Use link to read comments

SSW567: HW 2

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 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

Author: Cameron Conway

Summary:

a. -I was able to find 7 issues in the code in the system. Below I show the changes and justify the corrections. The comments will explain each change and why it was made. Look under **Code Compare**.
b. These tasks are more difficult for me than others. It wasn't obvious for me at first to have to go discover the bugs and understand what the legacy code was

attempting to achieve. The majority of my practice has been with stakeholders, where I get to ask them how the code should function. Trying to get inside the thoughts of clients and prior programmers is a great skill set that individuals will value. These extras assisted me in practicing such circumstances.

Adjusted Code

```
# -*- coding: utf-8 -*-
def classifyTriangle(a,b,c):
    """
    Your correct code goes here...  Fix the faulty logic below until the code passes all ofyou 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 and b == a and a==c:
     return 'Equilateral'
  elif ((a ** 2) + (b ** 2)) == (c ** 2) or ((b** 2) + (c ** 2)) == (a ** 2) or((c ** 2) + (b ** 2))
== (b ** 2) :
     return 'Right'
  elif (a != b) and  (b != c) and (a != c):
     return 'Scalene'
  else:
     return 'Isosceles'
```

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

Code Compared

1	def classifyTriangle(a,b,c):	1	def classifyTriangle(a,b,c):
2		2	
3	# require that the input values be >= 0 and <= 200	3	# require that the input values be >= 0 and <= 200
4	if a > 200 or b > 200 or c > 200:	4	if a > 200 or b > 200 or c > 200:
5	return 'InvalidInput'	5	return 'InvalidInput'
6		6	
7	if a < 0 or b < 0 or c < 0:	7	if a <= 0 or b <= b or c <= 0:
8	return 'InvalidInput'	8	return 'InvalidInput'
9		9	
10	# verify that all 3 inputs are integers	10	# verify that all 3 inputs are integers
11	# Python's "isinstance(object,type) returns True if the object is of the specified type	11	# Python's "isinstance(object,type) returns True if the object is of the
		12	specified type
12	if not(isinstance(a,int) and isinstance(b,int) and isinstance(c,int)):	13	if not(isinstance(a,int) and isinstance(b,int) and isinstance(c,int)):
13	return 'InvalidInput';	14	return 'InvalidInput';
14		15	
15	# This information was not in the requirements spec but	16	# This information was not in the requirements spec but
16	# is important for correctness	17	# is important for correctness
17	# the sum of any two sides must be strictly less than the third side	18	# the sum of any two sides must be strictly less than the third side
18	# of the specified shape is not a triangle	19	# of the specified shape is not a triangle
19	if (a >= (b + c)) or (b >= (a + c)) or (c >= (a + b)):	20	if (a >= (b - c)) or (b >= (a - c)) or (c >= (a + b)):
20	return 'NotATriangle'	21	return 'NotATriangle'

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

21		22	
22	# now we know that we have a valid triangle	23	# now we know that we have a valid triangle
23	if a == b and b == a and a==c:	24	if a == b and b == a:
24	return 'Equilateral'	25	return 'Equilateral'
25	elif ((a ** 2) + (b ** 2)) == (c ** 2) or ((b** 2) + (c ** 2)) == (a ** 2) or((c ** 2) + (b ** 2)) == (b ** 2) :	26	elif ((a * 2) + (b * 2)) == (c * 2):
26	return 'Right'	27	return 'Right'
27	elif (a != b) and (b != c) and (a != c):	28	elif (a != b) and (b != c) and (a != b):
28	return 'Scalene'	29	return 'Scalene'
29	else:	30	else:
30	return 'Isosceles'	31	return 'Isosceles'

Test Error Cases by Technician

	Test Run 1	Test Run 2	Test Run 3	Test Run 4
Tests Planned	" First Run "	" TestTriangle.py"	" TestTriangle.py"	" TestTriangle.py"
Tests Executed	Unit Test	Unit Test	Unit Test	Unit Test
Tests Passed	0	1	2	3
Defects Found	3	2	1	0

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

Error Codes	"Running unit tests FFF == FAIL: testEquilateralTriangles (__main__.TestTriangles) -- Traceback (most recent call last): File ""/Users/cjcnj2000/Library/CloudStorage/OneDrive-stevens.edu/Term VII/SSW 567/HW_2a/TestTriangle.py"", line 20, in testEquilateralTriangles self.assertEqual(classifyTriangle(1,1,1),'Equilateral','1,1,1 should be equilateral') AssertionError: 'InvalidInput' != 'Equilateral' - InvalidInput + Equilateral	Running unit tests .FF == FAIL: testRightTriangleA (__main__.TestTriangles) -- Traceback (most recent call last): File "/Users/cjcnj2000/Library/CloudStorage/OneDrive-stevens.edu/Term VII/SSW 567/HW_2a/TestTriangle.py", line 15, in testRightTriangleA self.assertEqual(classifyTriangle(3,4,5),'Right','3,4,5 is a Right triangle') AssertionError: 'Scalene' != 'Right' - Scalene + Right : 3,4,5 is a Right triangle == FAIL: testRightTriangleB	AILED (failures=2) cjcnj2000@Camerons-MacBook-Air HW_2a % python3 TestTriangle.py Running unit tests ..F == FAIL: testRightTriangleB (__main__.TestTriangles) -- Traceback (most recent call last): File "/Users/cjcnj2000/Library/CloudStorage/OneDrive-stevens.edu/Term VII/SSW 567/HW_2a/TestTriangle.py", line 17, in testRightTriangleB self.assertEqual(classifyTriangle(	Running unit tests ... -- Ran 3 tests in 0.000s OK

	: 1,1,1 should be equilateral == FAIL: testRightTriangleA (__main__.TestTriangles) -- Traceback (most recent call last): File ""/Users/cjcnj2000/Library/CloudStorage/OneDrive-stevens.edu/Term VII/SSW 567/HW_2a/TestTriangle.py"", line 15, in testRightTriangleA self.assertEqual(classifyTriangle(3,4,5),'Right','3,4,5 is a Right triangle') AssertionError: 'InvalidInput' != 'Right' - InvalidInput + Right : 3,4,5 is a Right triangle	(__main__.TestTriangles) -- Traceback (most recent call last): File "/Users/cjcnj2000/Library/CloudStorage/OneDrive-stevens.edu/Term VII/SSW 567/HW_2a/TestTriangle.py", line 17, in testRightTriangleB self.assertEqual(classifyTriangle(5,3,4),'Right','5,3,4 is a Right triangle') AssertionError: 'Scalene' != 'Right' - Scalene + Right : 5,3,4 is a Right triangle -- Ran 3 tests in 0.000s	5,3,4),'Right','5,3,4 is a Right triangle') AssertionError: 'Scalene' != 'Right' - Scalene + Right : 5,3,4 is a Right triangle -- Ran 3 tests in 0.000s FAILED (failures=1)	

	== FAIL: testRightTriangleB (__main__.TestTriangles) --- Traceback (most recent call last): File ""/Users/cjcnj2000/Library/CloudStorage/OneDrive-stevens.edu/Term VII/SSW 567/HW_2a/TestTriangle.py"", line 17, in testRightTriangleB self.assertEqual(classifyTriangle(5,3,4),'Right','5,3,4 is a Right triangle') AssertionError: 'InvalidInput' != 'Right' - InvalidInput + Right : 5,3,4 is a Right triangle ----------------------			

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

	----------------------- ----------------------- - Ran 3 tests in 0.000s FAILED (failures=3)"			
Defects Fixed	0	1	1	1

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