1. Description of assignment:

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 unit test 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.

2. **Author**: Arun Rao Nayineni

GitHub: https://github.com/ArunRao1997/SSW_567_HW_02a_Triangle

3. **Summary**

Initial Test Results:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	3, 4, 5	Right	InvalidInput	Fail
testRightTriangleB	5, 3, 4	Right	InvalidInput	Fail
testEquilateralTriangleA	1, 1, 1	Equilateral	InvalidInput	Fail
testEquilateralTriangleB	7, 7, 7	Equilateral	InvalidInput	Fail
testScaleneTriangleA	10, 11, 12	Scalene	InvalidInput	Fail
testScaleneTriangleB	100, 110, 112	Scalene	InvalidInput	Fail
testIsoscelesTriangleA	5, 5, 3	Isosceles	InvalidInput	Fail
testIsoscelesTriangleB	4, 6, 6	Isosceles	InvalidInput	Fail
testIsoscelesTriangleC	8, 6, 8	Isosceles	InvalidInput	Fail
testIsoscelesTriangleD	6, 6, 4	Isosceles	InvalidInput	Fail
testInvalidInput1	-1, -1, -1	InvalidInput	InvalidInput	Pass
testInvalidInput2	201, 201, 201	InvalidInput	InvalidInput	Pass
testInvalidInput3	"200", "200", "200"	InvalidInput	InvalidInput	Fail
testNotATriangle1	1, 3, 5	NotATriangle	InvalidInput	Fail
testNotATriangle2	1, 4, 5	NotATriangle	InvalidInput	Fail
testNotATriangle3	1, 17, 5	NotATriangle	InvalidInput	Fail

Test Run Matrix:

	Test Run 1	Test Run 2	Test Run 4	Test Run 4
Tests Planned	16	16	16	16
Tests Executed	16	16	16	16
Tests Passed	2	6	12	16
Defects Found	2	1	5	0
Defects Fixed	0	2	3	3

Final test results:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	3, 4, 5	Right angle and Scalene	Right angle and Scalene	Pass
testRightTriangleB	8, 6, 10	Right angle and Scalene	Right angle and Scalene	Pass
testEquilateralTriangleA	1, 1, 1	Equilateral	Equilateral	Pass
testEquilateralTriangleB	7, 7, 7	Equilateral	Equilateral	Pass
testScaleneTriangleA	3, 4, 6	Scalene	Scalene	Pass
testScaleneTriangleB	100, 110, 112	Scalene	Scalene	Pass
testIsoscelesTriangleA	5, 5, 3	Isosceles	Isosceles	Pass
testIsoscelesTriangleB	4, 6, 6	Isosceles	Isosceles	Pass
testIsoscelesTriangleC	8, 6, 8	Isosceles	Isosceles	Pass
testIsoscelesTriangleD	6, 6, 4	Isosceles	Isosceles	Pass
testInvalidInput1	-1, -1, -1	InvalidInput	InvalidInput	Pass
testInvalidInput2	201, 201, 201	InvalidInput	InvalidInput	Pass
testInvalidInput3	"200", "200", "200"	InvalidInput	InvalidInput	Pass
testNotATriangle1	1, 3, 5	NotATriangle	NotATriangle	Pass
testNotATriangle2	1, 4, 5	NotATriangle	NotATriangle	Pass
testNotATriangle3	1, 17, 5	NotATriangle	NotATriangle	Pass

I think **Test-driven development** is a very efficient way to rectify incorrect code. More errors were found as I ran the tests and fixed issues in the code.

The idea behind TDD is that the tests serve as specifications for the code and catch bugs early in the development process.

I think that the order of writing tests first and then writing the code that passes the tests can be a more effective way to catch bugs than writing the code first and then writing tests after the fact.

4. **Honor pledge:**

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

Detailed results:

I considered the possibility that the right-angled can also be scalene or isosceles. Since it is mentioned in the problem statement to consider only the integer values, I have ruled out the possibility that it is isosceles.

I have written the code that handles to check if it is a right-angled as well as scalene or just the scalene. I have written the test cases for the right angle, right angle and scalene, and scalene.

I have also validated and written the test cases for invalid inputs and sides that are not a triangle. Similarly, I've validated test cases for equilateral i.e., all three sides are equal, and isosceles i.e., either of the two sides is equal.

Test Sample Output:

```
Launching pytest with arguments C:/Users/Arun Rao Nayineni/SSW_567_HW_02a_Triangle/TestTriangle.py --no-header --no-summary -q in C:\Users\Arun Rao Nayineni\SSW_567_HW_02a_Triangle


============================= test session starts =============================

collecting ... collected 16 items


TestTriangle.py::TestTriangles::testEquilateralTriangleA PASSED          [  6%]

TestTriangle.py::TestTriangles::testEquilateralTriangleB PASSED          [ 12%]

TestTriangle.py::TestTriangles::testInvalidInput1 PASSED                 [ 18%]

TestTriangle.py::TestTriangles::testInvalidInput2 PASSED                 [ 25%]

TestTriangle.py::TestTriangles::testInvalidInput3 PASSED                 [ 31%]

TestTriangle.py::TestTriangles::testIsoscelesTriangleA PASSED            [ 37%]

TestTriangle.py::TestTriangles::testIsoscelesTriangleB PASSED            [ 43%]

TestTriangle.py::TestTriangles::testIsoscelesTriangleC PASSED            [ 50%]

TestTriangle.py::TestTriangles::testIsoscelesTriangleD PASSED            [ 56%]

TestTriangle.py::TestTriangles::testNotATriangle1 PASSED                 [ 62%]

TestTriangle.py::TestTriangles::testNotATriangle2 PASSED                 [ 68%]

TestTriangle.py::TestTriangles::testNotATriangle3 PASSED                 [ 75%]

TestTriangle.py::TestTriangles::testRightTriangleA PASSED                [ 81%]

TestTriangle.py::TestTriangles::testRightTriangleAndScalene PASSED       [ 87%]

TestTriangle.py::TestTriangles::testScaleneTriangleA PASSED              [ 93%]

TestTriangle.py::TestTriangles::testScaleneTriangleB PASSED              [100%]


============================== 16 passed in 0.05s ==============================


Process finished with exit code 0
```