HW 02a - Testing a legacy program and reporting on testing results

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

Author: Karthik Darapaneni
GitHub Repo: https://github.com/karthikds55/SSW567

Summary: The code defines a classifyTriangle function that classifies triangles based on their side lengths as Equilateral, Right, Isosceles, Scalene, or Invalid. It includes a TestTriangles class using the unittest framework to validate various triangle cases, including right, equilateral, scalene, isosceles, and invalid triangles. The run_tests function executes these tests and displays the results.

For initial test set -

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangle1()	3,4,5	Right	InvalidInput	Fail
testRightTriangle2()	5,3,4	Right	InvalidInput	Fail
testRightTriangle3()	8,6,10	Right	InvalidInput	Fail
testRightTriangle4()	21,6,10	InvalidInputOne	InvalidInput	Fail
testEquilateralTriangle1()	1,0,1	Equilateral	InvalidInput	Fail
testEquilateralTriangle2()	30,30,30	Equilateral	Equilateral	Pass
testEquilateralTriangle4()	10,1,10	InvalidInputTwo	InvalidInputTwo	Pass
testIsoscelesTriangle4()	6,6,4	Isosceles	InvalidInput	Fail
testScaleneTriangle1()	9,10,11	Scalene	InvalidInput	Fail
testNotATriangle1()	2,2,8	NotATriangle	InvalidInput	Fail

For Final test set-

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangle1()	3,4,5	Right	Right	Pass
testRightTriangle2()	5,3,4	Right	Right	Pass
testRightTriangle3()	8,9,10	Right	Right	Pass
testRightTriangle4()	21,6,10	InvalidInputOne	InvalidInputOne	Pass
testEquilateralTriangle1()	1,1,1	Equilateral	Equilateral	Pass
testEquilateralTriangle2()	30,30,30	Equilateral	Equilateral	Pass
testScaleneTriangle1()	9,10,11	Scalene	Scalene	Pass
testNotATriangle1()	2,2,8	NotATriangle	NotATriangle	Pass

	Test Run 1	Test Run2	Test Run3
Tests Planned	8	8	8
Tests Executed	8	8	8
Tests Passed	3	4	8
Defects Found	5	4	0
Defects Fixed	0	1	4

- **Reflection – this is where you actually think about the assignment after it is over – what did you learn? What worked, what didn't?**
  Following are the learning from the assignment:
  - Proper implementation of unit testing on some other person's code.
  - Including the unit test for edge cases
  - Fixing buggy code through unit testing.

Conditions mentioned in the Triangle.py had issues so the program was working with few inputs but was failing for most of the inputs.

5. Honor pledge

6. Detailed results, if any:

- Details of any assumptions or constraints

made Assumption about the upper limit of side length.
Constraints put over the datatype i.e., int datatype.

- **A description of whatever data inputs you used**

Data inputs were used for every necessary conditions like
- float datatype, alphabet were used as data set for input condition
- Upper limit for the length of triangle was checked
- Lower limit for the length of triangle was checked
- Different values for NotATriangle condition
- Different values for different triangle conditions

- **An explanation of the results of your work. In this case, upload copies of the code, the sanity tests, and the results, plus any other relevant results about the tools selection/installation/usage.**

Final Test Set run:

```
testEquilateralTriangle1 (__main__.TestTriangles.testEquilateralTriangle1) ... ok
testEquilateralTriangle2 (__main__.TestTriangles.testEquilateralTriangle2) ... ok
testNotATriangle1 (__main__.TestTriangles.testNotATriangle1) ... ok
testRightTriangle1 (__main__.TestTriangles.testRightTriangle1) ... ok
testRightTriangle2 (__main__.TestTriangles.testRightTriangle2) ... ok
testRightTriangle3 (__main__.TestTriangles.testRightTriangle3) ... ok
testRightTriangle4 (__main__.TestTriangles.testRightTriangle4) ... ok
testScaleneTriangle1 (__main__.TestTriangles.testScaleneTriangle1) ... ok

----------------------------------------------------------------------
Ran 8 tests in 0.010s

OK
```

