**Asmaa Abdul-Amin**

**CRN#: 32016**

**Write-Up: Learning Experience & Enhancements**

**What Did I Learn?**

This project reinforced my understanding of **function design, user input handling, and error validation in Python**. The task required me to implement two main functions:

1. **A shape area calculator** that accepts user input for different shapes and calculates their areas.
2. **A function (foo) that takes an unlimited number of arguments** and returns the sum, average, maximum, and minimum values.

I also expanded the functionality by modifying the shape function to **include an additional shape (triangle)** and prompting the user to provide necessary arguments (e.g., base and height). This improved my ability to handle **dynamic user input and ensure proper data validation**.

**Challenges Encountered**

One challenge was ensuring that the program handled **invalid inputs** correctly. Users could enter text instead of numbers, negative values, or even zero, which would lead to incorrect calculations. I solved this by implementing **error handling using try-except blocks** and ensuring that only **valid, positive numeric values** were accepted.

Another issue was making the foo function **flexible enough to accept any number of arguments**. At first, handling edge cases like an empty input list was tricky, but I added validation to return a meaningful error message instead of crashing.

**Enhancements Made**

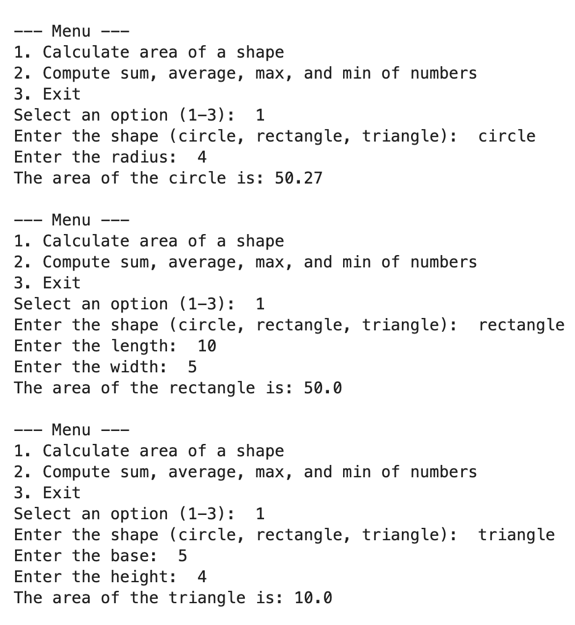
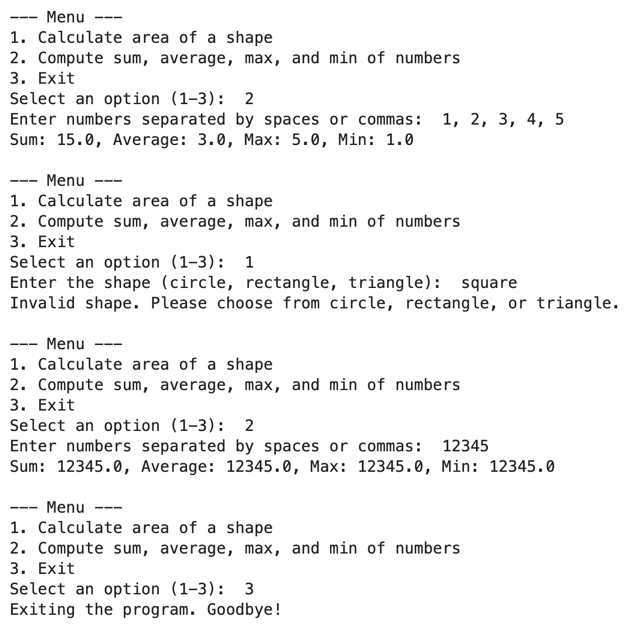
1. **Added an extra shape (Triangle) to the area calculator**
   * Users can now calculate the area of a **circle, rectangle, or triangle**.
   * The program **prompts the user** for required inputs (e.g., base and height for a triangle).
2. **Implemented foo() function for multiple values**
   * Accepts an **unlimited number of arguments**.
   * Returns **sum, average, maximum, and minimum** values.
   * Example:

foo(1, 3, 4) # Output: Sum: 8, Avg: 2.67, Max: 4, Min: 1

foo(5, 10, 15, 20, 25) # Output: Sum: 75, Avg: 15.0, Max: 25, Min: 5

1. **Created a version to allow user Input Handling**
   * Users can enter numbers **separated by spaces or commas** (e.g., "1 2 3" or "1,2,3" both work).
   * Invalid inputs (e.g., "abc" or negative values for shapes) trigger **clear error messages** instead of breaking the program.
2. **Interactive Menu for Multiple Calculations**
   * Users can perform **multiple calculations without restarting the program**.
   * A menu guides users through different options.

