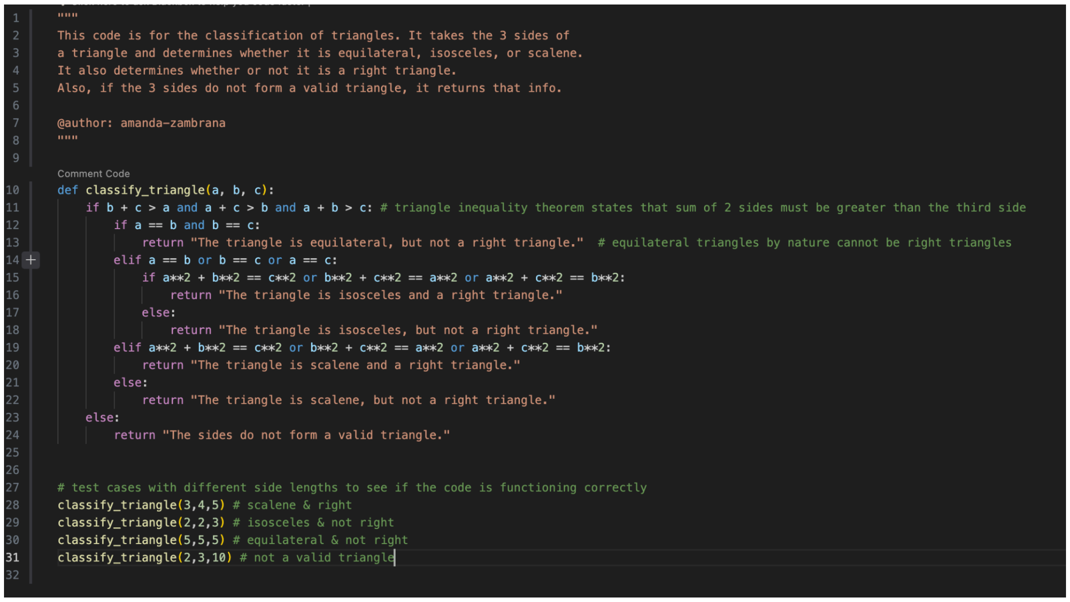
HW 01: Testing Triangle Classification

Amanda Zambrana

SSW-567

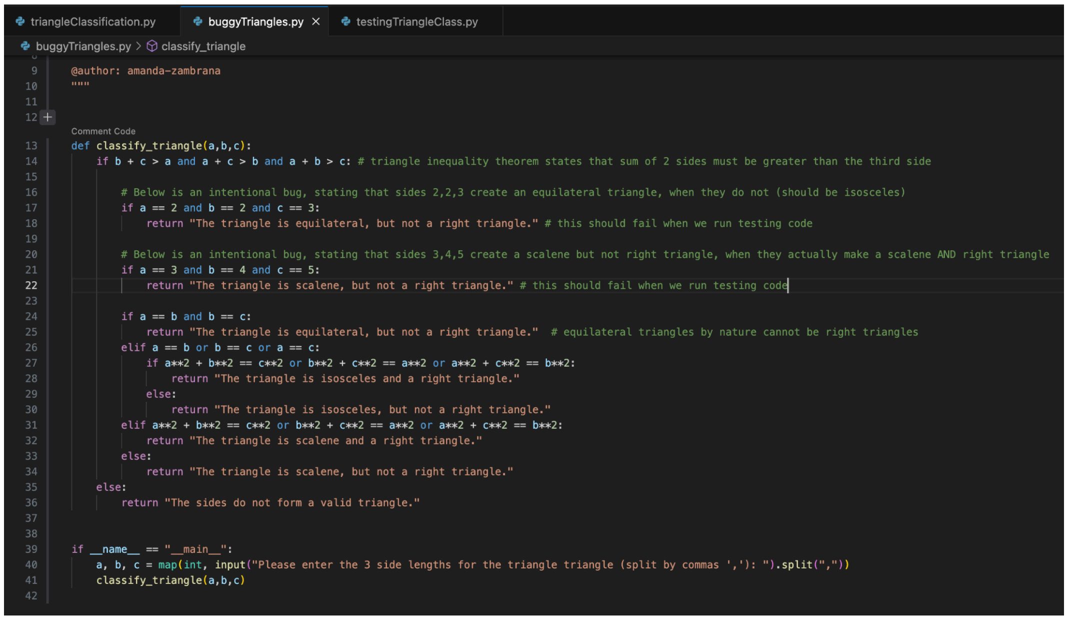
Deliverable 1: All 3 python files are submitted on canvas