Initial Test Set Run1 :

```
testEquilateralTriangle1 (__main__.TestTriangles.testEquilateralTriangle1) ... FAIL
testEquilateralTriangle2 (__main__.TestTriangles.testEquilateralTriangle2) ... ok
testNotATriangle1 (__main__.TestTriangles.testNotATriangle1) ... FAIL
testRightTriangle1 (__main__.TestTriangles.testRightTriangle1) ... FAIL
testRightTriangle2 (__main__.TestTriangles.testRightTriangle2) ... FAIL
testRightTriangle3 (__main__.TestTriangles.testRightTriangle3) ... FAIL
testRightTriangle4 (__main__.TestTriangles.testRightTriangle4) ... ok
testScaleneTriangle1 (__main__.TestTriangles.testScaleneTriangle1) ... ok

======================================================================
FAIL: testEquilateralTriangle1 (__main__.TestTriangles.testEquilateralTriangle1)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\569975540.py", line 30, in testEquilateralTriangle1
    self.assertEqual(classifyTriangle(1,0,1),'Equilateral','1,1,1 should be equilateral')
AssertionError: 'Scalene' != 'Equilateral'
- Scalene
+ Equilateral
 : 1,1,1 should be equilateral

======================================================================
FAIL: testNotATriangle1 (__main__.TestTriangles.testNotATriangle1)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\569975540.py", line 41, in testNotATriangle1
    self.assertEqual(classifyTriangle(2, 2, 8), 'NotATriangle')
AssertionError: 'Isosceles' != 'NotATriangle'
- Isosceles
+ NotATriangle


======================================================================
FAIL: testRightTriangle1 (__main__.TestTriangles.testRightTriangle1)
----------------------------------------------------------------------
Traceback (most recent call last):
```

```
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\569975540.py", line 17, in testRightTriangle1
    self.assertEqual(classifyTriangle(3,4,5),'Right','3,4,5 is a Right triangle')
AssertionError: 'Scalene' != 'Right'
- Scalene
+ Right
 : 3,4,5 is a Right triangle

======================================================================
FAIL: testRightTriangle2 (__main__.TestTriangles.testRightTriangle2)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\569975540.py", line 20, in testRightTriangle2
    self.assertEqual(classifyTriangle(5,3,4),'Right','5,3,4 is a Right triangle')
AssertionError: 'Scalene' != 'Right'
- Scalene
+ Right
 : 5,3,4 is a Right triangle

======================================================================
FAIL: testRightTriangle3 (__main__.TestTriangles.testRightTriangle3)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\569975540.py", line 23, in testRightTriangle3
    self.assertEqual(classifyTriangle(8,6,10),'Right','8,6,10 is a Right triangle')
AssertionError: 'Scalene' != 'Right'
- Scalene
+ Right
 : 8,6,10 is a Right triangle

----------------------------------------------------------------------
Ran 8 tests in 0.011s

FAILED (failures=5)
```

Initial Test Set Run 2:

```
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\804240640.py", line 17, in testRightTriangle1
    self.assertEqual(classifyTriangle(3,4,5),'Right','3,4,5 is a Right triangle')
AssertionError: 'Scalene' != 'Right'
- Scalene
+ Right
 : 3,4,5 is a Right triangle

======================================================================
FAIL: testRightTriangle2 (__main__.TestTriangles.testRightTriangle2)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\804240640.py", line 20, in testRightTriangle2
    self.assertEqual(classifyTriangle(3,4,5),'Right','5,3,4 is a Right triangle')
AssertionError: 'Scalene' != 'Right'
- Scalene
+ Right
 : 5,3,4 is a Right triangle

======================================================================
FAIL: testRightTriangle3 (__main__.TestTriangles.testRightTriangle3)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Legion\AppData\Local\Temp\ipykernel_35672\804240640.py", line 23, in testRightTriangle3
    self.assertEqual(classifyTriangle(8,9,10),'Right','8,6,10 is a Right triangle')
AssertionError: 'Scalene' != 'Right'
- Scalene
+ Right
 : 8,6,10 is a Right triangle

----------------------------------------------------------------------
Ran 8 tests in 0.009s

FAILED (failures=4)
```