**Screenshots of Test Runs**

# Test Cases for Area Calculation

area\_test\_cases = [

{"shape": "circle", "radius": 5},

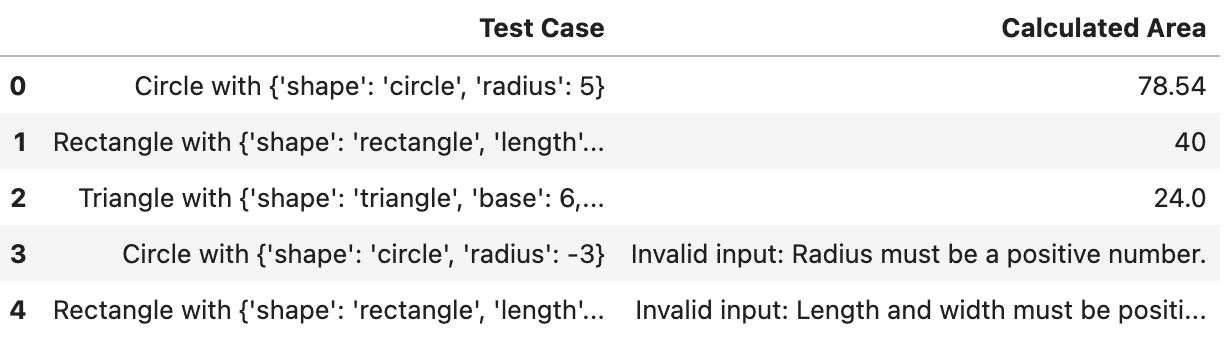
{"shape": "rectangle", "length": 10, "width": 4},

{"shape": "triangle", "base": 6, "height": 8},

{"shape": "circle", "radius": -3}, # Invalid case

{"shape": "rectangle", "length": 0, "width": 5}, # Invalid case

]



# Test Cases for Foo Function

foo\_test\_cases = [

(1, 3, 4),

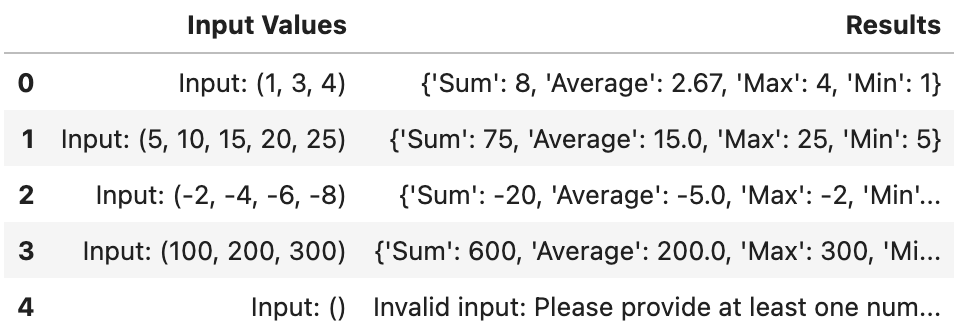
(5, 10, 15, 20, 25),

(-2, -4, -6, -8),

(100, 200, 300),

()

]



**Final Thoughts & Discussion Question**

This project helped me solidify my understanding of **function-based programming, input validation, and handling edge cases**. The interactive nature of the program made it more user-friendly, and **error handling improved its reliability**.

**Future Enhancements:**

* Extend the function to calculate **volumes of 3D shapes** (cylinder, sphere, cube, etc.).
* Implement a **GUI version** using **Tkinter** or **PyQt** for better user experience.