# HW 02a - Testing a legacy program and reporting on testing results

**Honor Pledge:**

I pledge my honor that I have abided by the Stevens Honor System.

**Author**: Sarthak Vaidya

**Assignment Description**:

These are the two files:  Triangle.py and TestTriangle.py

Triangle.py is a starter implementation of the triangle classification program.

TestTriangle.py contains a starter set of unittest test cases to test the classifyTriangle() function in the file Triangle.py file.

Update the test cases in in order to determine if the program is correctly implemented. You will need to update the test cases until you feel that your tests accurately test all of the conditions. You should run the complete set of tests against the original program to see how correct the triangle program is. Capture and then report those results in a formal report. For the first part do not make any changes to the triangle program. Changes should be made only to the test program.

Based on the results of your initial tests, you should make changes to the triangle program to fix all defects until all defects have been fixed. Run one final execution of the test program and capture and then report on those results in a formal test report described below.

Note that you should NOT simply replace the logic with your logic from Assignment 1.  Test teams typically don't have the luxury of rewriting code from scratch and instead must fix what's delivered to the test team.

**Summary**:

Initially I checked the code in Triangle.py. I recognised some of the errors while reviewing the code. I used these errors to devise test cases and also devised more test cases by predicting the errors that I might receive. I added these test cases to the incomplete TestTriangle.py file to check if the program in Triangle.py is error-free:

Test to check if the triangle is a right triangle
Test to check if the triangle is an equilateral triangle
Test to check the upper limit of the sides of the triangle
Test to check the lower limit of the sides of the triangle
Test to check if the given input does not form a triangle
Test to check if the triangle is isosceles
Test to check if the triangle is scalene
Test to check if the input values can be of string data type
Test to check if input values can be float data type

A total of 11 tests were run on the buggy implementation of the code. Out of these tests three tests passed and 8 failed. The tests that passed were the test to check the upper limit of the sides of the triangle, test to check the lower limit of the sides of the triangle and test to check if input values can be float data type. Although the test to check the lower limit of the sides of the triangle test passed, the code implementation to check this condition was wrong. Based on the results of this test run (Test Run 1) I formed my conclusions about the changes needed in the code in triangle.py.

The following conclusions were made on the basis of these tests and by code observation:
The logic for checking the lower limit of sides was wrong
The logic for checking if the given sides do not form a triangle was wrong
The logic for checking if the given triangle is an equilateral triangle was wrong
The logic for checking if the given triangle is a right triangle was wrong
The logic for checking if the given triangle is a scalene triangle was wrong.

Changes were made to the code for all the above logical errors and they were resolved. Then all the test cases were run once again and all the test cases passed. All the initial errors from test run 1 were resolved by making code changes and in test run 2 all the test cases passed.

In this assignment I learned how to write test cases after analysing the existing code. I was able to realise some of the test cases that were needed like to check the lower limit of the sides of the triangle, test to check if triangle is equilateral right by looking at the code. Some of the test cases like checking if string and float inputs are valid, I devised after I implemented other test cases. As mentioned in the code only values that are integers are valid, so I decided to test this condition by entering string and float values. The string input showed type error and the float input test passes as the program did not accept float values. I learned how to implement test runs to find the faults in any given code and to confirm the changes by using the same tests in a different test run. I got an understanding of implementing test cases to find bugs and to correct these bugs to make the program efficient.

Following is a table that denotes the test cases that were run on the buggy code along with their expected and actual results:

**Deliverable 1**:

| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
|---|---|---|---|---|
| testRightTriangleA | 3, 4, 5 | Right | InvalidInput | Fail |
| testRightTriangleB | 5, 3, 4 | Right | InvalidInput | Fail |
| testEquilateralTriangles | 1, 1, 1 | Equilateral | InvalidInput | Fail |
| testUpperLimit | 308, 407, 534 | InvalidInput | InvalidInput | Pass |
| testLowerLimit | -7, -15, -22 | InvalidInput | InvalidInput | Pass- As the results match we consider it pass. But the logic used here is wrong and faulty. |
| testIfTriangle | 1, 4, 5 | NotATriangle | InvalidInput | Fail |
| testIsoscelesTriangleA | 6, 6, 5 | Isoceles | InvalidInput | Fail |
| testIsoscelesTriangleB | 5, 7, 7 | Isosceles | InvalidInput | Fail |
| testScaleneTriangle | 7, 9, 11 | Scalene | InvalidInput | Fail |
| testStringTriangle | a, b, c | Error | Error | Type Error |
| testFloatTriangle | 5.2, 3.5, 8.9 | InvalidInput | InvalidInput | Pass |

