**HW 02a - Testing a legacy program and reporting on testing results**

* **Assignment Description:**

The objective of this assignment is for you to (a) develop a set of tests for an existing triangle classification program, (b) use those tests to find and fix defects in that program, and (c) report on your testing results for the Triangle problem.

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/74569/files/13942747/download?wrap=1)is a starter implementation of the triangle classification program.
  + [***TestTriangle.py***](https://sit.instructure.com/courses/74569/files/13942739/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.

* **Author**

Soham Mehul Joshi

* **Summary**

1. All tests were designed and executed successfully.
2. The initial few failures due to flawed reasoning in the classifyTriangle function.
3. After fixing the problems, all tests ran successfully.

Outcome:

1. This project gave me hands-on experience in finding and correcting bugs in an already existing code.
2. Learnt the triangle inequality theorem, applied in input validation.
3. Extensive test cases very helpful.
4. What did not work: First logic treatment did not handle the right and isosceles triangles correctly.

* **Honor Pledge:**

I hereby declare that I did this assignment by myself and did not indulge in any form of academic dishonesty.

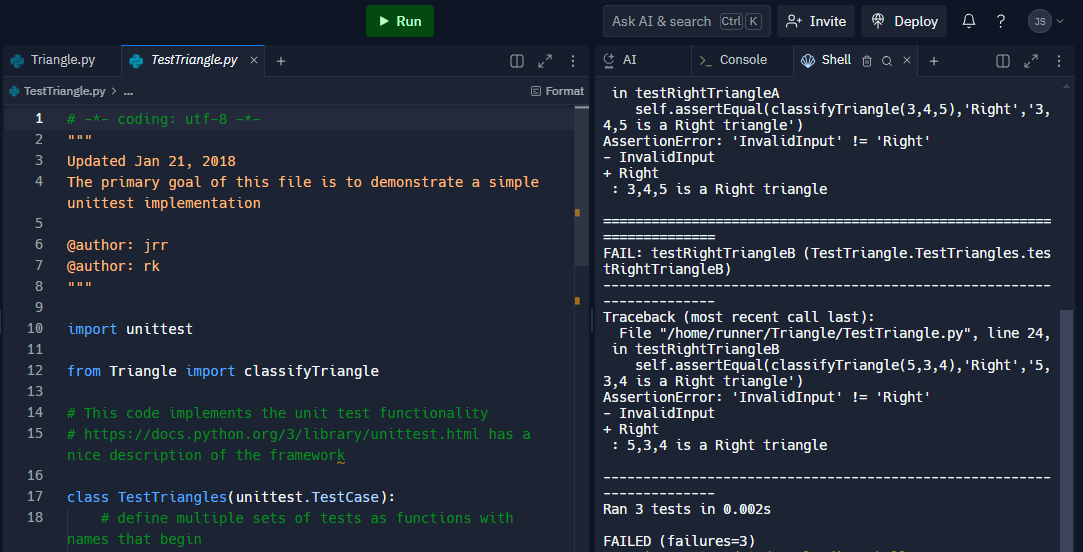
* **Detailed Results**:

Three test reports are summarised below to show how the test cases failed to passed once the logic in classifyTriangle was corrected:

1. Test Report 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Result | Actual Result | Pass/Fail |
| testEquilateralTriangles | (1,1,1) | Equilateral | Invalid Input | Fail |
| testRightTriangleA | (3,4,5) | Right | Invalid Input | Fail |
| testRightTriangleB | (5,3,4) | Right | Invalid Input | Fail |

