

Laboratory Activity No. 8

Converting TUI to GUI Programs

Course Code: CPE103	Program: BSCPE
Course Title: Object-Oriented Programming	Date Performed: 3/15/2025
Section: BSCPE 1-A	Date Submitted: 3/15/2025
Name: PALMES, LEWIS CLARK L.	Instructor: ENGR. MARIA RIZETTE SAYO
1. Objective(s):	
This activity aims to convert a TUI program to GUI program with the Pycharm framework	
2. Intended Learning Outcomes (ILOs):	
The students should be able to: 2.1 Identify the main components in a GUI Application 2.2 Create a simple GUI Application that converts TUI program to GUI program	
3. Discussion:	
In general, programs consist of three components—input, processing, and output. In TUI programs, input is usually obtained from an input statement or by importing data from a file. Output is usually given by a print statement or stored in a file. When we convert a TUI program to a GUI program, we replace input and print statements with Label/Entry pairs. Processing data and inputting and outputting data to files works much the same in both types of programs. The primary difference is that the processing in GUI programs is usually triggered by an event	
4. Materials and Equipment:	
Desktop Computer with Anaconda Python or Pycharm Windows Operating System	
5. Procedure:	

1. Type these codes in Pycharm:

```
#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()
```

2. Run the program and observe the output.

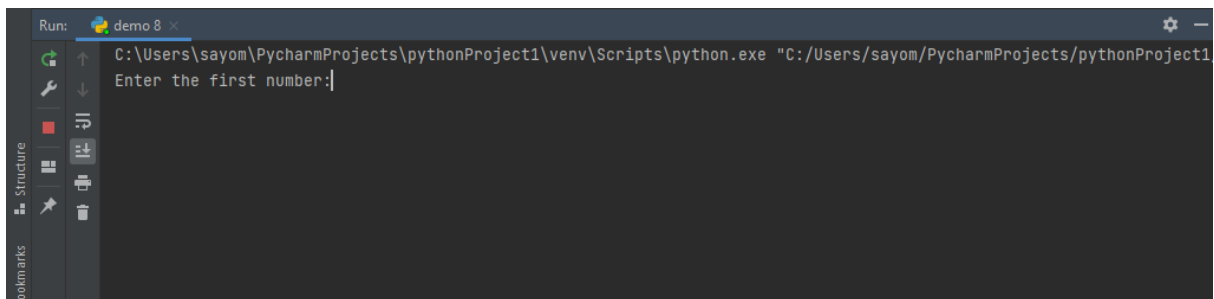


Figure 1. TUI form

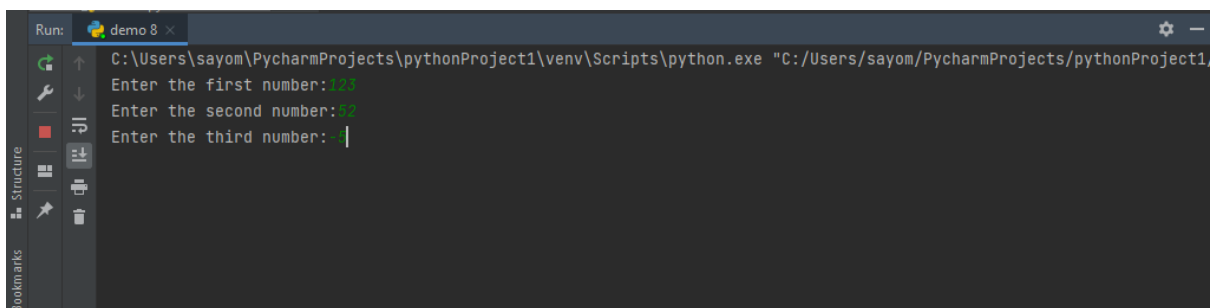


Figure 1(a) TUI form with three input numbers

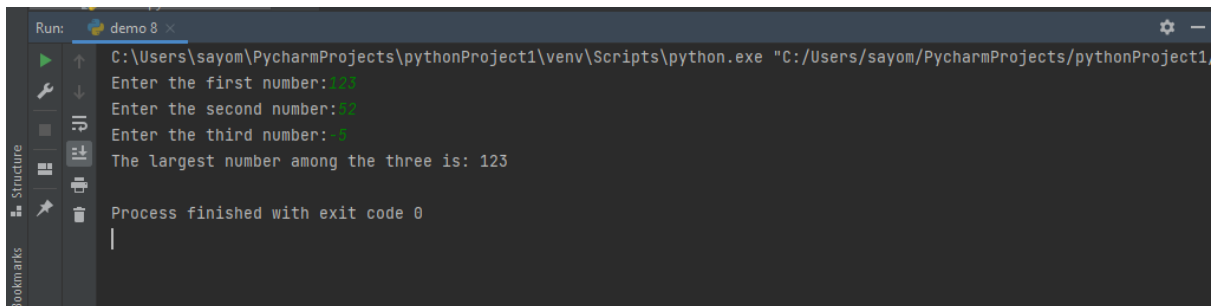


Figure 1(b) TUI form with output "The largest number among the three"

Please refer to this link:

[CPE-103-OOP-1-A/LAB 8.py at main · Lewis-Clark-Palmes/CPE-103-OOP-1-A](#)

Method 1 above shows a TUI program and a possible output in Figures 1(a) and (b) while Figure 2 shows the output of the GUI program in Method 2.

5. Procedure:

