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

1. **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.

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 test all 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 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 do not have the luxury of rewriting code from scratch and instead must fix what is delivered to the test team.

2.**Author**: Kunal Satija

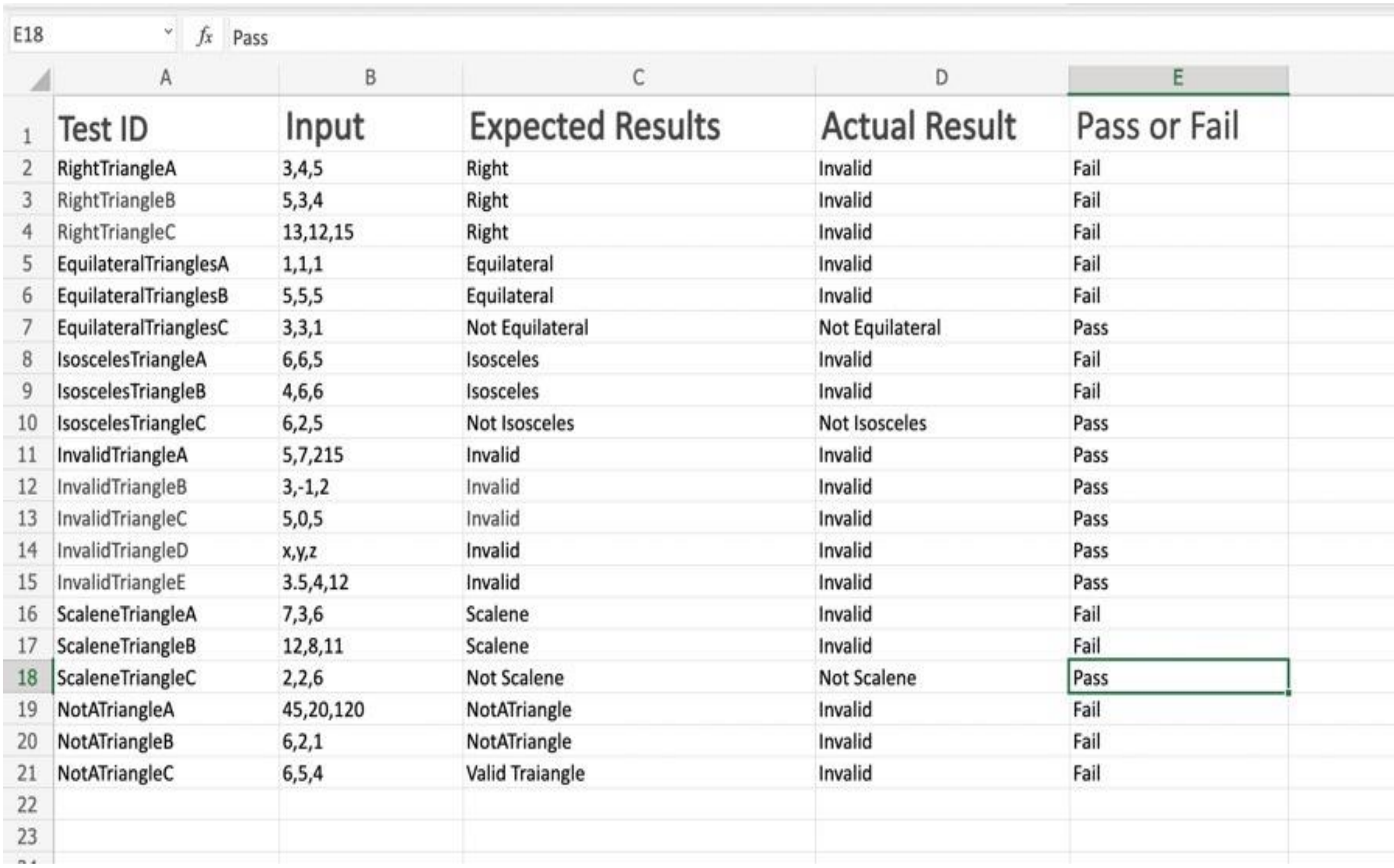
GitHub repository: https://github.com/kunalsatija009/Triangle567_HW-02

