**SSW-567 HW02(a)**

1. Assignment Description:

Sometimes you will be given a program that someone else has written, and you will be asked to fix, update and enhance that program. In this assignment you will start with an existing implementation of the classify triangle program that will be given to you. You will also be given a starter test program that tests the classify triangle program, but those tests are not complete.

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.

In order to determine if the program is correctly implemented, you will need to update the set of test cases in the test program. You will need to update the test program until you feel that your tests adequately test all of the conditions. Then you should run the complete set of tests against the original triangle program to see how correct the triangle program is. Capture and then report on those results in a formal test report described below. For this first part you should not make any changes to the classify triangle program. You should only change the test program.

Based on the results of your initial tests, you will then update the classify triangle program to fix all defects. Continue to run the test cases as you fix defects until all of the 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.

Triangle.py contains an implementation of the classifyTriangle() function with a few bugs.

TestTriangle.py contains the initial set of test cases

1. Author: Siddharth Kaza
2. Summary:
3. A screenshot of a computer program

   Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Test Report 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Input** | **Expected Results** | **Actual Result** | **Pass or Fail** |
| test\_equilaterial\_triangle | (3,3,3) | Equilateral | InvalidInput | FAILED |
| test\_equilaterial\_triangle | (10,10,10) | Equilateral | InvalidInput | FAILED |
| test\_equilaterial\_triangle | (0.1,0.1,0.1) | Equilateral | InvalidInput | FAILED |
| test\_right\_triangle | (5,12,13) | Right | InvalidInput | FAILED |
| test\_right\_triangle | (3,4,5) | Right | InvalidInput | FAILED |
| test\_right\_triangle | (5,3,4) | Right | InvalidInput | FAILED |
| test\_scalene\_triangle | (10,11,12) | Scalene | InvalidInput | FAILED |
| test\_isoceles\_triangle | (5,5,8) | Isoceles | InvalidInput | FAILED |
| test\_invalid\_triangle | (201,201,201) | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (0,0,0) | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (0,0,100) | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (0,0.5,10.29 | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (2,20,33) | NotATriangle | InvalidInput | FAILED |

Test Report 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Input** | **Expected Results** | **Actual Result** | **Pass or Fail** |
| test\_equilaterial\_triangle | (3,3,3) | Equilateral | Equilateral | PASSED |
| test\_equilaterial\_triangle | (10,10,10) | Equilateral | Equilateral | PASSED |
| test\_equilaterial\_triangle | (0.1,0.1,0.1) | Equilateral | Equilateral | PASSED |
| test\_right\_triangle | (5,12,13) | Right | Right | PASSED |
| test\_right\_triangle | (3,4,5) | Right | Right | PASSED |
| test\_right\_triangle | (5,3,4) | Right | Right | PASSED |
| test\_scalene\_triangle | (10,11,12) | Scalene | Scalene | PASSED |
| test\_isoceles\_triangle | (5,5,8) | Isoceles | Isoceles | PASSED |
| test\_invalid\_triangle | (201,201,201) | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (0,0,0) | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (0,0,100) | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (0,0.5,10.29 | InvalidInput | InvalidInput | PASSED |
| test\_invalid\_triangle | (2,20,33) | NotATriangle | NotATriangle | PASSED |

A screenshot of a computer program

Description automatically generated

|  |  |  |
| --- | --- | --- |
|  | Test Case 1 | Test Case 2 |
| Tests Planned | 13 tests planned | 13 tests planned |
| Tests Executed | All tests executed | All tests Executed |
| Tests Passed | 4 tests passed out of 13. Rest Failed | All 13 tests PASSED |
| Defects Found | 1. if a <= 0 or b <= b or c <= 0: (b should be less than or equal to 0 and not less than or equal to b 2. if not(isinstance(a,int) and isinstance(b,int) and isinstance(c,int)): (Does not account for floating point numbers) 3. if (a >= (b - c)) or (b >= (a - c)) or (c >= (a + b)): (Any side should be greater than the sum of its other two sides) 4. a == b and b == a: (all three sides should be equal and not any two given sides) 5. (a \* 2) + (b \* 2)) == (c \* 2): (Pythagoras theorem must be applied for right-angled triangle where the square of the hypotenuse is equal to the sum of squares of its other two sides. 6. elif (side\_a != side\_b) and (side\_b != side\_c) and (side\_a != side\_b): (Incorrect logic as in a scalene triangle all three sides should be different. | None |
| Defects Fixed | All defects found and fixed | 1. if a <= 0 or b <= 0 or c <= 0: 2. if not(isinstance(a,numbers.Number) and isinstance(b,numbers.Number) and isinstance(c,numbers.Number)): 3. if (a >= (b + c)) or (b >= (a + c)) or (c >= (a + b)): 4. if a == b == c: 5. elif ((a \*\* 2) + (b \*\* 2) == (c \*\* 2)) or ((a\*\*2)+(c\*\*2)==(b\*\*2)) or ((b\*\*2)+(c\*\*2)==(a\*\*2)): 6. elif (a != b) and (b != c) and (a != c): |

Reflection: This assignment was certainly tricky. Mainly to build out suitable test cases and to fix the logic in the code. I was having a lot of trouble getting pytest to work which also increased the time given to this assignment by a significant amount. I learnt about crafting suitable test cases and trying to fix the logic of a code without rewriting it completely which is what I did initially with the isosceles and scalene triangle code but then I found another way of fixing the logic without significantly overhauling the code itself.

1. Stevens Honor Pledge: I PLEDGE MY HONOR THAT I HAVE ABIDED BY THE STEVENS HONOR SYSTEM
2. Detailed results: I mainly used pytest as I found it to be more easier and intuitive to use as compared to unittest. One of the most important defects to be fixed was in the i) if a <= 0 or b <= b or c <= 0: Which should be

if a <= 0 or b <= 0 or c <= 0: This defect was important to fix as no matter how many changes you make to the logic of the code, the test case will always return an InvalidTriangle. All of the report summaries as well as screenshots are attached above. I am attaching the GitHub repository link along with this report too.

GitHub Repo Link: <https://github.com/sidhdhuk09/SSW-567-HW02a>