Method 2

```
from tkinter import *
```

```
window = Tk()
```

```
window.title("Find the largest number")
```

```
window.geometry("400x300+20+10")
```

```
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)
```

```
conOfent2 = StringVar()
```

```
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)
```

```
conOfent3=StringVar()
```

```
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)
```

```
conOfent4 = StringVar()
```

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

mainloop()
```

Results 2

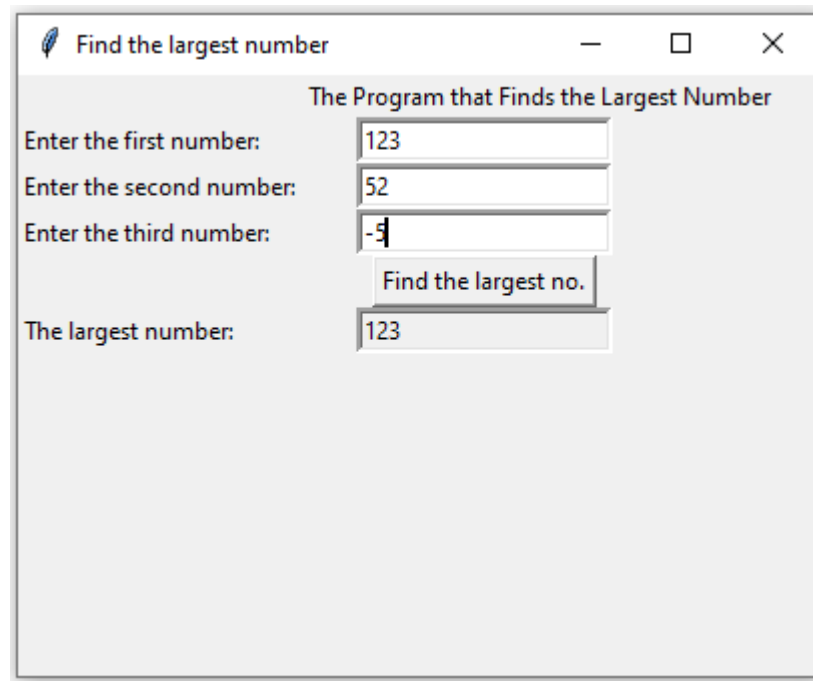


Figure 2. GUI program to find the largest number

Please refer to this link:

[CPE-103-OOP-1-A/Method 2 - Lab 8.py at main · Lewis-Clark-Palmes/CPE-103-OOP-1-A](#)

OBSERVATION: This contains converting a Text-based User Interface (TUI) program to a Graphical User Interface (GUI) program. TUI programs utilize command-line inputs and outputs, while GUI programs rely on visual elements like buttons and fields for a more interactive experience. The primary difference is that GUI processing is event-driven, triggered by user actions like button clicks.

Questions

1. What is TUI in Python?

TUI stands for Text-based User Interface, where interaction is performed through text inputs and outputs in the command line.

2. How to make a TUI in Python?

A TUI can be created using `input()` for data input and `print()` for displaying outputs, combined with logic for processing.

3. What is the difference between TUI and GUI?

TUI operates via text in a command-line interface, while GUI uses visual components like buttons and entry fields. GUI programs are event-driven, making them more user-friendly.

6. Supplementary Activity:

TUI Implementation

Please refer to this link:

[CPE-103-OOP-1-A/SUPPLEMENTARY TUI.py at main · Lewis-Clark-Palmes/CPE-103-OOP-1-A](https://github.com/Lewis-Clark-Palmes/CPE-103-OOP-1-A/blob/main/SUPPLEMENTARY%20TUI.py)

Simple TUI Calculator

