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.

Author:
Jack Nonnenmacher

Summary:
In this project we were given an incomplete test case for a buggy program. We were supposed to first create a complete test case and then debug the program. The results of each test case on the initial buggy program, the results of each test run and the results of each test case on the debugged program are shown below on each respective chart.

TestTriangle.py on initial buggy classifyTriangle()

Test Id	Input	Expected Result	Actual Result	Pass/Fail
testRightTriangleA	3,4,5	Right	InvalidInput	Fail
testRightTriangleB	5,3,4	Right	InvalidInput	Fail
testEquilateralTriangles	1,1,1	Equilateral	InvalidInput	Fail
testNegativeA	-4,3,2	InvalidInput	InvalidInput	Pass
testNegativeB	4,-3,2	InvalidInput	InvalidInput	Pass
testNegativeC	4,3,-2	InvalidInput	InvalidInput	Pass
testTooLargeA	130,130, 230	InvalidInput	InvalidInput	Pass
testTooLargeB	130,630, 30	InvalidInput	InvalidInput	Pass
testTooLargeC	301,130, 30	InvalidInput	InvalidInput	Pass
testNotATriangleA	2,5,8	NotATriangle	InvalidInput	Fail
testNotATriangleB	4,10,6	NotATriangle	InvalidInput	Fail
testRightTriangleC	3,5,4	Right	InvalidInput	Fail
testScaleneA	2,4,5	Scalene	InvalidInput	Fail
testScaleneB	6,9,5	Scalene	InvalidInput	Fail
testIsoA	4,4,6	Isosceles	InvalidInput	Fail
testIsoB	8,4,8	Isosceles	InvalidInput	Fail
testIsoC	4,3,3	Isosceles	InvalidInput	Fail

Test Run Matrix

	Test Run 1	Test Run 2	Test Run 3	Test Run 4
Tests Planned	17	17	17	17
Tests Executed	17	17	17	17
Tests Passed	6	15	15	17
Defects Found	7	1	1	0
Defects Fixed	6	0	1	0

TestTriangle.py on debugged classifyTriangle()

Test Id	Input	Expected Result	Actual Result	Pass/Fail
testRightTriangleA	3,4,5	Right	Right	Pass
testRightTriangleB	5,3,4	Right	Right	Pass
testEquilateralTriangles	1,1,1	Equilateral	Equilateral	Pass
testNegativeA	-4,3,2	InvalidInput	InvalidInput	Pass
testNegativeB	4,-3,2	InvalidInput	InvalidInput	Pass
testNegativeC	4,3,-2	InvalidInput	InvalidInput	Pass
testTooLargeA	130,130, 230	InvalidInput	InvalidInput	Pass
testTooLargeB	130,630, 30	InvalidInput	InvalidInput	Pass
testTooLargeC	301,130, 30	InvalidInput	InvalidInput	Pass
testNotATriangleA	2,5,8	NotATriangle	NotATriangle	Pass
testNotATriangleB	4,10,6	NotATriangle	NotATriangle	Pass
testRightTriangleC	3,5,4	Right	Right	Pass
testScaleneA	2,4,5	Scalene	Scalene	Pass

testScaleneB	6,9,5	Scalene	Scalene	Pass
testIsoA	4,4,6	Isosceles	Isosceles	Pass
testIsoB	8,4,8	Isosceles	Isosceles	Pass
testIsoC	4,3,3	Isosceles	Isosceles	Pass

Reflection:
The reason I chose the test cases that I did was because I wanted to test each type of triangle and make sure that each type of triangle worked for all the different types of inputs. For example in the case of a Isosceles triangle I wanted to make sure that the program detected it as Isosceles when the first two inputs were equal, when the last two were equal, when the first and the last inputs were equal, when the two inputs that are equal are larger than the third input and when the two inputs that are equal is smaller than the third input. Test run 1 was against the initial buggy program. After analyzing the buggy program I was confident in my ability to debug it in one go which made me surprised when two of the tests failed in test run 2. The tests that failed were testRightTriangleB and testRightTriangleC. I thought that these two had failed because a**2 in python did not make it a^2 so I changed this to a*a. It turns out this was not the problem so the same two tests failed in test run 3. After that test run I realized it was because it was a problem with detecting it as a right angle when c was not the largest side of the right triangle. After this realization I was able to debug this last bug leading to a successful test run 4. In this project I learned how to utilize git with unit tests. I also gathered more experience working with unit tests.

Honor Code:
I pledge on my honor that I have abided by Steven's honor code.
Jack Nonnenmacher

Detailed Results:
Assumptions made in this project were that an input cannot be larger than 200 and that the side length had to be an integer. I made these assumptions because they were assumptions made in the code that was given to me to debug. Constraints that the project had were that the largest side of a triangle cannot be greater than or equal to the sum of the other two sides and that a side length cannot be less than or equal to 0. The inputs for the classifyTriangles() function were the three side lengths of a triangle. The testing made sense and I am very confident that the classifyTriangles() function is functioning properly. The tools that I utilized on the project were a mac, github, and pycharm.