HW02a - Part 2 –

1. **Assignment Description**: update the logic in classifytTriangle() to fix all of the logic bugs you found by code inspection and with your test cases.
2. **Author**: dagrawa2 (DeeptiAgrawal)
3. **Summary**:
   Result after fixing defects Triangle.py

Test ID	Input	Expected Result	Actual Result	Pass Or Fail
testRightTriangleA	3,4,5	Right	Right	Pass
testRightTriangleB	5,3,4	Right	Right	Pass
testRightTriangleC	4,3,5	Right	Right	Pass
testEquilateralTriangles	1,1,1	Equilateral	Equilateral	Pass
testNotEquilateralTriangles	3,1,2	! Equilateral	! Equilateral	Pass
testInvalidInput	1.0,2.0,3.0	InvalidInput	InvalidInput	Pass
testInvalidInput1	0, 1, 0	InvalidInput	InvalidInput	Pass
testScaleneTriangle	2, 3, 4	Scalene	Scalene	Pass
testNotAScaleneTriangle	2, 1, 2	! Scalene	! Scalene	Pass
testIsocelesTriangleA	2, 2, 1	Isoceles	Isoceles	Pass
testIsocelesTriangleB	2, 1, 2	Isoceles	Isoceles	Pass
testIsocelesTriangleC	1, 2, 2	Isoceles	Isoceles	Pass
testNotAIsocelesTriangle	1, 2, 3	!Isoceles	! Isoceles	Pass
testNotATriangle	1, 10, 12	NotATriangle	NotATriangle	Pass

4. **Honor pledge**
   I pledge, I am adhering to Stevens code of conduct.
5. **Detailed results, if any:**
   For negative result, I am not able to see what was actual result, that's why I have mentioned **!**Equilateral, !Scalene and ! Isoceles in actual response.

	Test Run 1	Test Run 2
Tests Planned	14	14
Tests Executed	14	14
Tests Passed	5	14
Defects Found	5	0
Defects Fixed	5	0

Defects found.
Defect 1 – line number 34 - if a <= 0 or b <= b or c <= 0:
Defect 2 – line number 46 - if (a >= (b - c)) or (b >= (a - c)) or (c >= (a + b)):

Defect 3 – line number 52 - elif ((a * 2) + (b * 2)) == (c * 2):
Defect 4 – line number 54 - elif (a != b) and (b != c) and (a != b):
Defect 5 - line number 50 - if a == b and b == a:

Description of the strategy I have used to decide when you had a sufficient number of test cases -
In order to confirm I have sufficient number of test cases, I have used one negative and one positive test case of each type of triangle. Also wrote test cases to test invalid input.

I have used Input based on type of triangle based on below assumptions –

> 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'

And for invalid input I have tested float value (to test input is of type integer) and side with 0.