**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***](https://sit.instructure.com/courses/64673/files/10929548/download?wrap=1)is a starter implementation of the triangle classification program.
  + [***TestTriangle.py***](https://sit.instructure.com/courses/64673/files/10929547/download?wrap=1)**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*](https://sit.instructure.com/courses/64673/files/10929548/download?wrap=1)*contains an implementation of the classifyTriangle() function with a few bugs.*  
  
[*TestTriangle.py*](https://sit.instructure.com/courses/64673/files/10929547/download?wrap=1)*contains the initial set of test cases*

Author: Jiayin Huang

Summary: Based on the code and test cases provided, the function classifyTriangle() appears to have several bugs and inaccuracies in its implementation. 1. require that the input values be >= 0 and <= 200, one of the conditions: if a <= 0 or b <= b or c <= 0 isn’t right. It should be: if (a <= 0 or b <= 0 or c <= 0); 2. The condition to check if the triangle is a right triangle is incorrect. 3. The condition to check if the triangle is an isosceles triangle is incorrect. 4. The condition to check if the triangle is a scalene triangle is incorrect. After Fixing these issues so that all test cases pass.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Run1 | Test Run2 | Test Run3 | Test Run4 |
| Test Planned | (-1,2,2),  (0,2,2) | (1,1,1) | (3,4,5),  (5,4,3),  (5,12,13) | (5,5,7)  (8,8,10)  (8,15,20)  (12,17,26)  (12,3,20)  (6,7,18) |
| Test Executed | 2 | 1 | 3 | 4 |
| Test Passed | 2 | 1 | 3 | 4 |
| Defects Found | if a <= 0 or b <= b or c <= 0 | If a == b and b == a | elif ((a \* 2) + (b \* 2)) == (c \* 2) | elif (a != b) and (b != c) and (a != b) |
| Defects Fixed | if (a <= 0 or b <= 0 or c <= 0) | If a == b and b == c and a == c | elif (a \*\* 2 + b \*\* 2 == c \*\* 2) or (a \*\* 2 + c \*\* 2 == b \*\* 2) or (b \*\* 2 + c \*\* 2 == a \*\* 2) | elif a == b or b == c or a == c:  return 'Isosceles'  else:  return 'Scalene' |

Reflection: The indentation of Python code is very important, so pay special attention when writing code. The logic of judging whether it is a triangle must be accurate. After modifying the logic, function now correctly handles all types of triangles, as well as invalid input values.

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.—Jiayin Huang

the original ***Triangle.py result:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| testRightTriangleA | 3,4,5 | Right | InvalidInput | Fail |
| testRightTriangleB | 5,3,4 | Right | InvalidInput | Fail |
| testEquilateralTriangles | 1,1,1 | Equilateral | InvalidInput | Fail |

After debug result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| test\_negative\_triangles | -1,2,2 | InvalidInput | InvalidInput | Pass |
| test\_zero\_length\_triangles | 0,2,2 | InvalidInput | InvalidInput | Pass |
| testRightTriangleC | 5,12,13 | Right | Right | Pass |
| testIsoscelesTriangleA | 5,5,7 | Isosceles | Isosceles | Pass |
| testIsoscelesTriangleB | 8,8,10 | Isosceles | Isosceles | Pass |
| testScaleneTriangleA | 8,15,20 | Scalene | Scalene | Pass |
| testScaleneTriangleB | 12,17,26 | Scalene | Scalene | Pass |
| testNotATriangleA | 12,3,20 | NotATriangle | NotATriangle | Pass |
| testNotATriangleB | 6,7,18 | NotATriangle | NotATriangle | Pass |