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.

Author: Michael Moreno

Summary:

Initial Test Cases

Test ID	Input	Expected Results	Actual Results	Pass or Fail
1	3,4,5	Right	InvalidInput	Fail
2	5,3,4	Right	InvalidInput	Fail
3	1,1,1	Equilateral	InvalidInput	Fail

Final Test Cases

Test ID	**Input**	**Expected Results**	**Actual Results**	**Pass or Fail**
1	3,4,5	Right	Right	Pass
2	5,3,4	Right	Right	Pass
3	1,1,1	Equilateral	Equilateral	Pass
4	10,10,12	Isosceles	Isosceles	Pass
5	1,2,3	Scalene	Scalene	Pass
6	4,-1,3	InvalidInput	InvalidInput	Pass
7	5,1,12	NotATriangle	NotATriangle	Pass

Testing Matrix

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9
Tests Planned	3	7	7	7	7	7	7	7	7
Tests Executed	3	7	7	7	7	7	7	7	7
Tests Passed	0	0	0	0	1	2	3	5	7
Defects Found	-	-	1	2	1	1	3	2	0
Defects Fixed	0	0	1	2	1	1	3	2	0

This assignment was useful in gaining the experience of fixing a broken product. The code provided did not pass any test, I worked backwards starting with test cases in a different workflow from building a product from the ground-up. Trying to brute force all the logic and bug fixes at once after setting up the remaining test cases was not efficient as it resulted in missing clear problems and extended the amount of time that I took in fixing the code.

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

6. An assumption of strictly integer inputs was assumed since it was already implemented into the code we were given.

The values for the additional test cases were tested against general mathematical rules for triangles before implementation into the test cases.

The results show that by analyzing error messages from the unit tests can result in a swift resolution to issues within legacy code. Code is provided below.

Triangle.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 14 13:44:00 2016
Updated Jan 21, 2018

The primary goal of this file is to demonstrate a simple python program to
classify triangles

@author: jrr
@author: rk
"""

def classifyTriangle(a,b,c):
    """
    Your correct code goes here...  Fix the faulty logic below until the code
passes all of
    you test cases.

    This function returns a string with the type of triangle from three integer
values
    corresponding to the lengths of the three sides of the Triangle.

    return:
        If all three sides are equal, return 'Equilateral'
        If exactly one pair of sides are equal, return 'Isoceles'
        If no pair of  sides are equal, return 'Scalene'
        If not a valid triangle, then return 'NotATriangle'
        If the sum of any two sides equals the squate of the third side, then
return 'Right'

      BEWARE: there may be a bug or two in this code
    """

    # require that the input values be >= 0 and <= 200
    if a > 200 or b > 200 or c > 200:
        return 'InvalidInput'

    if a <= 0 or b <= 0 or c <= 0:
        return 'InvalidInput'

    # verify that all 3 inputs are integers
    # Python's "isinstance(object,type) returns True if the object is of the
specified type
```

```
    if not(isinstance(a,int) and isinstance(b,int) and isinstance(c,int)):
        return 'InvalidInput';

    # This information was not in the requirements spec but
    # is important for correctness
    # the sum of any two sides must be strictly less than the third side
    # of the specified shape is not a triangle
    if (a > (b + c)) or (b > (a + c)) or (c > (a + b)):
        return 'NotATriangle'

    # now we know that we have a valid triangle
    if a == b == c:
        return 'Equilateral'
    elif ((a ** 2) + (b ** 2)) == (c ** 2) or (a ** 2) + (c ** 2) == (b ** 2) or
(b ** 2) + (c ** 2) == (a ** 2):
        return 'Right'
    elif (a != b) and  (b != c) and (a != c):
        return 'Scalene'
    else:
        return 'Isosceles'
```

TestTriangle.py

```
# -*- coding: utf-8 -*-
"""
Updated Jan 21, 2018
The primary goal of this file is to demonstrate a simple unittest implementation

@author: jrr
@author: rk
"""

import unittest

from Triangle import classifyTriangle

# This code implements the unit test functionality
# https://docs.python.org/3/library/unittest.html has a nice description of the
framework

class TestTriangles(unittest.TestCase):
    # define multiple sets of tests as functions with names that begin
```

```
    def testRightTriangleA(self):
        self.assertEqual(classifyTriangle(3, 4, 5),'Right','3,4,5 is a Right
triangle')

    def testRightTriangleB(self):
        self.assertEqual(classifyTriangle(5, 3, 4),'Right','5,3,4 is a Right
triangle')

    def testEquilateralTriangles(self):
        self.assertEqual(classifyTriangle(1, 1, 1),'Equilateral','1,1,1 should be
equilateral')

    def testIsoscelesTriangles(self):
        self.assertEqual(classifyTriangle(10, 10, 12), 'Isosceles', '10,10,12 is
an Isosceles triangle')

    def testScaleneTriangles(self):
        self.assertEqual(classifyTriangle(1, 2, 3), 'Scalene', '1,2,3 is a
Scalene triangle')

    def testInvalidInput(self):
        self.assertEqual(classifyTriangle(4, -1, 3), 'InvalidInput', '4,-1,3 is
an ivalid input')

    def testNotTriangle(self):
        self.assertEqual(classifyTriangle(5, 1, 12), 'NotATriangle', '5,1,12 is
an not a triangle')


if __name__ == '__main__':
    print('Running unit tests')
    unittest.main()
```