Deliverable 2: Screenshots of the input and output of running the program.
First, here is a screenshot of the code with no inclusion of bugs that should run through the testing without any failures.



```python
"""
This code is for the classification of triangles. It takes the 3 sides of
a triangle and determines whether it is equilateral, isosceles, or scalene.
It also determines whether or not it is a right triangle.
Also, if the 3 sides do not form a valid triangle, it returns that info.

@author: amanda-zambrana
"""

Comment Code
def classify_triangle(a, b, c):
    if b + c > a and a + c > b and a + b > c: # triangle inequality theorem states that sum of 2 sides must be greater than the third side
        if a == b and b == c:
            return "The triangle is equilateral, but not a right triangle."  # equilateral triangles by nature cannot be right triangles
        elif a == b or b == c or a == c:
            if a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or a**2 + c**2 == b**2:
                return "The triangle is isosceles and a right triangle."
            else:
                return "The triangle is isosceles, but not a right triangle."
        elif a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or a**2 + c**2 == b**2:
            return "The triangle is scalene and a right triangle."
        else:
            return "The triangle is scalene, but not a right triangle."
    else:
        return "The sides do not form a valid triangle."


# test cases with different side lengths to see if the code is functioning correctly
classify_triangle(3,4,5) # scalene & right
classify_triangle(2,2,3) # isosceles & not right
classify_triangle(5,5,5) # equilateral & not right
classify_triangle(2,3,10) # not a valid triangle
```

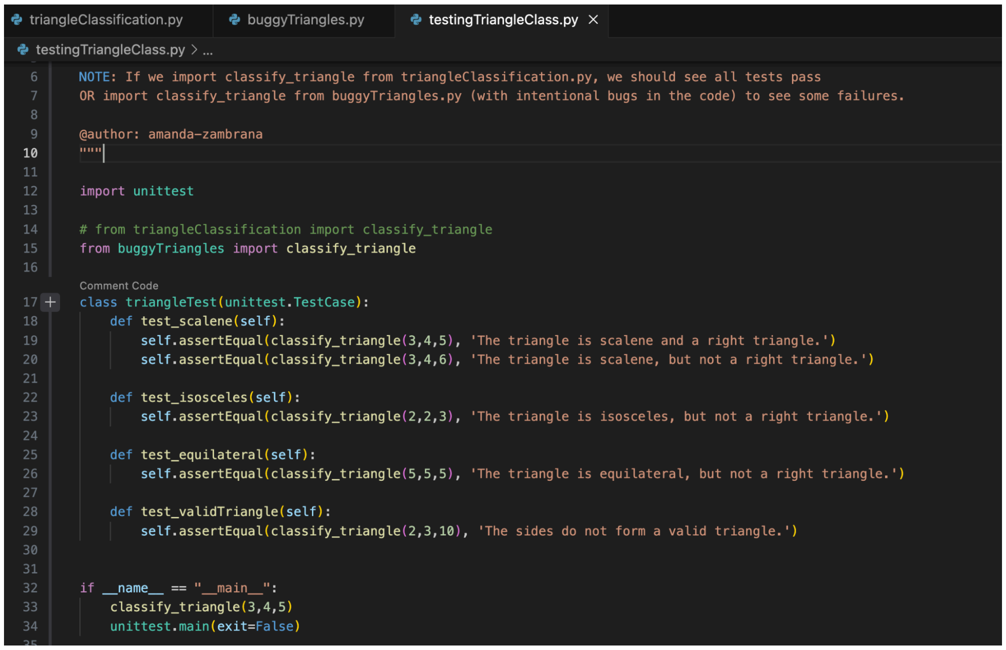
Next, this is a screenshot of the altered version of the code that has bugs included and has two failures when run with the testing code.



triangleClassification.py
buggyTriangles.py
testingTriangleClass.py
buggyTriangles.py > classify_triangle

