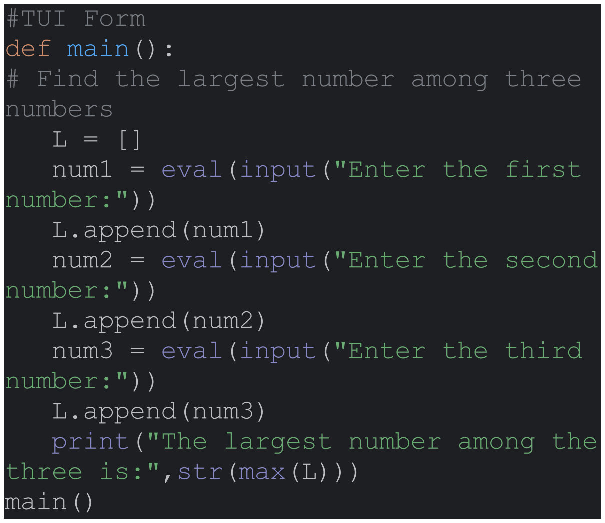
Activity Name #7 - TUI to GUI in Pycharm	
Fernandez, Don Eleazar T. Esteban, Prince Wally G. Sanchez, Christan R. Tandayu, Leoj Jeam B. Valleser, Kenn Jie L.	11/11/2024
CPE009B/CPE21S4	Engr. Ma. Rizette Sayo

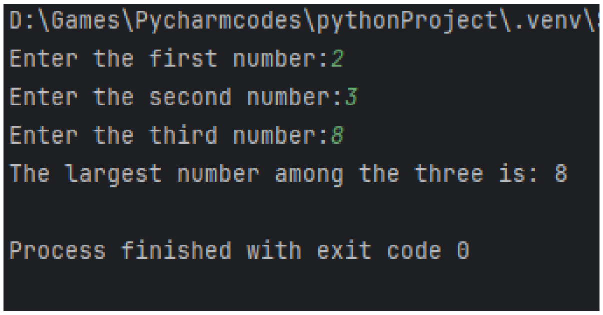
Procedure 1

Code



```
#TUI Form
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()
```

Console



```
D:\Games\Pycharmcodes\pythonProject\.venv\
Enter the first number:2
Enter the second number:3
Enter the third number:8
The largest number among the three is: 8

Process finished with exit code 0
```

Questions

1.
- A TUI (Text-based User Interface) in Python is a way to create programs that users interact with through text in the terminal. This type of interface doesn’t need graphics or windows, everything happens with text input and output, which makes it simpler and faster for quick tasks.
2.

- You can make a TUI in Python using libraries like curses, Rich, or prompt_toolkit. These tools let you add colors, tables, and prompts, so you can make text-based interactions more organized and user-friendly, all within the terminal. use another example

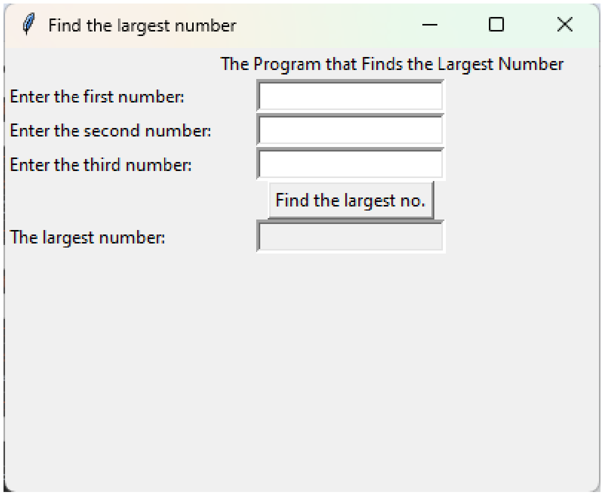
3.

- TUIs are text-only and run in the terminal, so they're lightweight and work well for simple or low-resource tasks. GUI have buttons, icons, and windows and need more system resources, but they're generally more visually intuitive for use

Procedure 2

```
from tkinter import *

window = Tk()
window.title("Find the largest number")
window.geometry("400x300+20+10")

conOfent2 = StringVar()
conOfent3 = StringVar()
conOfent4 = StringVar()
conOfLargest = StringVar()

def findLargest():
   L = []
   L.append(eval(conOfent2.get()))
   L.append(eval(conOfent3.get()))
   L.append(eval(conOfent4.get()))
   conOfLargest.set(max(L))

lbl1 = Label(window, text="The Program that Finds the Largest Number")
lbl1.grid(row=0, column=1, columnspan=2, sticky=EW)

lbl2 = Label(window, text="Enter the first number:")
lbl2.grid(row=1, column=0, sticky=W)
ent2 = Entry(window, bd=3, textvariable=conOfent2)
ent2.grid(row=1, column=1)

lbl3 = Label(window, text="Enter the second number:")
lbl3.grid(row=2, column=0)
ent3 = Entry(window, bd=3, textvariable=conOfent3)
ent3.grid(row=2, column=1)

lbl4 = Label(window, text="Enter the third number:")
lbl4.grid(row=3, column=0, sticky=W)
ent4 = Entry(window, bd=3, textvariable=conOfent4)
ent4.grid(row=3, column=1)

btn1 = Button(window, text="Find the largest no.", command=findLargest)
btn1.grid(row=4, column=1)

lbl5 = Label(window, text="The largest number:")
```

```
lbl5.grid(row=5, column=0, sticky=W)
ent5 = Entry(window, bd=3, state="readonly",
textvariable=conOfLargest)
ent5.grid(row=5, column=1)

mainloop()
```



