command(label="Undo")
edit_menu.add_command(label="Redo")
menu_bar.add_cascade(label="Edit", menu=edit_menu)

window.config(menu=menu_bar)


text_area = tk.Text(window, wrap="word", font=("Arial", 12), padx=10, pady=10)
text_area.pack(expand=True, fill="both", side="left", padx=(10, 5), pady=10)


clear_button = tk.Button(window, text="Clear", command=clear_text)
clear_button.pack(side="right", padx=10, pady=10)


window.mainloop()
```

Ouput; :



Observation: The GUI displays a basic Notepad user interface. Text can be opened, saved, and cleared using the "File" and "Edit" options in the menu bar. The "Clear" button effectively eliminates text, and the text field is roomy for input.

Supplementary Activity:

Code:

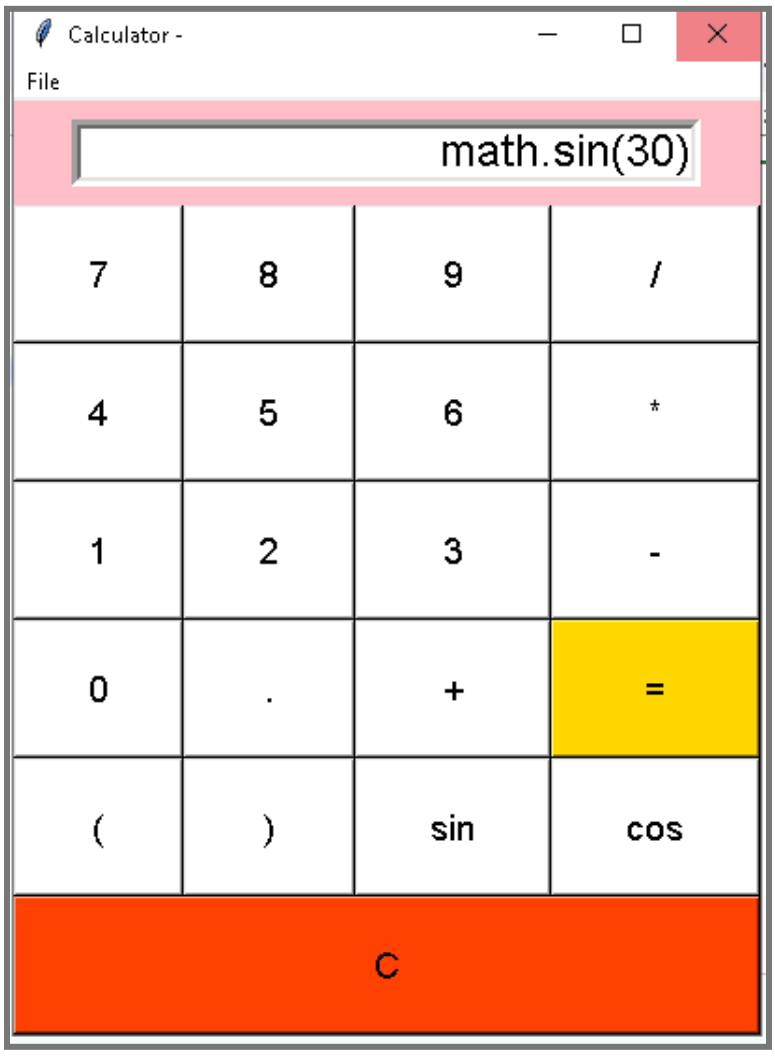
```
import tkinter as tk
from tkinter import Menu, messagebox, Toplevel
import math


def button_click(value):
    current_text = entry.get()
    entry.delete(0, tk.END)
    entry.insert(tk.END, current_text + str(value))


def clear():
    entry.delete(0, tk.END)


