```python
class TestHW01(ut.TestCase):
    #Test cases class with extended test cases.
    def test_classify_triangle_case_1(self):
        self.assertEqual(classify_triangle(1,3,2.5), "scalene")
        self.assertEqual(classify_triangle(3,1,2.5), "scalene")
        self.assertEqual(classify_triangle(3,4,5), "right")
        self.assertEqual(classify_triangle(4,3,5), "right")
        self.assertEqual(classify_triangle(4,4,5), "isosceles")
        self.assertEqual(classify_triangle(4,5,4), "isosceles")
        self.assertEqual(classify_triangle(4,4,4), "equilateral")

    def test_classify_triangle_case_2(self):
        self.assertEqual(classify_triangle(-4,-4,-4), "Wrong Input, Not a Triangle")
        self.assertEqual(classify_triangle(-3,-5,-4), "Wrong Input, Not a Triangle")
        self.assertEqual(classify_triangle(-3,-4,-4), "Wrong Input, Not a Triangle")
        self.assertEqual(classify_triangle(-2,-5,-4), "Wrong Input, Not a Triangle")
        self.assertEqual(classify_triangle(0,0,0), "Wrong Input, Not a Triangle")

```

Fig. 1 The Test Cases

```python
def classify_triangle(a, b, c):
    #the a,b,c represents the three sides of a triangle.
    sorted_sides = [a,b,c]
    sorted_sides.sort()
    for side in sorted_sides:
        if side < 0:
            return "Wrong Input, Not a Triangle"
    a, b, c = sorted_sides[0], sorted_sides[1], sorted_sides[2]
    if a*a + b*b == c*c:
        return "right"
    if a == b or b == c:
        if a == c:
            return "equilateral"
        else:
            return "isosceles"
    else:
        return "scalene"

```

Fig. 2 The Buggy Triangle Function Code

```
test_classify_triangle_case_1 (__main__.TestHW01) ... ok
test_classify_triangle_case_2 (__main__.TestHW01) ... FAIL

======================================================================
FAIL: test_classify_triangle_case_2 (__main__.TestHW01)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/qizhao/Documents/Spring 2020/567/HW01.py", line 23, in test_classify_triangle_case_2
    self.assertEqual(classify_triangle(0,0,0), "Wrong Input, Not a Triangle")
AssertionError: 'right' != 'Wrong Input, Not a Triangle'
- right
+ Wrong Input, Not a Triangle


----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)
```

Fig. 3 The test report

Deliverable 2: Upload a text file or screen shot to show the input and output of running the program and demonstrating that your program has been adequately tested.

As the Fig.1 and Fig.2 shown above, I choose several test cases for the triangle classify function, which including the requirement of different type of triangles, the order of the set of the sides, and the basic condition of being a triangle. Therefore, I believe my test cases are fully covered most of the cases.

And as the Fig. 3 shown, which is the test report of the example 'buggy' function, which intends to point out the defect of the classify_triangle function.

Deliverable 3: Describe your experience with this assignment, specifically:

- **WHAT CHALLENGES DID YOU ENCOUNTER WITH THIS ASSIGNMENT, IF ANY?**
- **WHAT DID YOU THINK ABOUT THE REQUIREMENTS SPECIFICATION FOR THIS ASSIGNMENT?**
- **WHAT CHALLENGES DID YOU ENCOUNTER WITH THE TOOLS?**
- **DESCRIBE THE CRITERIA YOU USED TO DETERMINE THAT YOU HAD SUFFICIENT TEST CASES, I.E. HOW DID YOU KNOW YOU WERE DONE?**

First, the main challenge I faced was that how to list the test cases based on the mathematic condition of a triangle. And what strategy should I use in order to fully covered the possibilities.
Second, I think the requirements are little missing of information. For example, what is the requirement of input variable and the output. And the requirement doesn't infer the case where none of the triangle the sides can make.
Third, there is few challenges I have regard of the Unit test tool.
Finally, I used to 'feel' the competence of my test cases or, in other words, by watching the code. I usually check the sufficiency with my previous experience.

Deliverable 4: Submit the URL of the GitHub repo containing your complete solution. The files in the repo should match what you have uploaded to Canvas.

https://github.com/qiblaqi/567-Spring-2020-qzhao/blob/master/567/HW01.py