Total Tests Run: 3, Tests Passed: 0, Tests Failed: 3

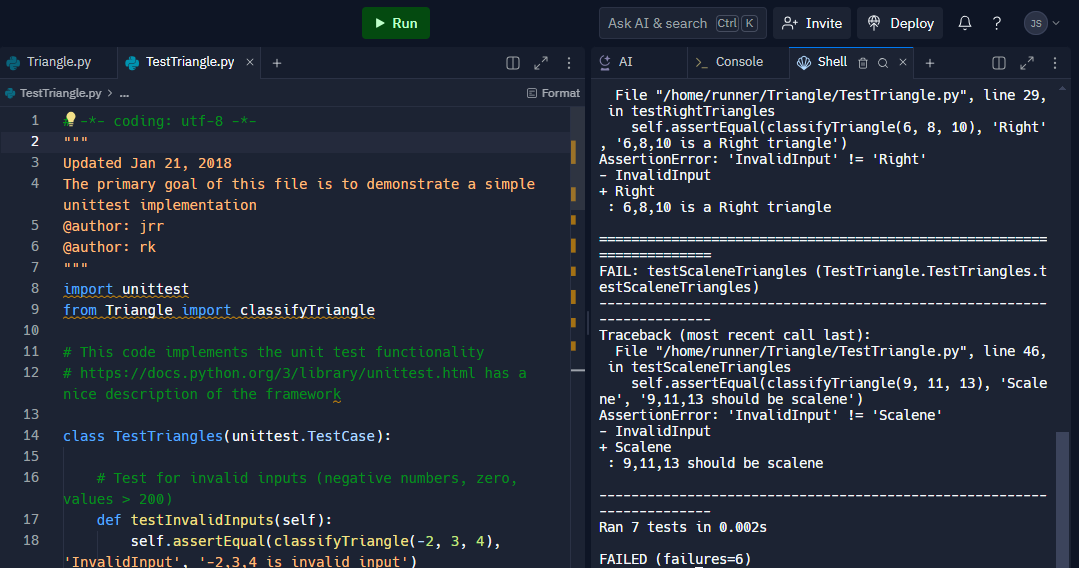


Result with no Changes

1. Test Report2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Result | Actual Result | Pass/Fail |
| testEdgeCases | (1, 2, 3) | Not a Triangle | Invalid Input | Fail |
| testEquilateralTriangles | (7,7,7) | Equilateral | Invalid Input | Fail |
| testIsoscelesTriangles | (10,10,15) | Isosceles | Invalid Input | Fail |
| testNotATriangle | (2,10,20) | Not a Triangle | Invalid Input | Fail |
| testRightTriangles | (6,8,10) | Right | Invalid Input | Fail |
| testScaleneTriangles | (9,11,13) | Scalene | Invalid Input | Fail |
| testInvalidInputs | (-2,4,3) | Invalid Input | Invalid Input | Pass |

Total Tests Run: 7, Tests Passed: 0, Tests Failed: 6

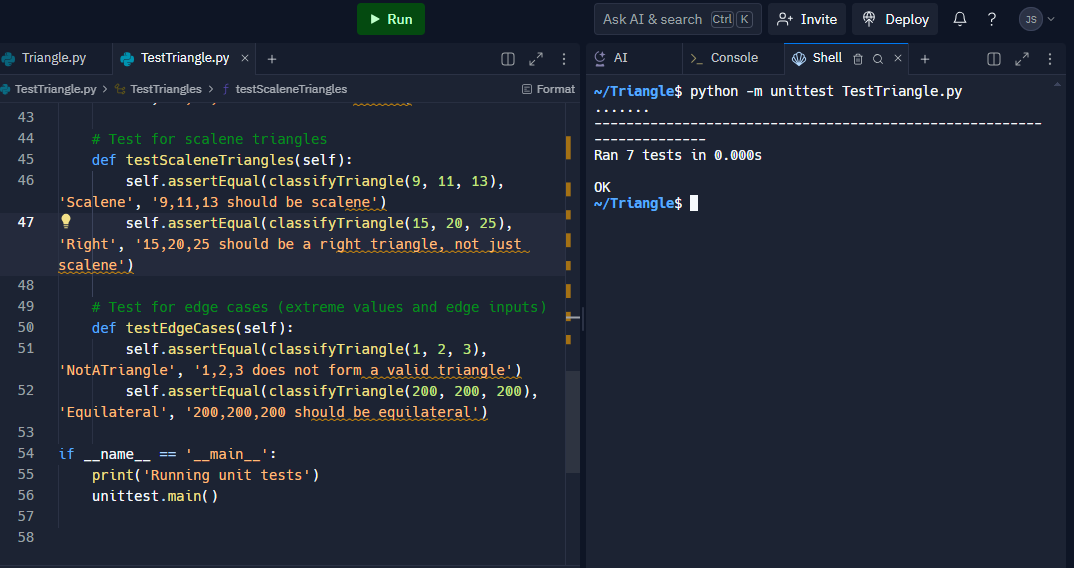


Changes in Test Code

1. Test Report3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Result | Actual Result | Pass/Fail |
| testEdgeCases | (1, 2, 3) | Not a Triangle | Not a Triangle | Pass |
| testEquilateralTriangles | (7,7,7) | Equilateral | Equilateral | Pass |
| testIsoscelesTriangles | (10,10,15) | Isosceles | Isosceles | Pass |
| testNotATriangle | (2,10,20) | Not a Triangle | Not a Triangle | Pass |
| testRightTriangles | (6,8,10) | Right | Right | Pass |
| testScaleneTriangles | (9,11,13) | Scalene | Scalene | Pass |
| testInvalidInputs | (-2,4,3) | Invalid Input | Invalid Input | Pass |

Total Tests Run: 7, Tests Passed: 7, Tests Failed: 0



Result with main Code Change

* **Final Test Results Table**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 |
| Tests Planned | 3 | 7 | 7 |
| Tests Executed | 3 | 7 | 7 |
| Tests Passed | 0 | 1 | 7 |
| Defects Found | 3 | 6 | 0 |
| Defects Fixed | 0 | 0 | 7 |