```
def add(a, b):  
    return a + b
```

```
def subtract(a, b):  
    return a - b
```

```
def multiply(a, b):  
    return a * b
```

```
def divide(a, b):  
    if b != 0:  
        return a / b  
    else:  
        return "Error! Division by zero."
```

```
def main():  
    print("Simple Calculator")  
    print("Options:")  
    print("1. Add")  
    print("2. Subtract")  
    print("3. Multiply")  
    print("4. Divide")
```

```
    choice = input("Select operation (1/2/3/4): ")
```

```
    num1 = float(input("Enter first number: "))  
    num2 = float(input("Enter second number: "))
```

```
    if choice == '1':  
        print(f"{num1} + {num2} = {add(num1, num2)}")  
    elif choice == '2':  
        print(f"{num1} - {num2} = {subtract(num1, num2)}")  
    elif choice == '3':  
        print(f"{num1} * {num2} = {multiply(num1, num2)}")  
    elif choice == '4':
```

```
        print(f"{num1} / {num2} = {divide(num1, num2)}")
    else:
        print("Invalid input.")

if __name__ == "__main__":
    main()
```

GUI Conversion of the Calculator:

Please refer to this link:

[CPE-103-OOP-1-A/SUPPLEMENTARY GUI.py at main · Lewis-Clark-Palmes/CPE-103-OOP-1-A](https://github.com/Lewis-Clark-Palmes/CPE-103-OOP-1-A-SUPPLEMENTARY_GUI.py)

```
import tkinter as tk

# Functions for calculation
def add():
    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")

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

# Create the layout
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)

# Buttons for operations
tk.Button(root, text="Add", command=add).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)

# Label to show result
tk.Label(root, text="Result:").grid(row=4, column=0)
```

```
result_label = tk.Label(root, textvariable=result)
result_label.grid(row=4, column=1)
```

```
# Start the main loop
root.mainloop()
```


Once you've successfully created the GUI version of the calculator, try adding the following features to enhance the program:

1. **Clear Button:** Add a button to clear the input fields and reset the result.
2. **History Feature:** Add a list or label to show the history of operations performed.
3. **Advanced Operations:** Implement additional operations such as square roots, powers, or trigonometric functions.
4. **Input Validation:** Add validation to ensure that the user only enters numeric values in the input fields.
5. **Styling:** Experiment with different styles (font sizes, button colors) to improve the appearance of the GUI.

Please refer to this link:

[CPE-103-OOP-1-A/SUPPLEMENTARY ADVANCE CAL.py at main · Lewis-Clark-Palmes/CPE-103-OOP-1-A](https://github.com/Lewis-Clark-Palmes/CPE-103-OOP-1-A/blob/main/SUPPLEMENTARY%20ADVANCE%20CAL.py)

6. Conclusion

Converting TUI programs to GUI enhances user experience by providing a more interactive and visually appealing interface. Through frameworks like tkinter, students can modernize applications while maintaining core functionality, showcasing how GUI programs offer usability improvements over TUI.