1. **Assignment Description:** Develop a set of tests for an existing triangle classification program. Use those tests to find and fix defects in that program, Report on your testing results for the Triangle problem

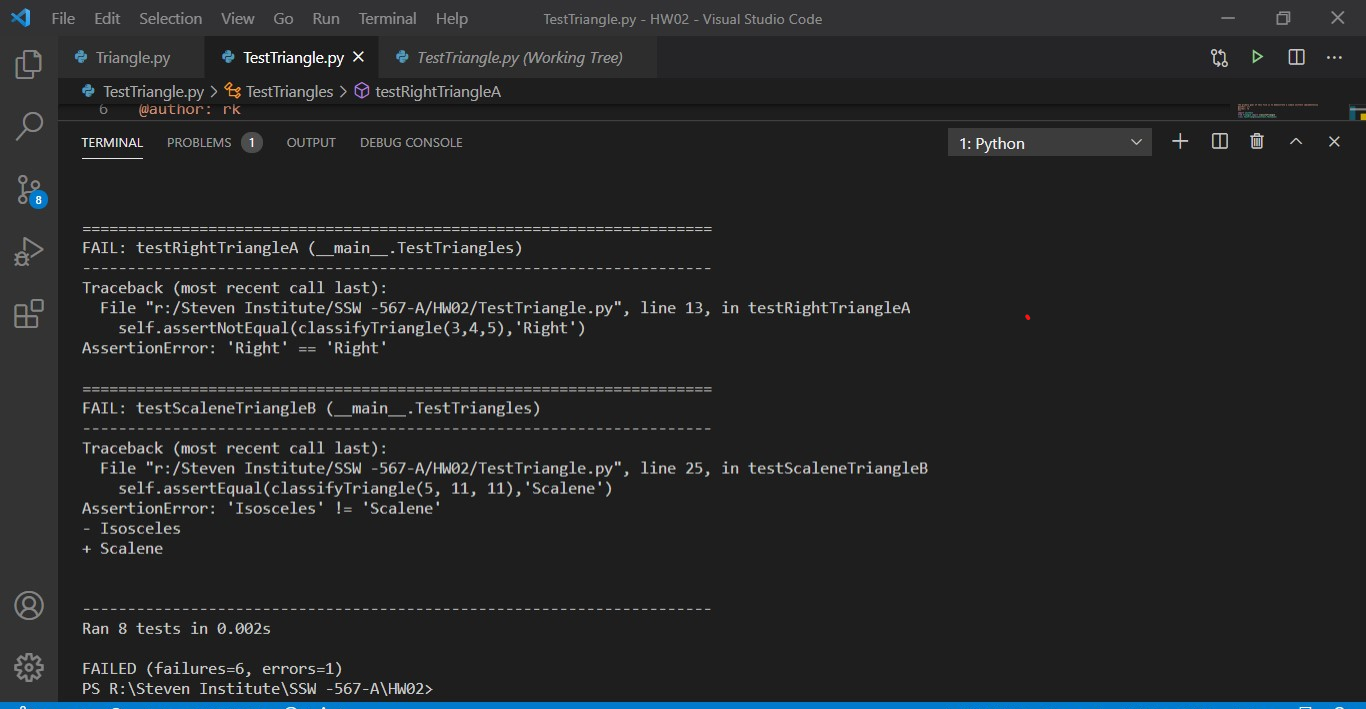
These are the two files:  Triangle.py and TestTriangle.py

* 1. [***Triangle.py***](https://sit.instructure.com/courses/34748/files/5082668/download?wrap=1)is a starter implementation of the triangle classification program.
  2. [***TestTriangle.py***](https://sit.instructure.com/courses/34748/files/5082660/download?wrap=1)**c**ontains a starter set of unittest test cases to test the classifyTriangle() function in the file Triangle.py file.