3. **Summary**:

Excel sheet link:

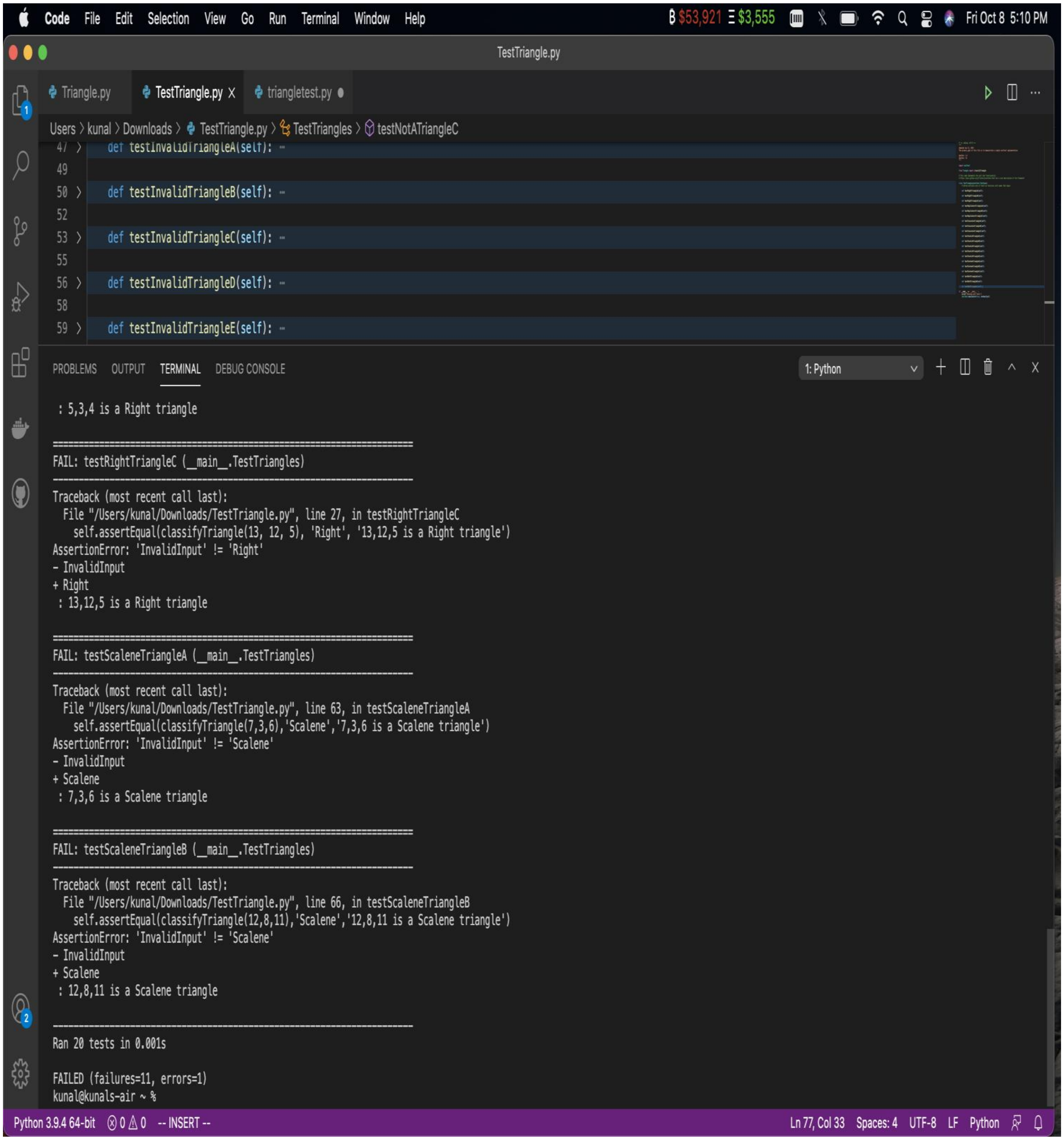
https://stevens0-my.sharepoint.com/:x:/g/personal/ksatija_stevens_edu/EUpsRdDqqkdKuqrGcWpd8JoBcjheweWP1JwHN0YCne7WTQ?e=To4Hcl

