1. **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.

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.

2. **Author**: Ruthy Levi

3. **Summary**:

- a summary of your results: https://github.com/ruthylevi/Triangle567

- reflection -- this is where you actually think about the assignment after it is over -- what did you learn? What worked, what didn't?:

I learned how it can be useful to make tests first before writing code. This way, you know what you want your code to look like. When I ran the tests I made along with the ones there against the buggy code, it really helped me pinpoint the bugs. I do prefer the style where you test and code back and forth more however; instead of doing one first.

5. **Honor pledge:** I pledge my honor that I have abided by the Stevens Honor System *--Ruthy Levi.*

6. **Detailed results**

Initial Test Report:

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	(5,5,5)	Equilateral	InvalidInput	Fail
testScaleneTriangleA	(6,8,11)	Scalene	InvalidInput	Fail
testScaleneTriangleB	(12,16,22)	Scalene	InvalidInput	Fail
testIsoscelesTriangleA	(5,5,4)	Isosceles	InvalidInput	Fail
testIsoscelesTriangleB	(10,10,8)	Isosceles	InvalidInput	Fail
testInvalidTriangleA	(250,250, 300)	InvalidInput	InvalidInput	Pass

testInvalidTriangleB	(“5”,”10”,” 8”)	InvalidInput	TypeError	Fail
testInvalidTriangleC	(-5,-5,-5)	InvalidInput	InvalidInput	Pass
testInvalidTriangleD	(5,6,20)	NotATriangle	InvalidInput	Fail

Test Runs:

	Test Run 1	Test Run2
Tests Planned	12	12
Tests Executed	12	12
Tests Passed	10	12
Defects Found	2	0
Defects Fixed	2	0

Final Test Report:

Test ID	Input	Expected Results	Actual Result	Pass or Fail

testRightTriangleA	(3,4,5)	Right	Right	Pass
testRightTriangleB	(5,3,4)	Right	Right	Pass
testEquilateralTriangleA	(1,1,1)	Equilateral	Equilateral	Pass
testEquilateralTriangleB	(5,5,5)	Equilateral	Equilateral	Pass
testScaleneTriangleA	(6,8,10)	Scalene	Scalene	Pass
testScaleneTriangleB	(12,16,20)	Scalene	Scalene	Pass
testIsoscelesTriangleA	(5,5,4)	Isosceles	Isosceles	Pass
testIsoscelesTriangleB	(10,10,8)	Isosceles	Isosceles	Pass
testInvalidTriangleA	(250,250, 300)	InvalidInput	InvalidInput	Pass
testInvalidTriangleB	(“5”,”10”,” 8”)	InvalidInput	InvalidInput	Pass
testInvalidTriangleC	(-5,-5,-5)	InvalidInput	InvalidInput	Pass
testInvalidTriangleD	(5,6,20)	NotATriangle	NotATriangle	Pass

The data inputs I used were equivalent to whatever I was testing. For instance, if I were testing for an equilateral triangle, I would check if the input’s result matched ‘Equilateral’.