**Describe your experience with this assignment.**

When completing the assignment, there are some challenges involved in code logic, test case design, and tool familiarity. The requirements were clear but I should have required thorough consideration of edge cases. Another thing is learning to use testing tools , unittest.

Also, sufficient test cases should be considered, which depend on feature complexity, boundary conditions, and code coverage. When all features, scenarios, and edge cases were covered, and tests passed, I considered the testing process complete. However, some cases are listed here.

1.What challenges did you encounter with this assignment, if any?

I should Write the code correctly to classify triangles and detect right triangles, which is somewhat challenging, especially when considering different types of triangles and edge cases.

Secondly, comprehensive test cases should be covered various scenarios. Also, using testing tools (unittest) to organize and run tests may require some learning and familiarity.

2.What did you think about the requirements specification for this assignment?

According to the provided requirements specification, I wrote a function triangle\_clarification() to classify triangles based on the lengths of their three sides. The specification is relatively clear, but in actual code implementation and testing, more boundary cases and special scenarios should be considered to ensure program correctness.

3. What challenges did you encounter with the tools?

Import of unittest is the first thing, and the test cases should cover different scenarios. It is true that some time required to use them more practically.

4. Describe the criteria you used to determine that you had sufficient test cases, i.e. how did you know you were done?

When implementing triangle\_clarification(), I used the following criteria to determine that I had sufficient test cases and that the testing process was complete:

The first thing is coverage of Classification Scenarios: the test cases cover all possible triangle classification scenarios, including equilateral, isosceles, scalene, and non-triangle cases.

Handling of Edge Cases: such as triangles with minimum and maximum possible side lengths, as well as invalid inputs.

Right Triangle Detection: I included test cases to verify the correct detection of right triangles, covering both right and non-right triangle scenarios.

Branch and Code Coverage: I aimed to achieve high code coverage, making sure that all branches and logical paths in the triangle\_clarification() function were tested.

Error and Exception Handling is also be considered.

Performance Testing: In some cases, I also considered performance testing.

Boundary Conditions should also be considered, including very small, very large, and extreme values to assess how the function handled them.

Once I had covered these aspects comprehensively, and all tests passed successfully, I considered the testing process for triangle\_clarification() complete.