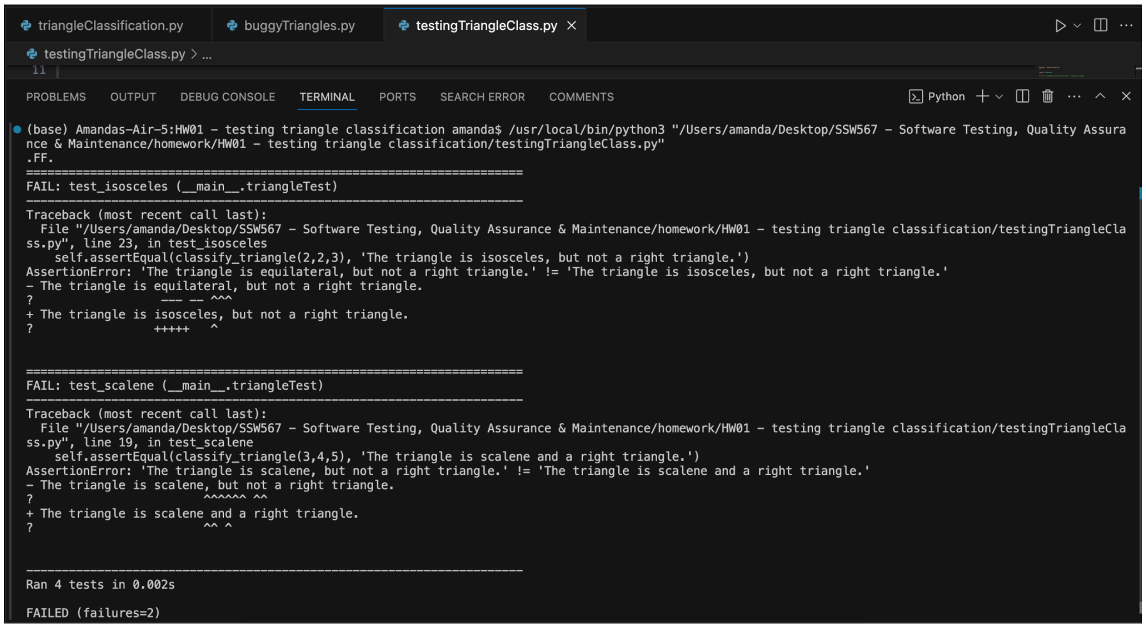
```python
@author: amanda-zambrana
"""


Comment Code
def classify_triangle(a,b,c):
    if b + c > a and a + c > b and a + b > c: # triangle inequality theorem states that sum of 2 sides must be greater than the third side

        # Below is an intentional bug, stating that sides 2,2,3 create an equilateral triangle, when they do not (should be isosceles)
        if a == 2 and b == 2 and c == 3:
            return "The triangle is equilateral, but not a right triangle." # this should fail when we run testing code

        # Below is an intentional bug, stating that sides 3,4,5 create a scalene but not right triangle, when they actually make a scalene AND right triangle
        if a == 3 and b == 4 and c == 5:
            return "The triangle is scalene, but not a right triangle." # this should fail when we run testing code

        if a == b and b == c:
            return "The triangle is equilateral, but not a right triangle."  # equilateral triangles by nature cannot be right triangles
        elif a == b or b == c or a == c:
            if a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or a**2 + c**2 == b**2:
                return "The triangle is isosceles and a right triangle."
            else:
                return "The triangle is isosceles, but not a right triangle."
        elif a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or a**2 + c**2 == b**2:
            return "The triangle is scalene and a right triangle."
        else:
            return "The triangle is scalene, but not a right triangle."
    else:
        return "The sides do not form a valid triangle."


if __name__ == "__main__":
    a, b, c = map(int, input("Please enter the 3 side lengths for the triangle triangle (split by commas ','): ").split(","))
    classify_triangle(a,b,c)
```

Next, this is a screenshot of the testing code with the input test cases used to test the buggyTriangle code.



```python
NOTE: If we import classify_triangle from triangleClassification.py, we should see all tests pass
OR import classify_triangle from buggyTriangles.py (with intentional bugs in the code) to see some failures.

@author: amanda-zambrana
"""

import unittest

# from triangleClassification import classify_triangle
from buggyTriangles import classify_triangle

class triangleTest(unittest.TestCase):
    def test_scalene(self):
        self.assertEqual(classify_triangle(3,4,5), 'The triangle is scalene and a right triangle.')
        self.assertEqual(classify_triangle(3,4,6), 'The triangle is scalene, but not a right triangle.')

    def test_isosceles(self):
        self.assertEqual(classify_triangle(2,2,3), 'The triangle is isosceles, but not a right triangle.')

    def test_equilateral(self):
        self.assertEqual(classify_triangle(5,5,5), 'The triangle is equilateral, but not a right triangle.')

    def test_validTriangle(self):
        self.assertEqual(classify_triangle(2,3,10), 'The sides do not form a valid triangle.')


if __name__ == "__main__":
    classify_triangle(3,4,5)
    unittest.main(exit=False)
```