3(a) Before fixing the defects:



E18 fx Pass

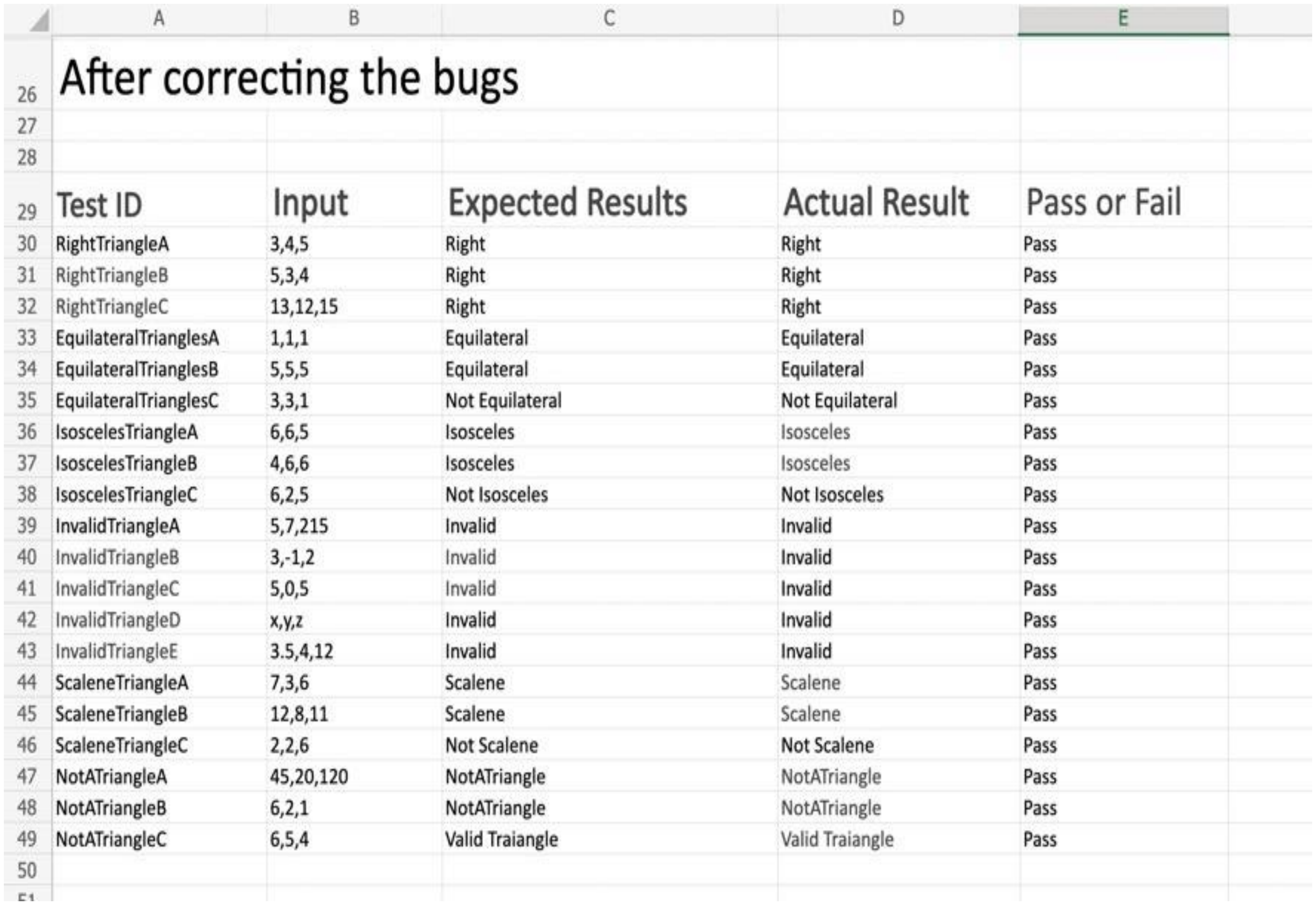
Test ID	Input	Expected Results	Actual Result	Pass or Fail
RightTriangleA	3,4,5	Right	Invalid	Fail
RightTriangleB	5,3,4	Right	Invalid	Fail
RightTriangleC	13,12,15	Right	Invalid	Fail
EquilateralTrianglesA	1,1,1	Equilateral	Invalid	Fail
EquilateralTrianglesB	5,5,5	Equilateral	Invalid	Fail
EquilateralTrianglesC	3,3,1	Not Equilateral	Not Equilateral	Pass
IsoscelesTriangleA	6,6,5	Isosceles	Invalid	Fail
IsoscelesTriangleB	4,6,6	Isosceles	Invalid	Fail
IsoscelesTriangleC	6,2,5	Not Isosceles	Not Isosceles	Pass
InvalidTriangleA	5,7,215	Invalid	Invalid	Pass
InvalidTriangleB	3,-1,2	Invalid	Invalid	Pass
InvalidTriangleC	5,0,5	Invalid	Invalid	Pass
InvalidTriangleD	x,y,z	Invalid	Invalid	Pass
InvalidTriangleE	3.5,4,12	Invalid	Invalid	Pass
ScaleneTriangleA	7,3,6	Scalene	Invalid	Fail
ScaleneTriangleB	12,8,11	Scalene	Invalid	Fail
ScaleneTriangleC	2,2,6	Not Scalene	Not Scalene	Pass
NotATriangleA	45,20,120	NotATriangle	Invalid	Fail
NotATriangleB	6,2,1	NotATriangle	Invalid	Fail
NotATriangleC	6,5,4	Valid Traiangle	Invalid	Fail


