

Laboratory Activity No. 7	
TUI to GUI in Pycharm	
Course Code: CPE009B	Program: Computer Engineering
Course Title: Object-Oriented Programming	Date Performed: 11/14/2024
Section: CPE21S1	Date Submitted: 11/14/2024
Name(s): Abarabar, John Nathan R. Aduana, Jake Norwin Bulambao, Adrian Justin M. Gaspar, Aaron Rowen Potestades, North Nygel G.	Instructor: Engr. Sayo, Maria Rizette

6. Output

1. Code:

```
#TUI Form
1 usage
def main():
    # Find the largest number among three numbers
    L = []
    num1 = eval(input("Enter the first number:"))
    L.append(num1)
    num2 = eval(input("Enter the second number:"))
    L.append(num2)
    num3 = eval(input("Enter the third number:"))
    L.append(num3)
    print("The largest number among the three is:",str(max(L)))
main()
```

Output:

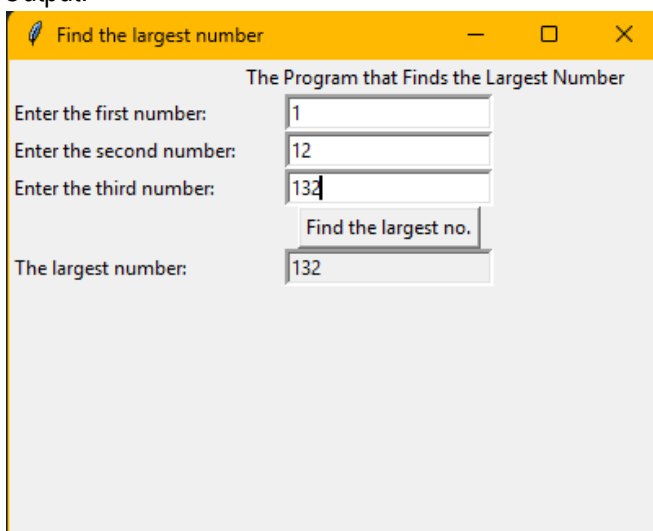
```
"C:\Users\TIPQC\Documents\Bulambao\lab 7\.venv\Scripts\python.exe"
Enter the first number:12
Enter the second number:32
Enter the third number:12
The largest number among the three is: 32

Process finished with exit code 0
|
```

2. Code:

```
TUIform.py* x GUIform.py* x
1 from tkinter import *
2 window = Tk()
3 window.title("Find the Largest number")
4 window.geometry("400x300+20+10")
5 def findLargest():
6     L = []
7     L.append(eval(conOfent2.get()))
8     L.append(eval(conOfent3.get()))
9     L.append(eval(conOfent4.get()))
10    conOfLargest.set(max(L))
11    lbl1 = Label(window, text = "The Program that Finds the Largest Number")
12    lbl1.grid(row=0, column=1, columnspan=2, sticky=EW)
13    lbl2 = Label(window, text = "Enter the first number:")
14    lbl2.grid(row=1, column = 0, sticky=W)
15    conOfent2 = StringVar()
16    ent2 = Entry(window, bd=3, textvariable=conOfent2)
17    ent2.grid(row=1, column = 1)
18    lbl3 = Label(window, text = "Enter the second number:")
19    lbl3.grid(row=2, column=0)
20    conOfent3=StringVar()
21    ent3 = Entry(window, bd=3, textvariable=conOfent3)
22    ent3.grid(row=2, column=1)
23    lbl4 = Label(window, text="Enter the third number:")
24    lbl4.grid(row=3, column =0, sticky=W)
25    conOfent4 = StringVar()
26    ent4 = Entry(window, bd=3, textvariable=conOfent4)
27    ent4.grid(row=3, column=1)
28    btn1 = Button(window, text = "Find the largest no.", command=findLargest)
29    btn1.grid(row=4, column = 1)
30    lbl5 = Label(window, text="The Largest number:")
31    lbl5.grid(row=5, column=0, sticky=W)
32    conOfLargest = StringVar()
33    ent5 = Entry(window, bd=3, state="readonly", textvariable=conOfLargest)
34    ent5.grid(row=5, column=1)
35    mainloop()
36
37
38
39
```

Output:

