Alan Atrach, Cesar Espejo

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.

Triangle.py contains an implementation of the classifyTriangle() function with a few bugs.

TestTriangle.py contains the initial set of test cases

Alan Atrach, Cesar Espejo

GitHub Repo: https://github.com/cespejo15/Triangle123

Summary:

Test ID	Input	Expected Results	Actual Results	Pass or Fail
1.0	None (Initial unchanged state).	All tests are expected to fail as the files given are said to be buggy.	All tests failed	Fail (failures = 3 out of 3 tests)
1.1	Changed testRightTriangleB to have different arguments, as side a is the longest side, but side c is supposed to be the hypotenuse for the Pythagorean theorem. Added 11 more tests including 2 for isosceles, 2 for scalene test, 1 more for equilateral test, 3 for not a triangle tests, and 3 for invalid input tests.	All tests are expected to fail except for the Invalid Input tests. This is because in Triangle.py, there is an "InvalidInput" return when b <= b, which would occur every time.	All tests failed except for the "InvalidInput" tests.	Fail (failures = 11 out of 14 tests)
2.0	None (Initial unchanged state of Triangle.py).	Same results as Test 1.1.	All tests failed except for the "InvalidInput" tests.	Fail (failures = 11 out of 14 tests)
2.1	Changed b<=b on line 34 to b<=0 Changed reach "* " on the right triangle check to the correct "**" for squaring the sides. Added "and b==c" for the equilateral triangle check.	All tests to pass. Might get some fails due to overlooked bugs.	3 more tests passed, but still got a majority of failures. All failures were a result of "NotATriangle" being returned.	Fail (failures = 8 out of 14 tests)

2.2	Fixed line 46 that checks the Triangle Inequality Theorem. Confirms that if the sum of any two sides is greater than the third side, it’s an invalid triangle.	All tests to pass. Maybe another bug or two left.	All tests passed	Pass

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5
Tests Planned	3	14	14	14	14
Tests Executed	3	14	14	14	14
Tests Passed	0	3	3	6	14
Defects Found	3	11	11	8	0
Defects Fixed	0	0	0	3	8

After iterating through each test run and updating the code after assessing each failure, we were able to eliminate all failures. This involved an initial test run of each file and identifying where the underlying problem was within the code. The main debugging of the code began in test 2.1 after we began fixing the classifyTriangle() function. The main bugs found were within the right triangle check, the equilateral triangle check, and the invalid triangle check. After correcting these parts of the main function, all tests passed and resulted in a successful classification of triangles.

After going through this iterative process, we learned how to effectively debug code. This includes understanding the logic of the code and breaking down where the issue was within a line of code. This also includes reading the output of the test runs and understanding why we got the outputs we did. Documenting changes and outputs of the code worked as it helped us recognize whether we were getting closer or farther from fixing the bugs.

Detailed Results:

This assignment was straightforward, and the summary highlights all the details.

We pledge our honor that we have abided by the Stevens Honor System. - Alan Atrach, Cesar Espejo