Code File Edit Selection View Go Run Terminal Window Help
Fri Oct 8 5:10 PM
TestTriangle.py
Triangle.py
TestTriangle.py
triangletest.py
Users > kunal > Downloads > TestTriangle.py > TestTriangles > testNotATriangleC
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: Python
Python 3.9.4 64-bit ⊗ 0 ⚠ 0 -- INSERT --
Ln 77, Col 33 Spaces: 4 UTF-8 LF Python

```python
    def testInvalidTriangleA(self): ...

    def testInvalidTriangleB(self): ...

    def testInvalidTriangleC(self): ...

    def testInvalidTriangleD(self): ...

    def testInvalidTriangleE(self): ...
```

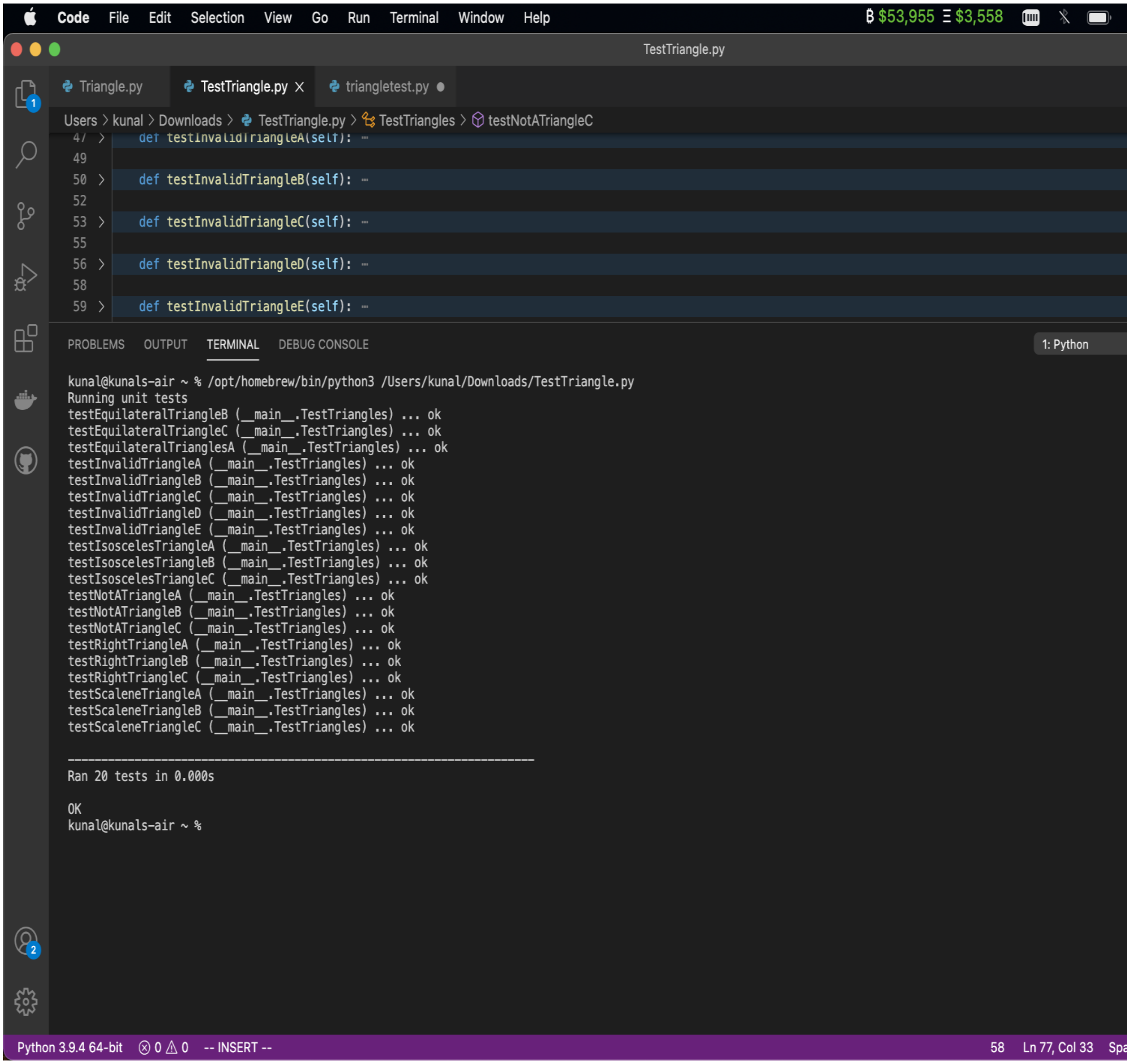
```
 : 5,3,4 is a Right triangle

======================================================================
FAIL: testRightTriangleC (__main__.TestTriangles)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kunal/Downloads/TestTriangle.py", line 27, in testRightTriangleC
    self.assertEqual(classifyTriangle(13, 12, 5), 'Right', '13,12,5 is a Right triangle')
AssertionError: 'InvalidInput' != 'Right'
- InvalidInput
+ Right
 : 13,12,5 is a Right triangle

======================================================================
FAIL: testScaleneTriangleA (__main__.TestTriangles)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kunal/Downloads/TestTriangle.py", line 63, in testScaleneTriangleA
    self.assertEqual(classifyTriangle(7,3,6),'Scalene','7,3,6 is a Scalene triangle')
AssertionError: 'InvalidInput' != 'Scalene'
- InvalidInput
+ Scalene
 : 7,3,6 is a Scalene triangle

======================================================================
FAIL: testScaleneTriangleB (__main__.TestTriangles)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kunal/Downloads/TestTriangle.py", line 66, in testScaleneTriangleB
    self.assertEqual(classifyTriangle(12,8,11),'Scalene','12,8,11 is a Scalene triangle')
AssertionError: 'InvalidInput' != 'Scalene'
- InvalidInput
+ Scalene
 : 12,8,11 is a Scalene triangle

----------------------------------------------------------------------
Ran 20 tests in 0.001s

FAILED (failures=11, errors=1)
kunal@kunals-air ~ %
```

