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*** **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.

Author:

Julio Lora

Summary:

As a result of using test cases to locate bugs and possible fixes, I was able to efficiently change the code to ensure it passed all necessary test cases. I found there was some logic that continuously led to an output of “InvalidInput” that was easy to fix. This assignment taught me how useful it is to test code, especially if it is not originally your work. What worked best for me was to test all possible inputs and cases to make sure they returned the correct outputs.

Honor Pledge:

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

Detailed Results:

Below are a couple of tables displaying all of the tests I ran to find the bugs, along with their expected and actual results. The first table (Table 1) table was used to find what was failing which helped me find where the errors were. These tests were generated based on all of the possible expected outputs. For example, I made sure to include tests for each type of triangle along with tests for each invalid input. As long as these passed I would know the program is working as expected.

Table 2 shows the results after I finished fixing all of the bugs. As you can see, everything is passing now. One change I made was adding “a == c” to the if statement used to

identify an equilateral triangle. Previously this was not included and to be equilateral, every side must be the same.

Before Fixing Bugs (Table 1):

Test ID	Input	Expected Results	Actual Results	Pass/Fail
testRightTriangleA	(3,4,5)	“Right”	“InvalidInput”	Fail
testRightTriangleB	(5,3,4)	“Right”	“InvalidInput”	Fail
testEquilateralTriangles	(1,1,1)	“Equilateral”	“InvalidInput”	Fail
testScaleneTriangles	(10,15,23)	“Scalene”	“InvalidInput”	Fail
testIsocelesTriangles	(6,6,8)	“Isoceles”	“InvalidInput”	Fail
testNotATriangle	(1,2,3)	“NotATriangle”	“InvalidInput”	Fail
testInput1	('Hi',6,8)	“InvalidInput”	TypeError: '>' not supported between instances of 'str' and 'int'	Fail
testInput2	(6,'Hi',8)	“InvalidInput”	TypeError: '>' not supported between instances of 'str' and 'int'	Fail
testInput3	(6,6,'Hi')	“InvalidInput”	TypeError: '>' not supported between instances of 'str' and 'int'	Fail
testInput4	(-4,3,8)	“InvalidInput”	“InvalidInput”	Pass
testInput5	(4,-3,8)	“InvalidInput”	“InvalidInput”	Pass
testInput6	(4,3,-8)	“InvalidInput”	“InvalidInput”	Pass
testInput7	(400,3,8)	“InvalidInput”	“InvalidInput”	Pass

After Fixing Bugs (Table 2):

Test ID	Input	Expected Results	Actual Results	Pass/Fail

testRightTriangleA	(3,4,5)	“Right”	“InvalidInput”	Fail
testRightTriangleB	(5,3,4)	“Right”	“InvalidInput”	Fail
testEquilateralTriangles	(1,1,1)	“Equilateral”	“InvalidInput”	Fail
testScaleneTriangles	(10,15,23)	“Scalene”	“InvalidInput”	Fail
testIsocelesTriangles	(6,6,8)	“Isoceles”	“InvalidInput”	Fail
testNotATriangle	(1,2,3)	“NotATriangle”	“InvalidInput”	Fail
testInput1	('Hi',6,8)	“InvalidInput”	TypeError: '>' not supported between instances of 'str' and 'int'	Fail
testInput2	(6,'Hi',8)	“InvalidInput”	TypeError: '>' not supported between instances of 'str' and 'int'	Fail
testInput3	(6,6,'Hi')	“InvalidInput”	TypeError: '>' not supported between instances of 'str' and 'int'	Fail
testInput4	(-4,3,8)	“InvalidInput”	“InvalidInput”	Pass
testInput5	(4,-3,8)	“InvalidInput”	“InvalidInput”	Pass
testInput6	(4,3,-8)	“InvalidInput”	“InvalidInput”	Pass
testInput7	(400,3,8)	“InvalidInput”	“InvalidInput”	Pass

Matrix:

	Test Run 1	**Test Run 2**
Tests Planned	13	13
Tests Executed	13	13
Tests Passed	4	13
Defects Found	9	0
Defects Fixed	0	9