SSW567 – HW02a

1. **Description**

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 is a starter implementation of the triangle classification program.

TestTriangle.py contains 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 contains an implementation of the classifyTriangle() function with a few bugs.

TestTriangle.py contains the initial set of test cases

1. **Author: Leena Domadia**

GitHub Repository: <https://github.com/LDomadia/SSW567-TriangleHW02a>

1. **Summary**

**Part 1:** Running initial test cases

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

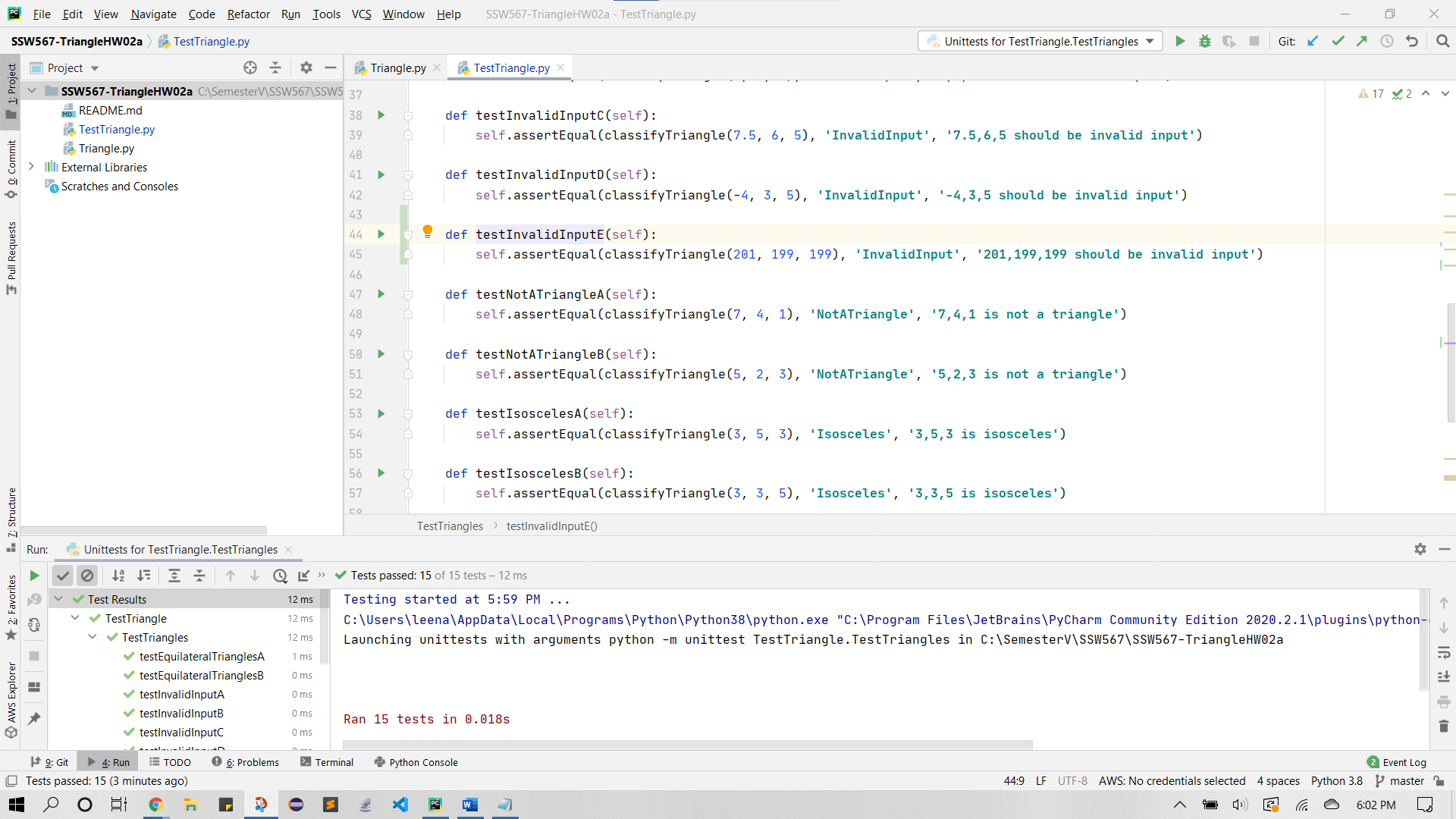
**Part 2:** Updating Triangle.py, adding more test cases to TestTriangle.py, and running all test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Results | Pass or Fail |
| **testRightTriangleA** | **3, 4, 5** | **NotATriangle** | **Right** | **Fail** |
| **testRightTriangleB** | **5, 3, 4** | **NotATriangle** | **Right** | **Fail** |
| **testEquilateralTriangles** | **1, 1, 1** | **NotATriangle** | **Equilateral** | **Fail** |
| **testInvalidInputA** | **201, 201, 201** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testInvalidInputB** | **3, 0, 5** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testInvalidInputC** | **7.5, 6, 5** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testInvalidInputD** | **-4, 3, 5** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testNotATriangle** | **7, 4, 1** | **NotATriangle** | **NotATriangle** | **Pass** |