3(b)After fixing the defects:



After correcting the bugs

Test ID	Input	Expected Results	Actual Result	Pass or Fail
RightTriangleA	3,4,5	Right	Right	Pass
RightTriangleB	5,3,4	Right	Right	Pass
RightTriangleC	13,12,15	Right	Right	Pass
EquilateralTrianglesA	1,1,1	Equilateral	Equilateral	Pass
EquilateralTrianglesB	5,5,5	Equilateral	Equilateral	Pass
EquilateralTrianglesC	3,3,1	Not Equilateral	Not Equilateral	Pass
IsoscelesTriangleA	6,6,5	Isosceles	Isosceles	Pass
IsoscelesTriangleB	4,6,6	Isosceles	Isosceles	Pass
IsoscelesTriangleC	6,2,5	Not Isosceles	Not Isosceles	Pass
InvalidTriangleA	5,7,215	Invalid	Invalid	Pass
InvalidTriangleB	3,-1,2	Invalid	Invalid	Pass
InvalidTriangleC	5,0,5	Invalid	Invalid	Pass
InvalidTriangleD	x,y,z	Invalid	Invalid	Pass
InvalidTriangleE	3.5,4,12	Invalid	Invalid	Pass
ScaleneTriangleA	7,3,6	Scalene	Scalene	Pass
ScaleneTriangleB	12,8,11	Scalene	Scalene	Pass
ScaleneTriangleC	2,2,6	Not Scalene	Not Scalene	Pass
NotATriangleA	45,20,120	NotATriangle	NotATriangle	Pass
NotATriangleB	6,2,1	NotATriangle	NotATriangle	Pass
NotATriangleC	6,5,4	Valid Traiangle	Valid Traiangle	Pass