Lastly, here is a screenshot of the output of running the testingTriangleClass test code that shows how I have tested the bugged code. It shows that those two bugs that were intentionally put into the code have caused two failures in the output here.



```
(base) Amandas-Air-5:HW01 - testing triangle classification amanda$ /usr/local/bin/python3 "/Users/amanda/Desktop/SSW567 - Software Testing, Quality Assura
nce & Maintenance/homework/HW01 - testing triangle classification/testingTriangleClass.py"
.FF.
======================================================================
FAIL: test_isosceles (__main__.triangleTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amanda/Desktop/SSW567 - Software Testing, Quality Assurance & Maintenance/homework/HW01 - testing triangle classification/testingTriangleCla
ss.py", line 23, in test_isosceles
    self.assertEqual(classify_triangle(2,2,3), 'The triangle is isosceles, but not a right triangle.')
AssertionError: 'The triangle is equilateral, but not a right triangle.' != 'The triangle is isosceles, but not a right triangle.'
- The triangle is equilateral, but not a right triangle.
?                  --- -- ^^^
+ The triangle is isosceles, but not a right triangle.
?                 +++++   ^


======================================================================
FAIL: test_scalene (__main__.triangleTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/amanda/Desktop/SSW567 - Software Testing, Quality Assurance & Maintenance/homework/HW01 - testing triangle classification/testingTriangleCla
ss.py", line 19, in test_scalene
    self.assertEqual(classify_triangle(3,4,5), 'The triangle is scalene and a right triangle.')
AssertionError: 'The triangle is scalene, but not a right triangle.' != 'The triangle is scalene and a right triangle.'
- The triangle is scalene, but not a right triangle.
?                        ^^^^^^ ^^
+ The triangle is scalene and a right triangle.
?                        ^^ ^


----------------------------------------------------------------------
Ran 4 tests in 0.002s

FAILED (failures=2)
```

Deliverable 3: My experience with this assignment.

- The challenges that I encountered with this assignment include learning the basic ins and outs of UnitTest for python, since I have not used it before.

- I thought that the requirements specifications for this assignment were very clear and detailed enough to make the development process of this code understandable. It was not difficult to translate the standard language written requirements into python code.
- The only challenge that I encountered with the tools was using trial and error a couple of times to get the hang of testing using UnitTest since I was new to this tool prior to this assignment.
- To determine that the test cases I had would accurately test my code, I used very simple triangle side length combinations to avoid over complicating my testing process. For example, using 5,5,5 for equilaterals or 3,4,5 for scalene right triangles are very easy side lengths to test with since I didn't have to second guess whether it was the code working/not working correctly or if I had made an error in the actual side lengths with what triangle type they really should be.
    - Also, I made sure that I had multiple types of triangle test cases so that I was able to test for many types of triangles. This helped me feel that I had more sufficient test cases to be finished testing since I was able to test many different aspects of the code.

Deliverable 4: Here is the URL for my GitHub repo with all of my files - https://github.com/amanda-zambrana/triangle-classification-testing

---

Requirements (questions from class PPT):

"Write a function classify_triangle() that takes three parameters: a, b, c representing the lengths of the sides of a triangle. Return a string that specifies whether the triangle is scalene, isosceles, or equilateral, and also whether it is a right triangle as well."

- *Are these good requirements?*
- *Are they complete? (ie. would you answer everything and pass all of the tests?)*
- *What's missing from the requirements? (ie. what might be added to these requirements?)*

- Yes, I would say that these are good requirements because they mention what parameters need to be taken in as well as the three different types of triangles that we can classify the input into. It even includes that we should determine whether the triangle is a right triangle or not. However, I would say that these requirements could be better.
- I would say that these requirements are incomplete because they might lead to misleading outcomes in testing. For instance, since it does not pass the requirement that three sides must form a valid triangle before it classifies which type of triangle, a test case might look like it is a certain type of triangle based on the requirements for that type although it does not pass the requirement for any triangle in general.

- Something that we can add to these requirements is to check whether or not the three given sides actually create a valid triangle based on the triangle inequality theorem (which states that the sum of 2 sides must be greater than the third side in order to form a valid triangle) before going into determining which type of triangle it is and whether it is a right triangle or not. I chose to add this requirement into my code.