Tarik Kdiry
HW02a - Testing a legacy program and reporting on testing results
9 September 2018

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.

1)

TestID	Input	Expected Results	Actual Result	Pass or Fail
`testRightTriangleA`	(3,4,5)	`3,4,5 is a Right triangle'`	AssertionError: 3,4,5 is a Right triangle	F

testRightTriangleB	(5,3,4)	5,3,4 is a Right triangle'	AssertionError: 5,3,4 is a Right triangle	F
testRightTriangleC	(9,12,15)	9,12,15 is a Right triangle'	AssertionError: 9,12,15 is a Right triangle	F
testEquilateralTriangles	(1,1,1)	1,1,1 should be equilateral'	AssertionError: 1,1,1 should be equilateral	F
testScaleneTriangleA	(1,2,3)	1,2,3 is a Scalene triangle'	AssertionError: 1,2,3 is a Scalene triangle	F
testScaleneTriangleB	(100, 200, 300)	InvalidInput'	InvalidInput'	P
testIsoscelesTriangleA	(9,15,15)	9,15,15 is an Isosceles triangle'	AssertionError: 9,15,15 is an Isosceles triangle	F
testIsoscelesTriangleB	(100, 300, 300)	InvalidInput'	InvalidInput'	P

2) Included

3) Included

4)

```
Frodo:Triangle567 tarikkdiry$ python -m unittest TestTriangle
........
----------------------------------------------------------------------
Ran 8 tests in 0.001s

OK
Frodo:Triangle567 tarikkdiry$
```

5)

TestID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	(3,4,5)	3,4,5 is a Right triangle'	AssertionError: 3,4,5 is a Right triangle	F
testRightTriangleB	(5,3,4)	5,3,4 is a Right triangle'	AssertionError: 5,3,4 is a Right triangle	F

TestID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	(3,4,5)	3,4,5 is a Right triangle'	AssertionError: 3,4,5 is a Right triangle	F
testRightTriangleB	(5,3,4)	5,3,4 is a Right triangle'	AssertionError: 5,3,4 is a Right triangle	F
testRightTriangleC	(9,12,15)	9,12,15 is a Right triangle'	AssertionError: 9,12,15 is a Right triangle	F
testEquilateralTriangles	(1,1,1)	1,1,1 should be equilateral'	AssertionError: 1,1,1 should be equilateral	F
testScaleneTriangleA	(1,2,3)	1,2,3 is a Scalene triangle'	AssertionError: 1,2,3 is a Scalene triangle	F
testScaleneTriangleB	(100, 200, 300)	InvalidInput'	InvalidInput'	P
testIsoscelesTriangleA	(9,15,15)	9,15,15 is an Isosceles triangle'	AssertionError: 9,15,15 is an Isosceles triangle	F
testIsoscelesTriangleB	(100, 300, 300)	InvalidInput'	InvalidInput'	P

6)

	Test Run 1	Test Run 2
Tests Planned	9	9
Tests Executed	9	9
Test Passed	2	9
Defects Found	6	0
Defects Fixed	6	0

7) https://github.com/tarikkdiry/Triangle567

Summary:

This assignment forced me to look at a project as a tester rather than just a plain developer for the first time. I can now see the importance of meticulously documenting

tests and their respective statuses. The only real technique I used in this assignment was TDD, in which I compared my code to the tests I created and went through each individual test case that I created. The data inputs were arbitrary numbers that are basic in geometry for first learning the difference between triangles, along with some invalid inputs to test the base cases of the program. No special tools were used this time. Scrutinizing each individual tests against the source code is what got the assignment done. What didn't work was testing with random numbers without confirming the correct answers for them beforehand, causing some issues.

NOTE: The test file won't let me run more than exactly 8 tests, no matter what I tried.