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 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: Cheng Tian

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.

Summary：

From this homework, I found that I need to write tests first, and then debug code. In this way I can know if my code can pass my precondition.

6. **Detailed results, if any:**

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5
Tests Planned	testInvalidInputA	testInvalidInputB	testInvalidInputC	testTrianglesA	testTrianglesB
Tests Executed	assertEqual(classifyTriangle(1000,2,3),'InvalidInput','1000,2,3 is a	assertEqual(classifyTriangle(1,-2,3),'InvalidInput','1,-2,3 is a	assertEqual(classifyTriangle(1,2.0,3),'InvalidInput','1,-2,3 is a	assertEqual(classifyTriangle(1,2,3),'NotATriangle','1,2,3 is a	assertEqual(classifyTriangle(199,100,3),'NotATriangle','199,100,3 is a

	InvalidInput')	InvalidInput')	InvalidInput')	NotATriangle')	NotATriangle')
Tests Passed	Passed	Passed	Passed	FAIL	FAIL
Defects Found	No	No	No	Input is an valid input but show invalidinput	Input is an valid input but show invalidinput
Defects Fixed	No	No	No	Line 31 b <= b b <=0	Line 31 b <= b b <=0

	Test Run 6	Test Run 7	Test Run 8	Test Run 9	Test Run 10
Tests Planned	testRightTriangleA	testRightTriangleB	testRightTriangleC	testEquilateralTriangIesA	testEquilateralTriangIesB

Tests Executed	assertEqual(classifyTriangle(3,4,5),'Right','3,4,5 is a Right triangle')	assertEqual(classifyTriangle(5,3,4),'Right','5,3,4 is a Right triangle')	assertEqual(classifyTriangle(5,12,13),'Right','5,12,13 is a Right triangle')	assertEqual(classifyTriangle(1,1,1),'Equilateral','1,1,1 should be equilateral')	assertEqual(classifyTriangle(2,2,2),'Equilateral','2,2,2 should be equilateral')
Tests Passed	FAIL	FAIL	Passed	Passed	Passed
Defects Found	(a * 2) + (b * 2)) == (c * 2)	(a > (b - c)) or (b > (a - c)) or (c > (a + b))	No	No	No
Defects Fixed	((a ** 2) + (b ** 2)) == (c ** 2) or ((b ** 2) + (c ** 2)) == (a ** 2) or ((a ** 2) + (c ** 2)) == (b ** 2)	not((a+b>c) and (b+c>a) and (a+b>c))	No	No	No

	Test Run 11	Test Run 12	Test Run 13	Test Run 14	Test Run 15	Test Run 16
Tests Planned	testScaleneTrianglesA	testScaleneTrianglesB	testScaleneTrianglesC	testIsoscelesTrianglesA	testIsoscelesTrianglesB	testIsoscelesTrianglesC
Tests Executed	assertEqual(classifyTriangle(3,5,6),'Scalene','3,5,6 should be Scalene')	assertEqual(classifyTriangle(5,6,3),'Scalene','5,6,3 should be Scalene')	assertEqual(classifyTriangle(200,199,198),'Scalene','200,199,198 should be Scalene')	assertEqual(classifyTriangle(3,3,4),'Isoceles','3,3,4 should be Isoceles')	assertEqual(classifyTriangle(4,3,3),'Isoceles','4,3,3 should be Isoceles')	assertEqual(classifyTriangle(3,4,3),'Isoceles','3,4,3 should be Isoceles')
Tests Passed	Passed	Passed	Passed	FAIL	Passed	FAIL
Defects Found	No	No	No	3,3,4 should be Isoceles	No	(a != b) and (b != c) and (a != b)

Defects Fixed	No	No	No	a == b and b == c and a == c	No	(a != b) and (b != c) and (a != c):