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.

- 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 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**: Emay Pandarakutty

3. **Summary**:

Test are identified to cover all the entry and exit point of each condition along with data boundary test with the addition of different datatype handling part, below table list all the identified test with inputs and expected outputs. Test was run on the original ***Triangle.py***.

Observed test results are captured and marked as FAIL or PASS as blow as Initial Test Results:

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
testRightTriangleC	(3,5,4)	'Right'	InvalidInput'	FAIL
testEquilateralTriangles	(1,1,1)	'Equilateral'	InvalidInput'	FAIL
testScaleneTriangles	(4,5,6)	'Scalene'	InvalidInput'	FAIL
testIsoscelesTrianglesAB	(7,7,8)	'Isoceles'	InvalidInput'	FAIL
testIsoscelesTrianglesAC	(7,8,7)	'Isoceles'	InvalidInput'	FAIL
testIsoscelesTrianglesBC	(8,7,7)	'Isoceles'	InvalidInput'	FAIL
testFloatInputA	(1.23,5,6)	'InvalidInput'	'InvalidInput'	PASS
testFloatInputB	(1,5.3,6)	'InvalidInput'	'InvalidInput'	PASS
testFloatInputC	(1,5,6.3)	'InvalidInput'	'InvalidInput'	PASS
testStringInputA	("a",5,6)	'InvalidInput'	error	FAIL
testStringInputB	(1,"b",6)	'InvalidInput'	error	FAIL
testStringInputC	(1,5,"c")	'InvalidInput'	error	FAIL
testMaxPlus1A	(201,4,5)	'InvalidInput'	'InvalidInput'	PASS
testMaxPlus1B	(3,201,5)	'InvalidInput'	'InvalidInput'	PASS
testMaxPlus1C	(3,4,201)	'InvalidInput'	'InvalidInput'	PASS
testMaxA	(200,198,199)	'Scalene'	InvalidInput'	FAIL
testMaxB	(199,200,198)	'Scalene'	InvalidInput'	FAIL
testMaxC	(198,199,200)	'Scalene'	InvalidInput'	FAIL
testMinA	(0,4,5)	'InvalidInput'	'InvalidInput'	PASS
testMinB	(3,0,5)	'InvalidInput'	'InvalidInput'	PASS
testMinC	(3,4,0)	'InvalidInput'	'InvalidInput'	PASS
testMinMinus1A	(-1,4,5)	'InvalidInput'	'InvalidInput'	PASS
testMinMinus1B	(3,-1,5)	'InvalidInput'	'InvalidInput'	PASS
testMinMinus1C	(3,4,-1)	'InvalidInput'	'InvalidInput'	PASS
testNotaTriangleA	(16,8,7)	'NotATriangle'	InvalidInput'	FAIL
testNotaTriangleB	(7,16,8)	'NotATriangle'	InvalidInput'	FAIL
testNotaTriangleC	(7,8,16)	'NotATriangle'	InvalidInput'	FAIL

Learning from the first test results is analyzed and ***Triangle.py*** is enhanced/corrected for the logical error/bugs. Tests are executed again on the enhanced ***Triangle.py*** and this process is repeated until the all the test are passed. Results are listed below.

Final Test Results:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
testRightTriangleA	(3,4,5)	'Right'	'Right'	PASS
testRightTriangleB	(5,3,4)	'Right'	'Right'	PASS
testRightTriangleC	(3,5,4)	'Right'	'Right'	PASS
testEquilateralTriangles	(1,1,1)	'Equilateral'	'Equilateral'	PASS
testScaleneTriangles	(4,5,6)	'Scalene'	'Scalene'	PASS
testIsoscelesTrianglesAB	(7,7,8)	'Isoceles'	'Isoceles'	PASS
testIsoscelesTrianglesAC	(7,8,7)	'Isoceles'	'Isoceles'	PASS
testIsoscelesTrianglesBC	(8,7,7)	'Isoceles'	'Isoceles'	PASS
testFloatInputA	(1.23,5,6)	'InvalidInput'	'InvalidInput'	PASS
testFloatInputB	(1,5.3,6)	'InvalidInput'	'InvalidInput'	PASS
testFloatInputC	(1,5,6.3)	'InvalidInput'	'InvalidInput'	PASS
testStringInputA	("a",5,6)	TypeError'	'TypeError'	PASS
testStringInputB	(1,"b",6)	'TypeError'	'TypeError'	PASS
testStringInputC	(1,5,"c")	'TypeError'	'TypeError'	PASS
testMaxPlus1A	(201,4,5)	'InvalidInput'	'InvalidInput'	PASS
testMaxPlus1B	(3,201,5)	'InvalidInput'	'InvalidInput'	PASS
testMaxPlus1C	(3,4,201)	'InvalidInput'	'InvalidInput'	PASS
testMaxA	(200,198,199)	'Scalene'	'Scalene'	PASS
testMaxB	(199,200,198)	'Scalene'	'Scalene'	PASS
testMaxC	(198,199,200)	'Scalene'	'Scalene'	PASS
testMinA	(0,4,5)	'InvalidInput'	'InvalidInput'	PASS
testMinB	(3,0,5)	'InvalidInput'	'InvalidInput'	PASS
testMinC	(3,4,0)	'InvalidInput'	'InvalidInput'	PASS
testMinMinus1A	(-1,4,5)	'InvalidInput'	'InvalidInput'	PASS
testMinMinus1B	(3,-1,5)	'InvalidInput'	'InvalidInput'	PASS
testMinMinus1C	(3,4,-1)	'InvalidInput'	'InvalidInput'	PASS
testNotaTriangleA	(16,8,7)	'NotATriangle'	'NotATriangle'	PASS
testNotaTriangleB	(7,16,8)	'NotATriangle'	'NotATriangle'	PASS
testNotaTriangleC	(7,8,16)	'NotATriangle'	'NotATriangle'	PASS

Test Matrix:

Results of the primary execution is and fix all the logical errors identified in ***Triangle.py***. believing fixing the logical error clears all the test, but observed 3 tests where still failing, this time not only logic added to the ***Triangle.py***, but test expected output also updated to capture the ***Triangle.py*** update, which cleared all the listed failures. See the test iteration matrix below

	Test Run 1	Test Run2	Test Run3
Tests Planned	29	29	29
Tests Executed	29	29	29
Tests Passed	12	26	29
Defects Found	17	3	0
Defects Fixed	0	14	3

Assumptions/limitations:

1. Test are written by assuming datatype can be string, float or int.
2. Only TypeError exception is handle at the higher level. ***Triangle.py***. could be improved handle all possible exception.
3. Test is driven from code is interesting, HW01 requirement was used in framing most of the test.

Repository: https://github.com/Emay-Pandarakutty/Triangle567

4. Honor pledge:

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.