Aidan Nestor
Professor Morgan
SSW 567
4 October 2024

Assignment 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*** **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.

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

TestTriangle.py contains the initial set of test cases

Author: Aidan Nestor

Summary

I came up with nine test cases in addition to the three that were provided for a total of 12 test cases. Initially, only two of these tests passed when ran against the Triangle.py file. I then made significant changes to the Traingle.py file while running the same cases to see what would pass and what wouldn't. I eventually got the program to pass all of the test cases. I thought this assignment was pretty fun. I enjoyed the debugging process after all the test cases were created because I knew exactly what the problem was instead of having little information on the error. The test cases made the debugging process much faster. The process of creating the test cases was informative because I had to think of many different test cases and scenarios to get use from the cases. What worked for me was reading the comments in the Triangle.py file so I knew any constraints before doing my test cases so that I was informed while making my cases. I decided I had enough test cases when I was testing every block of code in a way relevant to the constraints provided. I would try to make as many test cases as I could for each section to be thorough. My test runs are shown below from when I was debugging.

	Test Run 1	Test Run 2	Test Run 3	Test Run 4	Test Run 5	Test Run 6
Tests Planned	12	12	12	12	12	12
Tests Executed	12	12	12	12	12	12
Tests Passed	2	5	6	9	10	12
Defects Found	1	1	3	1	1	0
Defects Fixed	1	1	3	1	1	0

Pledge: I pledge my Honor that I have abided by the Stevens Honors System.

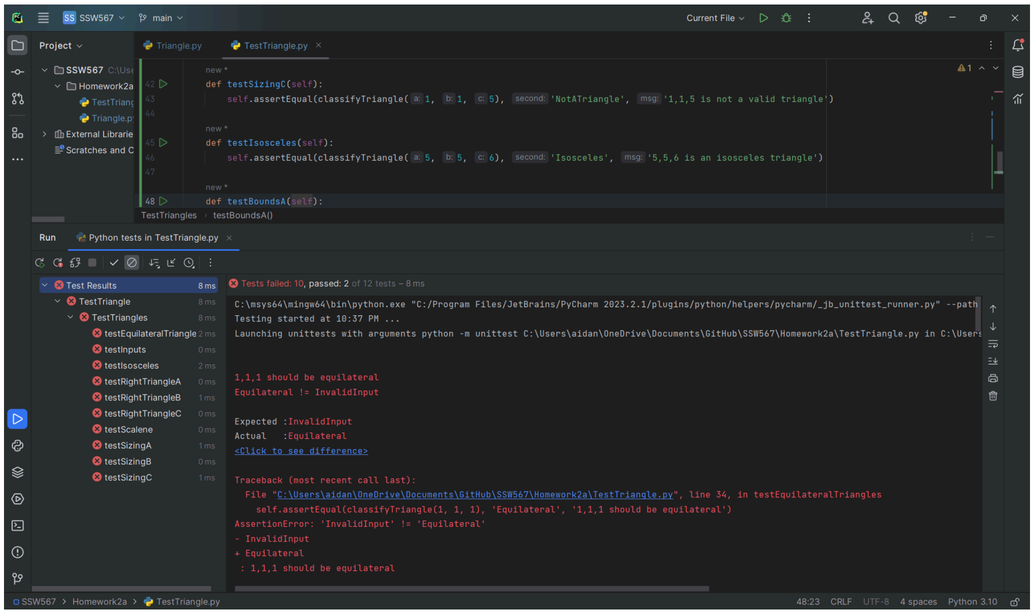
Detailed Results

I did not use any specific technique for this assignment. It was quite straightforward. The rest of my results are provided below for this assignment.

This table shows the details of my test cases before the correction of Triangle.py.