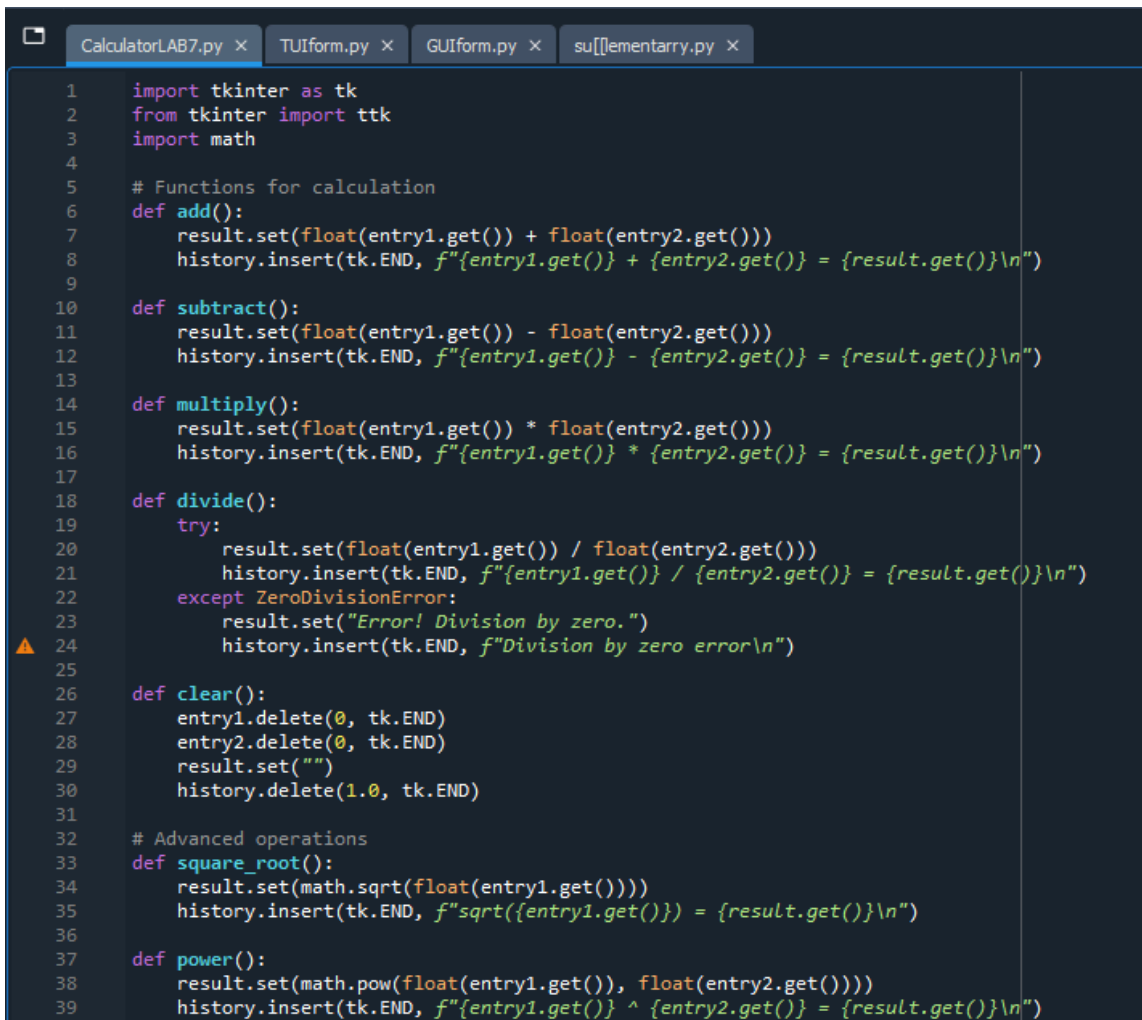


Questions:

1. What is TUI in Python?
-TUI or called as Text-User Interface is a type of interface that allows a user to interact with a program or application using text-based interfaces like text and keyboard inputs instead of graphical elements. It provides users with a more sophisticated text-based interface without the use of GUI and provides its users with a much more different experience.
2. How to make a TUI in Python
-To create a TUI the user can utilize something known as curses or unwid which are types of libraries to create terminal screen management, interactive widgets, user input and many more. Start it up by setting up the terminal screen, defining the layout for processing keyboard or mouse events and lastly use event loops to keep the interface responsive and update the display based on user interaction.
3. What is the difference between TUI and GUI?
-From the name itself there is a major difference as to how both interfaces handle processes and inputs. For GUI, it mostly relies on visual elements like windows, buttons, and icons to enable user interaction, typically requiring a graphical display and using a mouse. GUI is also much more interactive and user friendly due to being able to use graphical representations and gives them a broader basis. TUI is more or within the command-line environment, using text-based elements like menus, text fields, and arrows for navigation. TUI is faster and much more simple making it better to be used in a field where there are limited resources as they are faster and more efficient due to that.

