EXPLORER

HOMEWORK

- helloworld.py
- triangle.py

Welcome | helloworld.py | triangle.py

triangle.py > ...

```python
import unittest
import math

def classifyTriangle(a,b,c):
    """

    This function returns a string with the type of triangle from three  values
    corresponding to the lengths of the three sides of the Triangle.

    return:
        If all three sides are equal, return 'Equilateral'
        If exactly one pair of sides are equal, return 'Isoceles'
        If no pair of  sides are equal, return 'Scalene'
        If not a valid triangle, then return 'NotATriangle'
        If the sum of any two sides equals the squate of the third side, then return 'Right'


    """
    # First, test if the values create a triangle, then test what type of triangle it is
    if a + b <= c or a + c <= b or b + c <= a:
        return 'NotTriangle'
    #Right and Right Isoceles Test
    if a**2 + b**2 == c**2 or b**2 + c**2 == a**2 or a**2+c**2 == b:
        if a == b or b == c:
            return 'Right Isoceles'
        return 'Right'
    #Equilateral test
    if a == b and b == c:
        return 'Equilateral'
    #Isoceles Test
    elif a == b or b == c or a == c:
        return 'Isoceles'

    else:
        return 'Scalene'


def runClassifyTriangle(a, b, c):
    """ invoke classifyTriangle with the specified arguments and print the result """
    print('classifyTriangle(',a, ',', b, ',', c, ')=',classifyTriangle(a,b,c),sep="")


# The remainder of this code implements the unit test functionality

# https://docs.python.org/3/library/unittest.html has a nice description of the framework

class TestTriangles(unittest.TestCase):
    # define multiple sets of tests as functions with names that begin
    # with 'test'.  Each function may include multiple tests
    def testRightTriangle(self): # test invalid inputs
        self.assertEqual(classifyTriangle(3,4,5),'Right','3,4,5 is a Right Scalene triangle')
        self.assertEqual(classifyTriangle(5,12,13), 'Right', '5,12,13 is a Right Scalene Triangle')

    def testEquilateralTriangle(self):
        self.assertEqual(classifyTriangle(1,1,1),'Equilateral','1,1,1 is an Equilateral triangle')

    def testIsocelesTriangle(self):
        self.assertEqual(classifyTriangle(2,2,3),'Isoceles','2,2,3 is an Isoceles triangle')
        self.assertEqual(classifyTriangle(1, 1, math.sqrt(2)), 'Isoceles', '1,1, math.sqrt(2) is an Isoceles triangle')
        self.assertEqual(classifyTriangle(3, 3, math.sqrt(18)), 'Isoceles', '3,3, math.sqrt(18) is a Right Isoceles triangle')

    def testScaleneTriangle(self):
        self.assertEqual(classifyTriangle(4,2,3),'Scalene','4,2,3 is a Scalene triangle')

    def testNotTriangle(self):
        self.assertEqual(classifyTriangle(1,2,30), 'NotTriangle', '1, 2, 30 is not a triangle')

    #def testRightIsoceles
    #self.assertEqual(classifyTriangle(1,1,math.sqrt(2)), 'Right', '1,1, math.sqrt(2) is a Right Isoceles triangle')

    #def testInput(self):
        #self.assertFalse(classifyTriangle(null,null,null), 'Input', 'no numbers are inputed')

if __name__ == '__main__':
    # examples of running the code
    runClassifyTriangle(1,1,math.sqrt(2))
    runClassifyTriangle(7,5,9)
    runClassifyTriangle(9,16,25)
    runClassifyTriangle(100,4,2)
    runClassifyTriangle(7,4,4)
    runClassifyTriangle(1,1,1)

    #unittest.main(exit=False) # this runs all of the tests - use this line if running from Spyder
    unittest.main(exit=True) # this runs all of the tests - use this line if running from the command line
```

PROBLEMS | OUTPUT | TERMINAL | PORTS | DEBUG CONSOLE

Python

```
C:\Users\Gleb\OneDrive - stevens.edu\SSW 567\Homework>C:/Users/Gleb/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Gleb/OneDrive - stevens.edu/SSW 567/Homework/triangle.py"
classifyTriangle(1,1,1.4142135623730951)=Isoceles
classifyTriangle(7,5,9)=Scalene
classifyTriangle(9,16,25)=NotTriangle
classifyTriangle(100,4,2)=NotTriangle
classifyTriangle(7,4,4)=Isoceles
classifyTriangle(1,1,1)=Equilateral
.....
----------------------------------------------------------------------
Ran 5 tests in 0.000s

OK

C:\Users\Gleb\OneDrive - stevens.edu\SSW 567\Homework>
```

OUTLINE

TIMELINE

Gleb Myshkin, Matthew Feroz

We pledge our honor we have abided by the Stevens Honor System.

**Deliverable 3**: Describe your experience with this assignment, specifically:

- What challenges did you encounter with this assignment, if any?
    - We had some problems with making a test for the right scalene and right isosceles triangle. We couldn't implement a test so that it could see if the numbers that we input were both a right triangle and an isosceles/scalene triangle as both types of triangles can also be a right triangle at the same time. We decided to not use this test scenario and only used the right, scalene, equilateral, isosceles, and non-triangle tests. In the beginning, we also had a few issues with the math used to find out what type of triangle we are dealing with, but in the end, we were able to find every equation needed for the tests that we conducted.
- What did you think about the requirements specification for this assignment?
    - I believe that the requirements could have been a little more specific if we needed to find every single type of triangle (right isosceles or right scalene). In our code we only implemented the exact triangles that were asked but we were a bit confused if we needed more.
- What challenges did you encounter with the tools?
    - In the beginning, we had some problems with the Live Share extension that we used on Visual Studio Code to collaborate with each other (me and Matthew Feroz). It was a bit confusing on how to connect to each other for us to work on the code, but we eventually sorted it out. The only thing that I couldn't do was run the code while working on it together, only Matthew had the ability to do that. I was able to run it after finishing the collaboration with Matthew.
- Describe the criteria you used to determine that you had sufficient test cases, i.e. how did you know you were done?
    - The main criteria that we used for our test cases was making sure that the output of our program successfully ran every triangle number we inputted. This would allow us to know if the code that we made successfully tests the input triangles because we know that the numbers that we inputted were correct.