Test ID	Input	Expected Results	Actual Results	Pass/Fail
testRightTriangleA	3,4,5	Right	InvalidInput	Fail
testRightTriangleB	5,3,4	Right	InvalidInput	Fail
testRightTriangleC	3,5,4	Right	InvalidInput	Fail
testInputs	“5”,[3,2],4	InvalidInput	Error	Fail (Error)
testEquilateralTriangles	1,1,1	Equilateral	InvalidInput	Fail
testSizingA	5,1,1	NotATriangle	InvalidInput	Fail
testSizingB	1,5,1	NotATriangle	InvalidInput	Fail
testSizingC	1,1,5	NotATriangle	InvalidInput	Fail
testIsosceles	5,5,6	Isosceles	InvalidInput	Fail
testBoundsA	200,250,250	InvalidInput	InvalidInput	Pass
testBoundsB	-200,-250,-250	InvalidInput	InvalidInput	Pass
testScalene	18,28,38	Scalene	InvalidInput	Fail

Output of test cases before correction of Triangle.py.


SSW567
main
Current File
Project
Triangle.py
TestTriangle.py
TestTriangles › testBoundsA()
Run
Python tests in TestTriangle.py
48:23 CRLF UTF-8 4 spaces Python 3.10
SSW567 › Homework2a › TestTriangle.py

```
    new *
    def testSizingC(self):
        self.assertEqual(classifyTriangle( a: 1,  b: 1,  c: 5),  second: 'NotATriangle',  msg: '1,1,5 is not a valid triangle')

    new *
    def testIsosceles(self):
        self.assertEqual(classifyTriangle( a: 5,  b: 5,  c: 6),  second: 'Isosceles',  msg: '5,5,6 is an isosceles triangle')

    new *
    def testBoundsA(self):
```

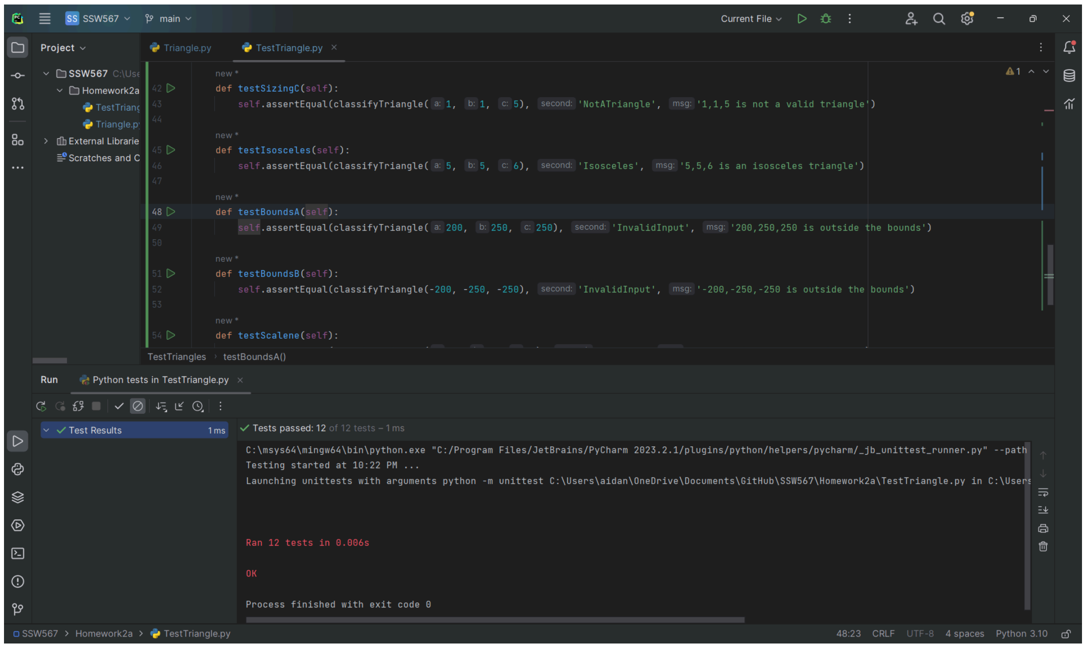
```
Test Results                      8 ms
  TestTriangle                    8 ms
    TestTriangles                 8 ms
      testEquilateralTriangle     2 ms
      testInputs                  0 ms
      testIsosceles               2 ms
      testRightTriangleA          0 ms
      testRightTriangleB          1 ms
      testRightTriangleC          0 ms
      testScalene                 0 ms
      testSizingA                 1 ms
      testSizingB                 0 ms
      testSizingC                 1 ms
```

```
Tests failed: 10, passed: 2 of 12 tests – 8 ms
C:\msys64\mingw64\bin\python.exe "C:/Program Files/JetBrains/PyCharm 2023.2.1/plugins/python/helpers/pycharm/_jb_unittest_runner.py" --path
Testing started at 10:37 PM ...
Launching unittests with arguments python -m unittest C:\Users\aidan\OneDrive\Documents\GitHub\SSW567\Homework2a\TestTriangle.py in C:\Users


1,1,1 should be equilateral
Equilateral != InvalidInput

Expected :InvalidInput
Actual   :Equilateral
<Click to see difference>

Traceback (most recent call last):
  File "C:\Users\aidan\OneDrive\Documents\GitHub\SSW567\Homework2a\TestTriangle.py", line 34, in testEquilateralTriangles
    self.assertEqual(classifyTriangle(1, 1, 1), 'Equilateral', '1,1,1 should be equilateral')
AssertionError: 'InvalidInput' != 'Equilateral'
- InvalidInput
+ Equilateral
 : 1,1,1 should be equilateral
```