1. **Author:** Amisha Patel

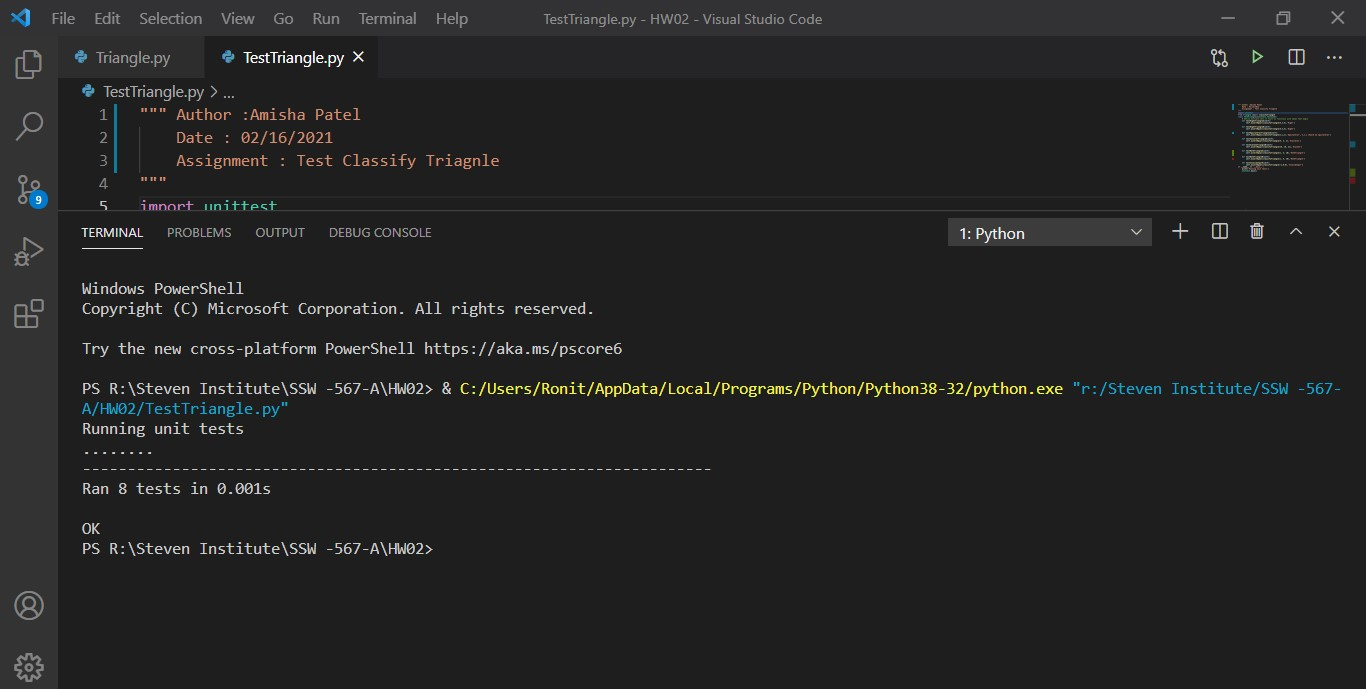
**Summary:** The program identifies a triangle type (right, equilateral, scalene, isosceles) based on given inputs of the lengths of the sides. The test cases included different possible inputs including 0, negative values and positive values that cannot form a triangle.

**Part 1:**

Total 8 test cases were run from the TestTriangle.py file to test Triangle.py code. The six test cases initially failed and two passed.

As per below table gives the details of the test case before defect fixing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| testRightTriangleA | 3,4,5 | Right | Right | Pass |
| testRightTriangleB | 5,4,3 | Right | Right | Pass |
| testEquilateralTrianglesA | 1,0,1 | Equilateral | InvalidInput | Fail |
| testIsocelesTriangleA | 4,14,2 | Isosceles | InvalidInput | Fail |
| testScaleneTriangleB | 5,11,11 | Scalene | InvalidInput | Fail |
| testNotATriangleA | 1,6,0 | NotATriangle | InvalidInput | Fail |
| testNotATriangleB | -5,9,1 | NotATriangle | InvalidInput | Fail |
| testInvalidInputA | a,0,0 | InvalidInput | NameError | Fail |

As per below table below gives the details of the test case defect fixed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Result | Pass or Fail |
| testRightTriangleA | 3,4,5 | Right | Right | Pass |
| testRightTriangleB | 5,4,3 | Right | Right | Pass |
| testEquilateralTrianglesA | 1,1 ,1 | Equilateral | Equilateral | Pass |
| testIsocelesTriangleA | 4,4,2 | Isosceles | Isosceles | Pass |
| testScaleneTriangleB | 5,10,11 | Scalene | Scalene | Pass |
| testNotATriangleA | 1,6,10 | NotATriangle | NotATriangle | Pass |
| testNotATriangleB | 1,6,1 | NotATriangle | NotATriangle | Pass |
| testInvalidInputA | -1,0,0 | InvalidInput | InvalidInput | Pass |

**Test Report on testing the bug free code:**

**Test Report Summary**

|  |  |  |
| --- | --- | --- |
|  | Test Run 1 | Test Run 2 |
| Tests Planned | 8 | 8 |
| Tests Executed | 8 | 8 |
| Tests passed | 2 | 8 |
| Defects found | 6 | 0 |
| Defects fixed | 0 | 6 |

Reflection:

I learned the domain application of the triangle.py was accurate; it was a few typo’s, incomplete coverage, and order of execution errors that caused the initial tests to fail. The only test that worked was the “testInvalidInputA” with the sides greater than 200. Otherwise, most of the results were ‘InvalidInput’ where they were not suppose to be.

Honor Pledge:

Yes

Detailed Results:

I assumed there will be 3 inputs always into classifyTriangle function that would be of any type. I used the unittest module to automatically execute and present each test result. Source code can be found out <https://github.com/Erik-Bornako/HW02_SSW567_Triangle>