Supplementary

Code

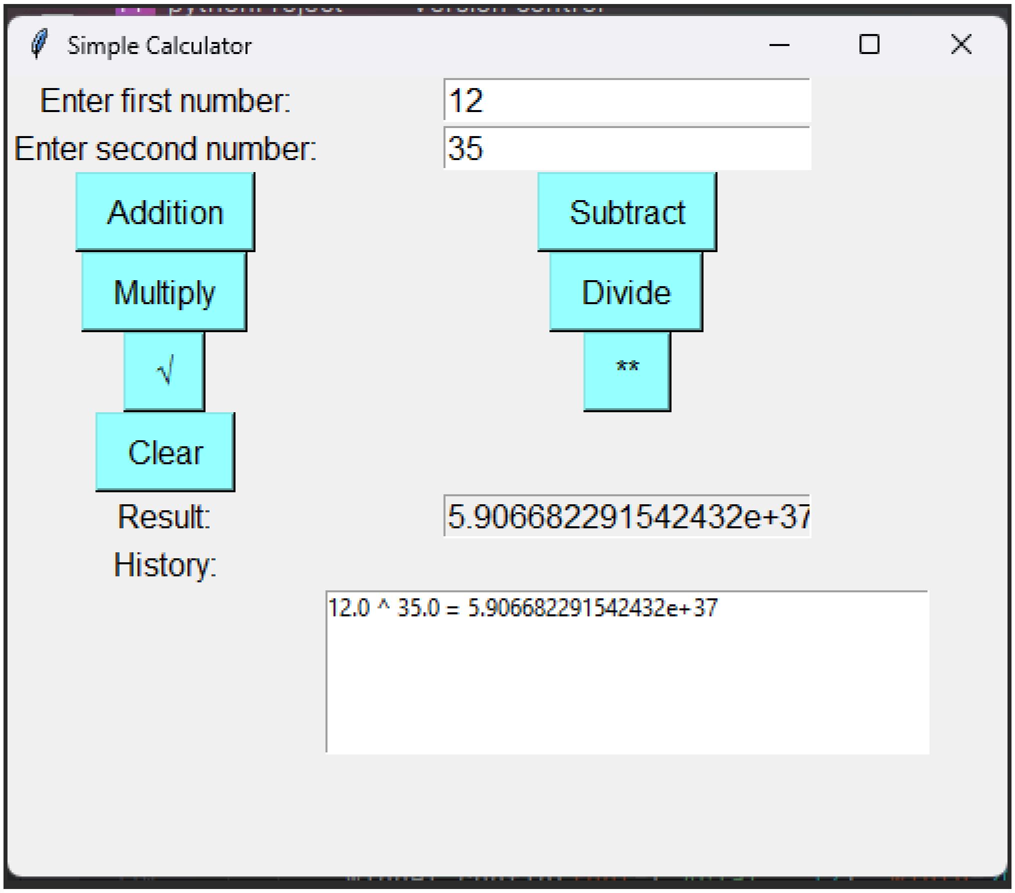
```
import tkinter as tk

# Functions for calculation
def addition():
    result.set(float(entry1.get()) + float(entry2.get()))

def subtract():
    result.set(float(entry1.get()) - float(entry2.get()))

def multiply():
    result.set(float(entry1.get()) * float(entry2.get()))

def divide():
    try:
        result.set(float(entry1.get()) / float(entry2.get()))
    except ZeroDivisionError:
        result.set("Error! Division by zero.")

# Create the main window
root = tk.Tk()
root.title("Simple Calculator")
root.geometry("400x300")  # Set a larger window size

# Create StringVar to hold the result
result = tk.StringVar()
```

```python
# Create the layout with padding
tk.Label(root, text="Enter first number:").grid(row=0, column=0, padx=10, pady=10)
entry1 = tk.Entry(root)
entry1.grid(row=0, column=1, padx=10, pady=10)

tk.Label(root, text="Enter second number:").grid(row=1, column=0, padx=10, pady=10)
entry2 = tk.Entry(root)
entry2.grid(row=1, column=1, padx=10, pady=10)

tk.Button(root, text="Add", command=addition).grid(row=2, column=0)
tk.Button(root, text="Subtract", command=subtract).grid(row=2, column=1)
tk.Button(root, text="Multiply", command=multiply).grid(row=3, column=0)
tk.Button(root, text="Divide", command=divide).grid(row=3, column=1)

tk.Label(root, text="Result:").grid(row=4, column=0)
tk.Entry(root, textvariable=result, state="readonly").grid(row=4, column=1)

root.mainloop()
```