Test case details after I corrected Triangle.py correction.

Test ID	Input	Expected Results	Actual Results	Pass/Fail
testRightTriangleA	3,4,5	Right	Right	Pass
testRightTriangleB	5,3,4	Right	Right	Pass
testRightTriangleC	3,5,4	Right	Right	Pass
testInputs	“5”,[3,2],4	InvalidInput	InvalidInput	Pass
testEquilateralTriangles	1,1,1	Equilateral	Equilateral	Pass
testSizingA	5,1,1	NotATriangle	NotATriangle	Pass
testSizingB	1,5,1	NotATriangle	NotATriangle	Pass
testSizingC	1,1,5	NotATriangle	NotATriangle	Pass
testIsosceles	5,5,6	Isosceles	Isosceles	Pass
testBoundsA	200,250,250	InvalidInput	InvalidInput	Pass
testBoundsB	-200,-250,-250	InvalidInput	InvalidInput	Pass
testScalene	18,28,38	Scalene	Scalene	Pass

Output of test cases after correction.



```
    new *
    def testSizingC(self):
        self.assertEqual(classifyTriangle( a: 1,  b: 1,  c: 5),  second: 'NotATriangle',  msg: '1,1,5 is not a valid triangle')

    new *
    def testIsosceles(self):
        self.assertEqual(classifyTriangle( a: 5,  b: 5,  c: 6),  second: 'Isosceles',  msg: '5,5,6 is an isosceles triangle')

    new *
    def testBoundsA(self):
        self.assertEqual(classifyTriangle( a: 200,  b: 250,  c: 250),  second: 'InvalidInput',  msg: '200,250,250 is outside the bounds')

    new *
    def testBoundsB(self):
        self.assertEqual(classifyTriangle(-200, -250, -250),  second: 'InvalidInput',  msg: '-200,-250,-250 is outside the bounds')

    new *
    def testScalene(self):
```

Tests passed: 12 of 12 tests – 1 ms

```
C:\msys64\mingw64\bin\python.exe "C:/Program Files/JetBrains/PyCharm 2023.2.1/plugins/python/helpers/pycharm/_jb_unittest_runner.py" --path
Testing started at 10:22 PM ...
Launching unittests with arguments python -m unittest C:\Users\aidan\OneDrive\Documents\GitHub\SSW567\Homework2a\TestTriangle.py in C:\Users



Ran 12 tests in 0.006s

OK

Process finished with exit code 0
```

The results of my work show that I was able to successfully create test cases and debug using them. There can always be more test cases so it may not be perfect but I believe I fixed the Triangle.py file completely. The inputs for each test case were picked specifically for each case I wrote so that I could gather the most information on the classifyTriangle() function. For example, for the testInputs test, I included both a String and a List to see what result the test would yield. When I noticed that this gave back an error instead of a failed test, I realized that the inputs weren't reaching the part of the code where the input type was checked, and was able to debug the program as a result.