def calculate():
    try:
        result = eval(entry.get())
```

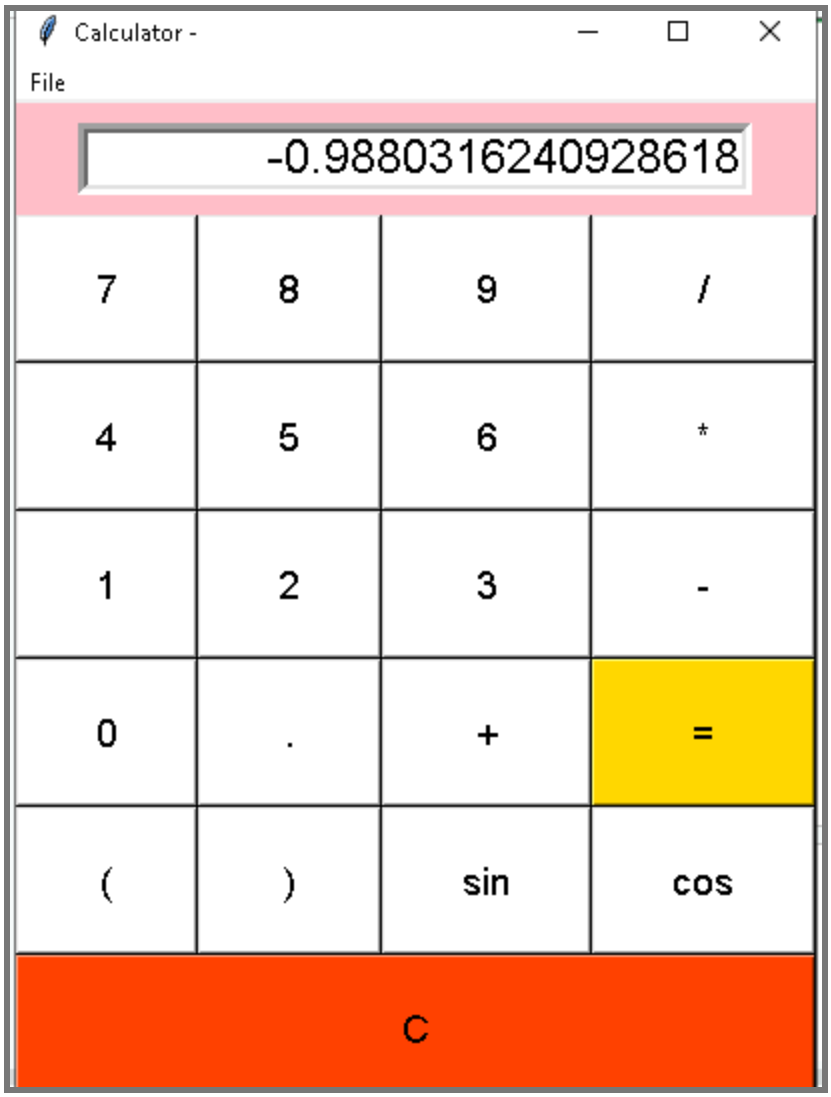
```
        entry.delete(0, tk.END)
        entry.insert(tk.END, str(result))
        return result
    except:
        entry.delete(0, tk.END)
        entry.insert(tk.END, "Error")
        return None


def save_calculations():
    calculation = entry.get()
    result = calculate()
    if result is not None:
        try:
            with open("calculations_history.txt", "a") as file:
                # Write in the format: math.sin(30) = -0.9880316240928618
                file.write(f"{calculation} = {result}\n")
            messagebox.showinfo("Saved", "Calculations have been saved
successfully!")
        except:
            messagebox.showerror("Error", "Failed to save calculations.")


def show_history():
    history_window = Toplevel(window)
    history_window.title("Calculation History")
    history_window.geometry("400x300")
    history_window.configure(bg="#FFC0CB")

    try:
        with open("calculations_history.txt", "r") as file:
            history_text = file.read()
    except FileNotFoundError:
        history_text = "No history available."


    history_display = tk.Text(history_window, wrap="word", font=("Courier New",
10), bg="#FFC0CB", fg="black")
    history_display.insert(tk.END, history_text)
    history_display.config(state="disabled")  # Make it read-only
    history_display.pack(expand=True, fill="both", padx=10, pady=10)