Modified

```python
import tkinter as tk
from tkinter import messagebox
import math

# Functions for calculation
def addition():
    validate_input()
    result_value = float(entry1.get()) + float(entry2.get())
    result.set(result_value)
    history_list.insert(tk.END, f"{entry1.get()} + {entry2.get()} = {result_value}")

def subtract():
    validate_input()
    result_value = float(entry1.get()) - float(entry2.get())
    result.set(result_value)
    history_list.insert(tk.END, f"{entry1.get()} - {entry2.get()} = {result_value}")

def multiply():
    validate_input()
    result_value = float(entry1.get()) * float(entry2.get())
    result.set(result_value)
    history_list.insert(tk.END, f"{entry1.get()} * {entry2.get()} = {result_value}")

def divide():
    validate_input()
    try:
        result_value = float(entry1.get()) / float(entry2.get())
        result.set(result_value)
        history_list.insert(tk.END, f"{entry1.get()} / {entry2.get()} = {result_value}")
    except ZeroDivisionError:
        result.set("Error! Division by zero.")
        messagebox.showerror("Error", "Cannot divide by zero.")

def square_root():
    validate_input(entry1)
    value = float(entry1.get())
    result_value = math.sqrt(value)
    result.set(result_value)
    history_list.insert(tk.END, f"√{value} = {result_value}")

def power():
    validate_input()
    base = float(entry1.get())
    exponent = float(entry2.get())
    result_value = math.pow(base, exponent)
    result.set(result_value)
    history_list.insert(tk.END, f"{base} ^ {exponent} = {result_value}")

def clear():
    entry1.delete(0, tk.END)
    entry2.delete(0, tk.END)
    result.set("")
    history_list.delete(0, tk.END)

def validate_input(entry=None):
    if entry:
        try:
            float(entry.get())
        except ValueError:
            messagebox.showerror("Invalid input", "Please enter a valid number.")
            entry.delete(0, tk.END)

# Create the main window
root = tk.Tk()
root.title("Simple Calculator")
root.geometry("500x400")  # Set a larger window size

# Create StringVar to hold the result
result = tk.StringVar()

# Create the layout with padding
tk.Label(root, text="Enter first number:").grid(row=0, column=0)
entry1 = tk.Entry(root)
entry1.grid(row=0, column=1)

tk.Label(root, text="Enter second number:").grid(row=1, column=0)
entry2 = tk.Entry(root)
entry2.grid(row=1, column=1)

# Operation Buttons
```

```python
buttons = [
    ("Addition", addition),
    ("Subtract", subtract),
    ("Multiply", multiply),
    ("Divide", divide),
    ("√", square_root),
    ("**", power),
    ("Clear", clear)
]
for i, (text, command) in enumerate(buttons):
    tk.Button(root, text=text, command=command, bg="DarkSlateGray1",
font=("Purisa", 5)).grid(row=2 + i // 2, column=i % 2)


# Result Display
tk.Label(root, text="Result:").grid(row=6, column=0)
tk.Entry(root, textvariable=result, state="readonly").grid(row=6,
column=1)

# History Listbox
tk.Label(root, text="History:").grid(row=7, column=0)
history_list = tk.Listbox(root, width=50, height=5)
history_list.grid(row=10, column=1)

# Styling
for widget in root.winfo_children():
    if isinstance(widget, tk.Button):
        widget.config(font=("Arial", 12), padx=10, pady=5)
    elif isinstance(widget, tk.Label):
        widget.config(font=("Arial", 12))
    elif isinstance(widget, tk.Entry):
        widget.config(font=("Arial", 12), width=20)

root.mainloop()
```

Output	

Enter first number: 12
Enter second number: 35
Result: 5.906682291542432e+37
History:

```
12.0 ^ 35.0 = 5.906682291542432e+37
```

Conclusion
TUI are much easier to create and use as this only uses the terminal which doesn't need any graphical designs unlike GUI. It helps make it more visually appealing while functioning the same code like the TUI. For example in the activity, if we used TUI for the simple calculator we would need to type specific call out (1 : Add, 2 : Subtract , 3 : Multiply , 4: Divide). In GUI, the buttons are presented in the interface which makes it much easier for the user to use the application. Which makes TUI applications ideal for systems with limited resources or when a simple text based interface is needed. In Supplementary Activity, the given code has the basic operations for a calculator like addition, subtraction, multiplication, and division and displays the output in the text field. The Activity added new functions like square roots and exponentiation, along with a history feature that tracks and displays previous calculations.