**Part 3**: Updating Triangle.py, adding more test cases to TestTriangle.py, and running all test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Input | Expected Results | Actual Results | Pass or Fail |
| **testRightTriangleA** | **3, 4, 5** | **Right** | **Right** | **Pass** |
| **testRightTriangleB** | **5, 3, 4** | **Right** | **Right** | **Pass** |
| **testEquilateralTrianglesA** | **1, 1, 1** | **Equilateral** | **Equilateral** | **Pass** |
| **testEquilateralTrianglesB** | **200, 200, 200** | **Equilateral** | **Equilateral** | **Pass** |
| **testInvalidInputA** | **201, 201, 201** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testInvalidInputB** | **3, 0, 5** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testInvalidInputC** | **7.5, 6, 5** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testInvalidInputD** | **-4, 3, 5** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testInvalidInputE** | **201, 199, 199** | **InvalidInput** | **InvalidInput** | **Pass** |
| **testNotATriangleA** | **7, 4, 1** | **NotATriangle** | **NotATriangle** | **Pass** |
| **testNotATriangleB** | **5, 2, 3** | **NotATriangle** | **NotATriangle** | **Pass** |
| **testIsoscelesA** | **3, 5, 3** | **Isosceles** | **Isosceles** | **Pass** |
| **testIsoscelesB** | **3, 3, 5** | **Isosceles** | **Isosceles** | **Pass** |
| **testIsoscelesC** | **5, 3, 3** | **Isosceles** | **Isosceles** | **Pass** |
| **testScaleneA** | **6, 3, 5** | **Scalene** | **Scalene** | **Pass** |

**Output of the Test Cases:**

****

Testing started at 5:59 PM ...

C:\Users\leena\AppData\Local\Programs\Python\Python38\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2020.2.1\plugins\python-ce\helpers\pycharm\\_jb\_unittest\_runner.py" --target TestTriangle.TestTriangles

Launching unittests with arguments python -m unittest TestTriangle.TestTriangles in C:\SemesterV\SSW567\SSW567-TriangleHW02a

Ran 15 tests in 0.018s

OK

Process finished with exit code 0

**Summary:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test Run 1 | Test Run 2 | Test Run 3 |
| **Tests Planned** | **3** | **8** | **15** |
| **Tests Executed** | **3** | **8** | **15** |
| **Tests Passed** | **0** | **5** | **15** |
| **Defects Found** | **5** | **1** | **0** |
| **Defects Fixed** | **5** | **1** | **0** |

**Reflection:**

In this assignment, I was able to use what I learned in class about testing programs and apply it to TestingTriangle.py. I was able to update Triangle.py as test cases in TestTriangle.py failed. I realized that after Test Run 3, which had 13 successful test cases, my code didn’t account for all possible case scenarios. Therefore, I added two more test cases for a total of 15 test cases. Now, I believe my code has been adequately tested.

1. **Honor Pledge**

*“I pledge my Honor that I have abided by the Stevens Honor System.” – Leena Domadia*

1. **Detailed Results**

Regarding my data inputs, I made sure to include various different values. This includes zeros, negatives, positives, floats, and values greater than 200.

In the end, all 15 of my unique test cases successfully passed. All copies of the code will be uploaded along with this document and available on my GitHub Repo: <https://github.com/LDomadia/SSW567-TriangleHW02a>