window = tk.Tk()
window.title("Calculator - ")
window.geometry("400x500")
window.configure(bg="#FFC0CB")
```

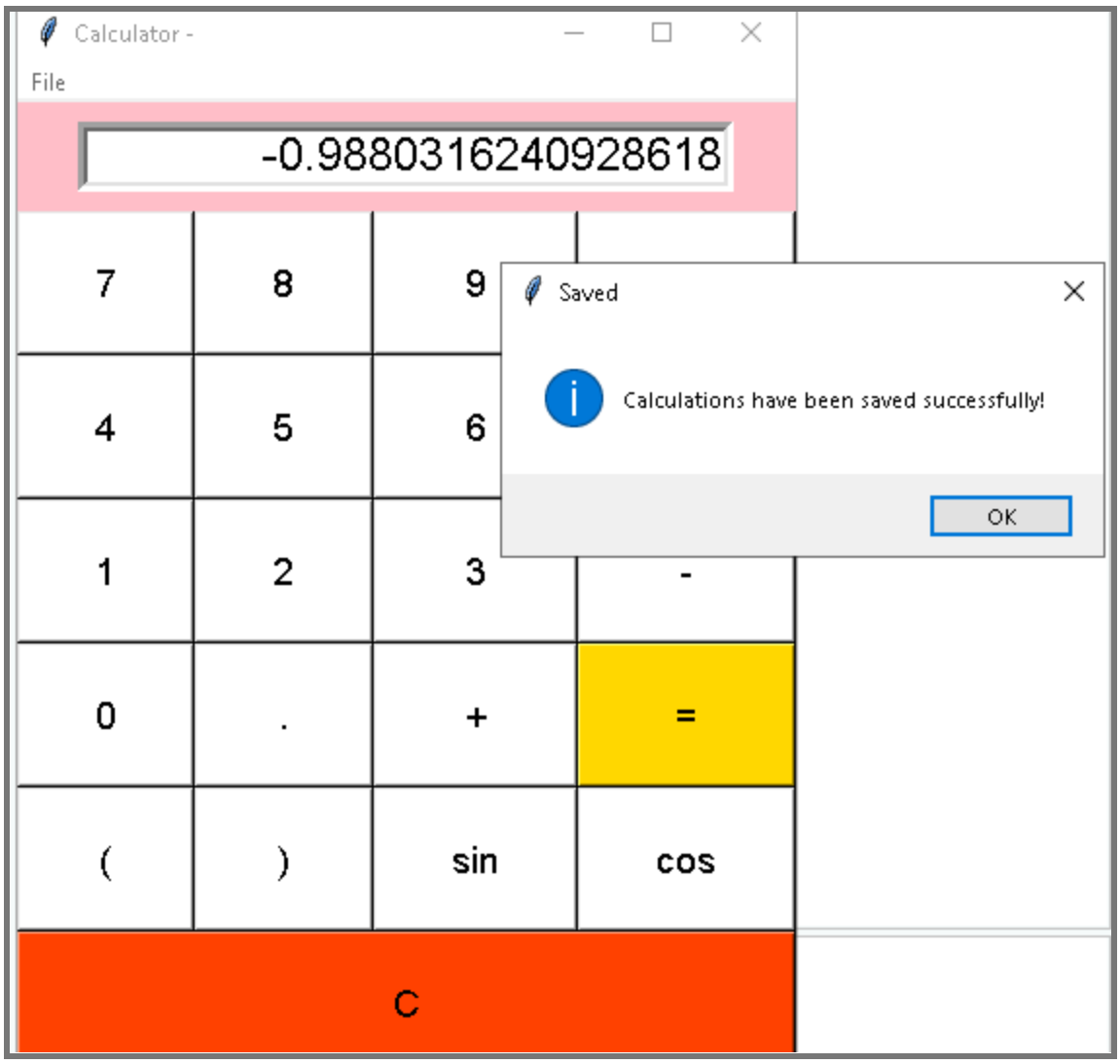
```
menu_bar = Menu(window)
file_menu = Menu(menu_bar, tearoff=0)
file_menu.add_command(label="Save", command=save_calculations)
file_menu.add_command(label="History", command=show_history)
file_menu.add_separator()
file_menu.add_command(label="Exit", command=window.quit)
menu_bar.add_cascade(label="File", menu=file_menu)
window.config(menu=menu_bar)


entry = tk.Entry(window, width=25, font=("Arial", 18), bd=5, insertwidth=2,
justify="right")
entry.grid(row=0, column=0, columnspan=4, pady=10)


