SSW-567 HW-2A: Testing a legacy program and reporting on testing results.

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 y,ou 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.

Author: Dhruv Patel

First, I decided to write a few test cases which are to be tested on the Trianglr.py file using Unittest. The results are represented in the table below:

Initial Testing:

Test ID	Input	Expected Results	Actual Results	Pass or Fail
testInvalidInputA	-1, -1, -1	InvalidInput	InvalidInput	Pass
testInvalidInputB	“2”, “10”, “10”	InvalidInput	Type Error	Fail
testNotATriangleA	6, 1, 1	NotATriangle	InavlidInput	Fail
testNotATriangleB	1, 5, 2	NotATriangle	InavlidInput	Fail
testNotATriangleC	1, 6, 15	NotATriangle	InavlidInput	Fail
testNotATriangleD	1, 17, 5	NotATriangle	InavlidInput	Fail
testEquilateralTriangleA	2, 2, 2	Equilateral	InvalidInput	Fail
testEquilateralTriangleB	**5, 1, 5**	**Equilateral**	**Equilateral**	**Pass**
testEquilateralTriangleC	25, 25, 25	Equilateral	InvalidInput	Fail

SSW-567 HW-2A: Testing a legacy program and reporting on testing results.

testEquilateralTriangleD	1, 1, 1	Equilateral	InvalidInput	Fail
testRightTriangleA	3, 4, 5	Right	InvalidInput	Fail
testRightTriangleB	5, 3, 4	Right	InvalidInput	Fail
testRightTriangleC	8, 6, 10	Right	InvalidInput	Fail
testRightTriangleD	**10, 7, 25**	**Right**	**Right**	**Pass**
testScaleneTrianlgeA	4, 5, 6	Scalene	InvalidInput	Fail
testScaleneTrianlgeB	90, 100, 110	Scalene	InvalidInput	Fail
testScaleneTrianlgeC	**90, 90, 110**	**Scalene**	**Scalene**	**Pass**
testScaleneTrianlgeD	12, 13, 15	Scalene	InvalidInput	Fail
testIsocelestriangleA	5, 5, 13	Isoceles	InvalidInput	Fail
testIsocelestriangleB	4, 4, 26	Isoceles	InvalidInput	Fail
testIsocelestriangleC	**8, 5, 9**	**Isoceles**	**Isoceles**	**Pass**
testIsocelestriangleD	16, 16, 4	Isoceles	InvalidInput	Fail

Test Run Matrix:

Tests Planned	Test Run 1	Test Run 2	Test Run 3	Test run 4
Tests Executed	22	22	22	22
Tests Passed	5	6	16	22
Defects Found	1	3	2	0
Defects Fixed	0	1	3	2

After completing four test runs I was able to fix the problems/bugs present in Triangle.py. The results of the final test are given below.

Final Testing:

testInvalidInputA	-1, -1, -1	InvalidInput	InvalidInput	Pass
testInvalidInputB	“2”, “10”, “10”	InvalidInput	InvalidInput	Pass
testNotATriangleA	6, 1, 1	NotATriangle	NotATriangle	Pass
testNotATriangleB	1, 5, 2	NotATriangle	NotATriangle	Pass
testNotATriangleC	1, 6, 15	NotATriangle	NotATriangle	Pass
testNotATriangleD	1, 17, 5	NotATriangle	NotATriangle	Pass
testEquilateralTriangleA	2, 2, 2	Equilateral	Equilateral	Pass
testEquilateralTriangleB	5, 1, 5	Equilateral	Equilateral	Pass
testEquilateralTriangleC	25, 25, 25	Equilateral	Equilateral	Pass
testEquilateralTriangleD	1, 1, 1	Equilateral	Equilateral	Pass
testRightTriangleA	3, 4, 5	Right	Right	Pass
testRightTriangleB	5, 3, 4	Right	Right	Pass
testRightTriangleC	8, 6, 10	Right	Right	Pass
testRightTriangleD	10, 7, 25	Right	Right	Pass

SSW-567 HW-2A: Testing a legacy program and reporting on testing results.

testScaleneTrianlgeA	4, 5, 6	Scalene	Scalene	Pass
testScaleneTrianlgeB	90, 100, 110	Scalene	Scalene	Pass
testScaleneTrianlgeC	90, 90, 110	Scalene	Scalene	Pass
testScaleneTrianlgeD	12, 13, 15	Scalene	Scalene	Pass
testIsocelestriangleA	5, 5, 13	Isosceles	Isosceles	Pass
testIsocelestriangleB	4, 4, 26	Isosceles	Isosceles	Pass
testIsocelestriangleC	8, 5, 9	Isosceles	Isosceles	Pass
testIsocelestriangleD	16, 16, 4	Isosceles	Isosceles	Pass

As I tested the code again and again I was able to find the defects present and fix all the defects in my 4^{th} test runs. Therefore, I think Test-Driven debugging is an effective method to fix buggy codes. But I think that one should write the tests as you code rather than writing the tests after the code is completed.

Pledge:

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.