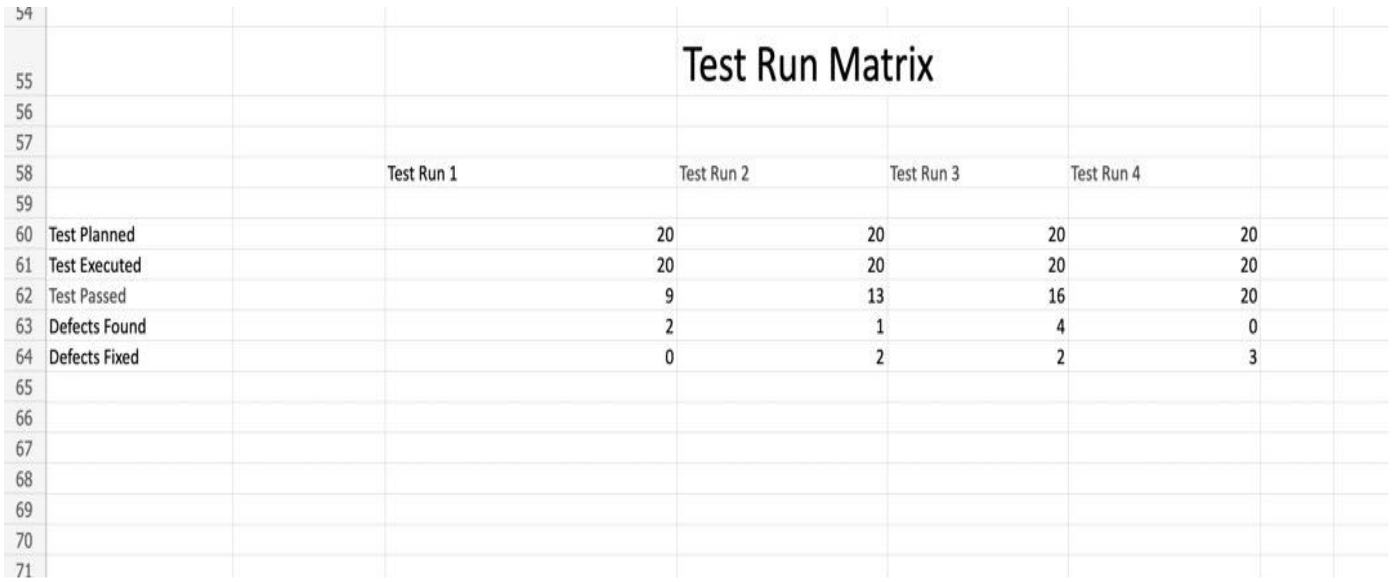


Code File Edit Selection View Go Run Terminal Window Help
TestTriangle.py
Triangle.py
TestTriangle.py
triangletest.py
Users > kunal > Downloads > TestTriangle.py > TestTriangles > testNotATriangleC
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: Python
Python 3.9.4 64-bit
-- INSERT --
Ln 77, Col 33

```
    def testInvalidTriangleA(self): ⋯

    def testInvalidTriangleB(self): ⋯

    def testInvalidTriangleC(self): ⋯

    def testInvalidTriangleD(self): ⋯

    def testInvalidTriangleE(self): ⋯
```

```
kunal@kunals-air ~ % /opt/homebrew/bin/python3 /Users/kunal/Downloads/TestTriangle.py
Running unit tests
testEquilateralTriangleB (__main__.TestTriangles) ... ok
testEquilateralTriangleC (__main__.TestTriangles) ... ok
testEquilateralTrianglesA (__main__.TestTriangles) ... ok
testInvalidTriangleA (__main__.TestTriangles) ... ok
testInvalidTriangleB (__main__.TestTriangles) ... ok
testInvalidTriangleC (__main__.TestTriangles) ... ok
testInvalidTriangleD (__main__.TestTriangles) ... ok
testInvalidTriangleE (__main__.TestTriangles) ... ok
testIsoscelesTriangleA (__main__.TestTriangles) ... ok
testIsoscelesTriangleB (__main__.TestTriangles) ... ok
testIsoscelesTriangleC (__main__.TestTriangles) ... ok
testNotATriangleA (__main__.TestTriangles) ... ok
testNotATriangleB (__main__.TestTriangles) ... ok
testNotATriangleC (__main__.TestTriangles) ... ok
testRightTriangleA (__main__.TestTriangles) ... ok
testRightTriangleB (__main__.TestTriangles) ... ok
testRightTriangleC (__main__.TestTriangles) ... ok
testScaleneTriangleA (__main__.TestTriangles) ... ok
testScaleneTriangleB (__main__.TestTriangles) ... ok
testScaleneTriangleC (__main__.TestTriangles) ... ok

----------------------------------------------------------------------
Ran 20 tests in 0.000s

OK
kunal@kunals-air ~ %
```

3(c) Test Run Matrix



Test Run Matrix

	Test Run 1	Test Run 2	Test Run 3	Test Run 4
Test Planned	20	20	20	20
Test Executed	20	20	20	20
Test Passed	9	13	16	20
Defects Found	2	1	4	0
Defects Fixed	0	2	2	3

3(d)

Test-Driven debugging is a very efficient way to correct lousy code. As I fixed bugs in the code and ran tests, other bugs became apparent. However, writing tests while writing code is a more effective way to check for errors than writing all your code, all problems, in my opinion.

4. **Honor pledge**

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