buttons = [
   ("7", 1, 0), ("8", 1, 1), ("9", 1, 2), ("/", 1, 3),
   ("4", 2, 0), ("5", 2, 1), ("6", 2, 2), ("*", 2, 3),
   ("1", 3, 0), ("2", 3, 1), ("3", 3, 2), ("-", 3, 3),
   ("0", 4, 0), (".", 4, 1), ("+", 4, 2), ("=", 4, 3),
   ("(", 5, 0), (")", 5, 1), ("sin", 5, 2), ("cos", 5, 3)
]


for (text, row, col) in buttons:
   if text == "=":
       button = tk.Button(window, text=text, padx=20, pady=20, font=("Arial",
14),
                          bg="#FFD700", command=calculate)
   elif text == "sin":
       button = tk.Button(window, text=text, padx=20, pady=20, font=("Arial",
14),
                          bg="white", command=lambda:
button_click("math.sin("))
   elif text == "cos":
       button = tk.Button(window, text=text, padx=20, pady=20, font=("Arial",
14),
                          bg="white", command=lambda:
button_click("math.cos("))
   else:
       button = tk.Button(window, text=text, padx=20, pady=20, font=("Arial",
14),
                          bg="white", command=lambda t=text: button_click(t))
   button.grid(row=row, column=col, sticky="nsew")


btn_clear = tk.Button(window, text="C", padx=20, pady=20, font=("Arial", 14),
                      bg="#FF4500", command=clear)
btn_clear.grid(row=6, column=0, columnspan=4, sticky="nsew")
```

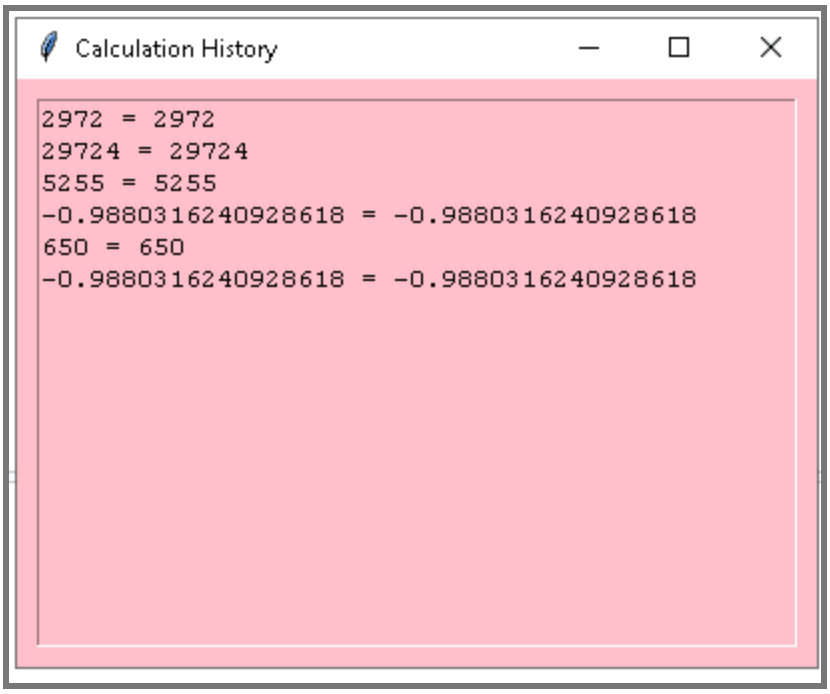
```
for i in range(7):
   window.grid_rowconfigure(i, weight=1)
for i in range(4):
   window.grid_columnconfigure(i, weight=1)


window.mainloop()
```

Output:




Calculator -
File
-0.9880316240928618
7
8
9
/
4
5
6
*
1
2
3
-
0
.
+
=
(
)
sin
cos
C


Calculator -
File
-0.9880316240928618
7
8
9
4
5
6
1
2
3
-
0
.
+
=
(
)
sin
cos
C
Saved
Calculations have been saved successfully!
OK



Conclusion:

Using Python grid, VBox, and HBox layouts, I gained an understanding of the fundamentals of GUI layout management. I gained experience arranging components and observing how they change when the window is resized by creating several GUI programs. Making a calculator for the extra task improved my comprehension of file operations and event handling.