Author

Victor Giraldo

Github Link

Github

Honor Pledge

I pledge my honor that I have abided by the Stevens Honor System. - Victor Giraldo

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 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.

Summary

I learned how to fix a broken product in this assignment. Initially, the code provided passed absolutely none of the tests. Even some of the ones I wrote that were new. I worked backward with the test cases, which was different from the usual workflow I've followed before. Generally speaking, I code something first, then test afterwards. Trying to brute force the logic and bug fixes all at once after adding some more test cases was not efficient at all. It resulted in missing some pretty blatant problems with the code, and extended the time it took to fix the logic for determining what type of triangle we were working with.

Constraints

The only assumption made was that we must have strictly integer inputs. This was already a part of the template code we were given.

Data Inputs/Techniques

The values in the additional test cases were also tested against general rules for triangles and shapes before attempting to classify what type of triangle we had. This included negative numbers, string variable types, zeros, etc.

My technique for debugging was to work through the logic for one portion of the code at a time, and try to think of relevant test cases for each of them. I would run these test cases, compare the actual results and expected results, and continue to refactor if it was not what I expected. It is also quite easy to arrive at the problem by simply analyzing the error messages from the unit tests.

	Test Run 1	**Test Run 2**	**Test Run 3**	**Test Run 4**	**Test Run 5**	**Test Run 6**	**Test Run 7**
Tests Planned	7	7	7	7	7	7	7
Tests Executed	7	7	7	7	7	7	7
Tests Passed	0	0	1	2	3	5	7
Defects Found	-	2	1	1	1	-	-
Defects Fixed	-	2	1	1	1	-	-

Table 1: Matrix for Summary Results

Test Runs

This section includes the initial and final test runs.

Initial Test Run

Test Id	**Inputs**	**Expected Results**	**Actual Result**	**Pass or Fail**
1	3,4,5	Right Triangle	Invalid Input	Fail
2	5,3,4	Right Triangle	Invalid Input	Fail
3	1,1,1	Equilateral Triangle	Invalid Input	Fail
4	5,7,9	Scalene Triangle	Invalid Input	Fail
5	6,8,6	Isoceles Triangle	Invalid Input	Fail
6	3,7,2	Not A Triangle	Invalid Input	Fail
7	0,4,5	Invalid Input	Invalid Input	Pass
8	201,70,80	Invalid Input	Invalid Input	Pass

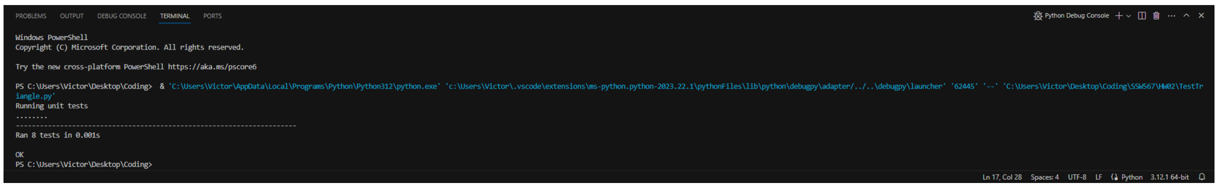
Table 2: A table with the initial test runs.

Final Test Run

Test Id	Inputs	Expected Results	Actual Result	Pass or Fail
1	3,4,5	Right Triangle	Right Triangle	Pass
2	5,3,4	Right Triangle	Right Triangle	Pass
3	1,1,1	Equilateral Triangle	Equilateral Triangle	Pass
4	5,7,9	Scalene Triangle	Scalene Triangle	Pass
5	6,8,6	Isoceles Triangle	Isoceles Triangle	Pass
6	3,7,2	Not A Triangle	Not A Triangle	Pass
7	0,4,5	Invalid Input	Invalid Input	Pass
8	201,70,80	Invalid Input	Invalid Input	Pass

Table 3: A table with the final test runs after the test logic was updated

Results Screenshot



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Victor\Desktop\Coding>  & 'C:\Users\Victor\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Victor\.vscode\extensions\ms-python.python-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '62445' '--' 'C:\Users\Victor\Desktop\Coding\SSW567\HW02\TestTr
iangle.py'
Running unit tests
........
----------------------------------------------------------------------
Ran 8 tests in 0.001s

OK
PS C:\Users\Victor\Desktop\Coding>
```

Figure 1: Screenshot showing output of updated tests.