7. Supplementary Activity

CODE



```
CalculatorLAB7.py x TUIform.py x GUIform.py x supplementary.py x
1 import tkinter as tk
2 from tkinter import ttk
3 import math
4
5 # Functions for calculation
6 def add():
7     result.set(float(entry1.get()) + float(entry2.get()))
8     history.insert(tk.END, f"{entry1.get()} + {entry2.get()} = {result.get()}\n")
9
10 def subtract():
11     result.set(float(entry1.get()) - float(entry2.get()))
12     history.insert(tk.END, f"{entry1.get()} - {entry2.get()} = {result.get()}\n")
13
14 def multiply():
15     result.set(float(entry1.get()) * float(entry2.get()))
16     history.insert(tk.END, f"{entry1.get()} * {entry2.get()} = {result.get()}\n")
17
18 def divide():
19     try:
20         result.set(float(entry1.get()) / float(entry2.get()))
21         history.insert(tk.END, f"{entry1.get()} / {entry2.get()} = {result.get()}\n")
22     except ZeroDivisionError:
23         result.set("Error! Division by zero.")
24         history.insert(tk.END, f"Division by zero error\n")
25
26 def clear():
27     entry1.delete(0, tk.END)
28     entry2.delete(0, tk.END)
29     result.set("")
30     history.delete(1.0, tk.END)
31
32 # Advanced operations
33 def square_root():
34     result.set(math.sqrt(float(entry1.get())))
35     history.insert(tk.END, f"sqrt({entry1.get()}) = {result.get()}\n")
36
37 def power():
38     result.set(math.pow(float(entry1.get()), float(entry2.get())))
39     history.insert(tk.END, f"{entry1.get()} ^ {entry2.get()} = {result.get()}\n")
40
```

CalculatorLAB7.py x

TUIform.py x

GUIform.py x

su[[ementarry.py x

```
40
41 # Create the main window
42 root = tk.Tk()
43 root.title("Enhanced Calculator")
44
45 # Create StringVar to hold the result
46 result = tk.StringVar()
47
48 # Style configuration
49 style = ttk.Style()
50 style.configure("TLabel", font=("Helvetica", 14), background="Gray")
51 style.configure("TButton", font=("Helvetica", 12), padding=10)
52 style.configure("TEntry", font=("Helvetica", 14), background="Gray")
53 style.configure("TFrame", background="Gray")
54 root.config(bg="#1c1c1c")
55
56 # Create the layout
57 mainframe = ttk.Frame(root, padding="10 10 10 10", style="TFrame")
58 mainframe.grid(column=0, row=0, sticky=(tk.W, tk.E, tk.N, tk.S))
59
60 ttk.Label(mainframe, text="Enter first number:", style="TLabel").grid(row=0, column=0, padx=5, pady=5)
61 entry1 = ttk.Entry(mainframe, style="TEntry")
62 entry1.grid(row=0, column=1, padx=5, pady=5)
63
64 ttk.Label(mainframe, text="Enter second number:", style="TLabel").grid(row=1, column=0, padx=5, pady=5)
65 entry2 = ttk.Entry(mainframe, style="TEntry")
66 entry2.grid(row=1, column=1, padx=5, pady=5)
67
68 # Buttons for operations
69 ttk.Button(mainframe, text="Add", command=add, style="TButton").grid(row=2, column=0, padx=5, pady=5)
70 ttk.Button(mainframe, text="Subtract", command=subtract, style="TButton").grid(row=2, column=1, padx=5, pady=5)
71 ttk.Button(mainframe, text="Multiply", command=multiply, style="TButton").grid(row=3, column=0, padx=5, pady=5)
72 ttk.Button(mainframe, text="Divide", command=divide, style="TButton").grid(row=3, column=1, padx=5, pady=5)
73 ttk.Button(mainframe, text="Square Root", command=square_root, style="TButton").grid(row=4, column=0, padx=5, pady=5)
74 ttk.Button(mainframe, text="Power", command=power, style="TButton").grid(row=4, column=1, padx=5, pady=5)
75
76 # Label to show result
77 ttk.Label(mainframe, text="Result:", style="TLabel").grid(row=5, column=0, padx=5, pady=5)
78 result_label = ttk.Label(mainframe, textvariable=result, style="TLabel")
79 result_label.grid(row=5, column=1, padx=5, pady=5)
80
81 # Clear button
82 ttk.Button(mainframe, text="Clear", command=clear, style="TButton").grid(row=6, column=0, padx=5, pady=5)
83
84 # History
85 ttk.Label(mainframe, text="History:", style="TLabel").grid(row=7, column=0, padx=5, pady=5)
86 history = tk.Text(mainframe, height=10, width=30, font=("Helvetica", 12), bg="#f0f0f0", wrap=tk.WORD)
87 history.grid(row=7, column=1, padx=5, pady=5)
88
89 # Input validation
90 def validate_entry(char):
91     return char.isdigit() or char == '.' or char == '-'
92
93 vcmd = (root.register(validate_entry), '%S')
94 entry1.config(validate='key', validatecommand=vcmd)
95 entry2.config(validate='key', validatecommand=vcmd)
96
97 # Start the main loop
98 root.mainloop()
```

