Author: Arjun Khatri

Summary: First thing we had to do was add extensive test cases so all cases were checked. When we first tested the original file only 2 test cases passed out of 10. After we refactored our code we were able to get all 10 test cases to pass! When we had to go back in and change the code I enjoyed that we were not allowed to change the overall structure. This felt like a situation that would occur in the real world and felt like a great simulation of that. I learned how to right extensive test cases that check for all cases and how to refactor code. At first I had to configure a conda environment this didn’t work at first but I eventually figured it out.

**Description of assignment:**

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***](https://sit.instructure.com/courses/71371/files/12972054/download?wrap=1)is a starter implementation of the triangle classification program.
  + [***TestTriangle.py***](https://sit.instructure.com/courses/71371/files/12972044/download?wrap=1)**c**ontains 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*](https://sit.instructure.com/courses/71371/files/12972054/download?wrap=1)*contains an implementation of the classifyTriangle() function with a few bugs.*  
  
[*TestTriangle.py*](https://sit.instructure.com/courses/71371/files/12972044/download?wrap=1)*contains the initial set of test cases*

| Test ID | Input | Expected Result | 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 |
| testIsoscelesTriangleA | (5, 5, 3) | Isosceles | InvalidInput | Fail |
| testIsoscelesTriangleB | (4, 6, 6) | Isosceles | InvalidInput | Fail |
| testScaleneTriangle | (4, 2, 3) | Scalene | InvalidInput | Fail |
| testInvalidTriangleA | (1, 10, 12) | NotATriangle | InvalidInput | Fail |
| testInvalidTriangleB | (10, 1, 1) | NotATriangle | InvalidInput | Fail |

| Test ID | Input | Expected Result | 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 |
| testIsoscelesTriangleA | (5, 5, 3) | Isosceles | Isosceles | Pass |
| testIsoscelesTriangleB | (4, 6, 6) | Isosceles | Isosceles | Pass |
| testScaleneTriangle | (4, 2, 3) | Scalene | Scalene | Pass |
| testInvalidTriangleA | (1, 10, 12) | NotATriangle | NotATriangle | Pass |
| testInvalidTriangleB | (10, 1, 1) | NotATriangle | NotATriangle | Pass |

| **Test Run 1** | **Test Run 2** |
| --- | --- |
| Tests Planned | 10 | 10 |
| Tests Executed | 10 | 10 |
| Tests Passed | 2 | 10 |
| Defects Found | 8 | 0 |
| Defects Fixed | 8 | 0 |

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