Following is a table that denotes the test cases that were run on the improved code along with their expected and actual results:

**Deliverable 5**:

| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
|---|---|---|---|---|
| testRightTriangleA | 3, 4, 5 | Right | Right | Pass |
| testRightTriangleB | 5, 3, 4 | Right | Right | Pass |
| testEquilateralTriangles | 1, 1, 1 | Equilateral | Equilateral | Pass |
| testUpperLimit | 308, 407, 534 | InvalidInput | InvalidInput | Pass |
| testLowerLimit | -7, -15, -22 | InvalidInput | InvalidInput | Pass- Correct logic implemented |
| testIfTriangle | 1, 4, 5 | NotATriangle | NotATriangle | Pass |
| testIsoscelesTriangleA | 6, 6, 5 | Isoceles | Isoceles | Pass |
| testIsoscelesTriangleB | 5, 7, 7 | Isosceles | Isoceles | Pass |
| testScaleneTriangle | 7, 9, 11 | Scalene | Scalene | Pass |
| testStringTriangle | a, b, c | Error | Error | Type Error |
| testFloatTriangle | 5.2, 3.5, 8.9 | InvalidInput | InvalidInput | Pass |

As the table denotes that all the test cases pass after changes have been made to the code in Triangle.py.

Following is the summary of all the test cases that were run along with their respective results:

**Deliverable 6**:

| | Test Run 1 | Test Run 2 |
|---|---|---|
| Tests Planned | testRightTriangleA, testRightTriangleB, testEquilateralTriangles, testUpperLimit, testLowerLimit, testIfTriangle, testIsoscelesTriangleA, testIsoscelesTriangleB, testScaleneTriangle, testStringTriangle, testFloatTriangle | testRightTriangleA, testRightTriangleB, testEquilateralTriangles, testUpperLimit, testLowerLimit, testIfTriangle, testIsoscelesTriangleA, testIsoscelesTriangleB, testScaleneTriangle, testStringTriangle, testFloatTriangle |
| Tests Executed | testRightTriangleA, testRightTriangleB, testEquilateralTriangles, testUpperLimit, testIfTriangle, testIsoscelesTriangleA, testIsoscelesTriangleB, testScaleneTriangle, testStringTriangle, testFloatTriangle | testRightTriangleA, testRightTriangleB, testEquilateralTriangles, testUpperLimit, testLowerLimit, testIfTriangle, testIsoscelesTriangleA, testIsoscelesTriangleB, testScaleneTriangle, testStringTriangle, testFloatTriangle |
| Tests Passed | testFloatTriangle, testUpperLimit, testLowerLimit | testRightTriangleA, testRightTriangleB, testEquilateralTriangles, testUpperLimit, testLowerLimit, testIfTriangle, testIsoscelesTriangleA, testIsoscelesTriangleB, testScaleneTriangle, testStringTriangle, testFloatTriangle |
| Defects Found | The logic for checking if the lower limit of sides is wrong.<br>The logic for checking if the given sides do not form a triangle is wrong.<br>The logic for checking if the given triangle is an equilateral triangle is wrong.<br>The logic for checking if the given triangle is a right triangle is wrong.<br>The logic for checking if the given triangle is a scalene triangle is wrong. | None |
| Defects Fixed | Rewrote the logic for all the logical errors mentioned above. | None |

**Deliverable 7**:

Link to Git Repo: https://github.com/Sarthak15997/SSW567A_HW02a

Please check Triangle.py and TestTriangle.py files for updated code.