OUTPUT:

Enhanced Calculator

Enter first number:

Enter second number:

Add

Subtract

Multiply

Divide

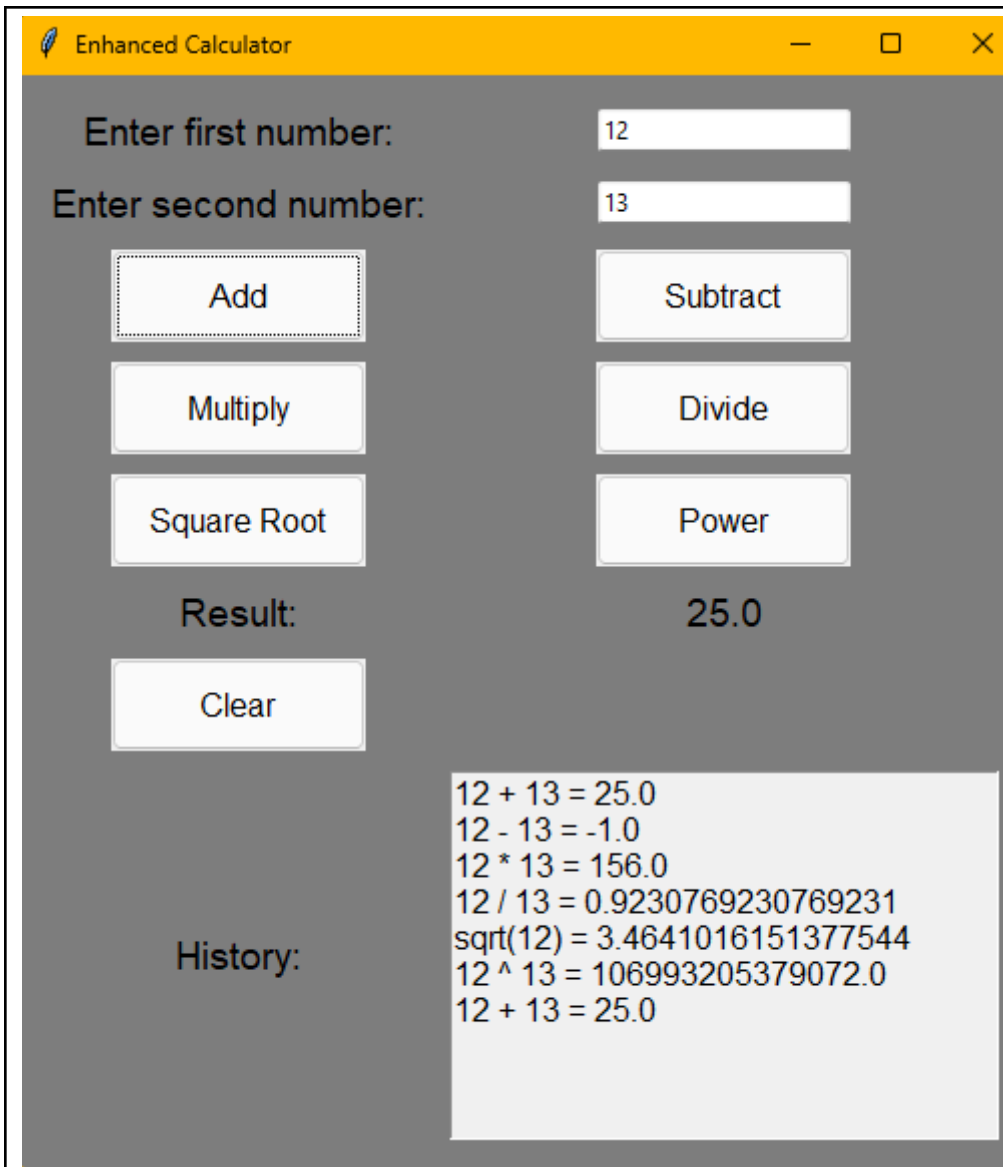
Square Root

Power

Result:

Clear

History:



8. Conclusion

From this activity, we were able to learn the process of converting a TUI program into a GUI program, where we replace the text interface of our previous programs with a graphical interface, to make ease-of-use better for the program. GUI can make a world of difference when compared to a TUI as the average user is able to understand visuals better than they can a text terminal, especially if that text terminal is cluttered. Experience in doing this conversion is important for our future, as we may have a role where we have to make graphics interfaces for programs in the future, and is overall a good thing to have in your skillset.

9. Assessment Rubric