**Assignment 1**

**File 1: Classify\_triangle**

def classify\_triangle(a, b, c):

    # First, check if the input values form a valid triangle

    if a + b <= c or b + c <= a or a + c <= b:

        return "Not a triangle"

    # Check for equilateral triangle

    if a == b == c:

        return "Equilateral"

    # Check for right triangle using the Pythagorean theorem

    if a\*\*2 + b\*\*2 == c\*\*2 or b\*\*2 + c\*\*2 == a\*\*2 or a\*\*2 + c\*\*2 == b\*\*2:

        return "Right"

    # Check for isosceles triangle

    if a == b or b == c or a == c:

        return "Isosceles"

    # If none of the above, it's a scalene triangle

    return "Scalene"

**File 2: Test\_classify\_triangle**

import unittest

from classify\_triangle import classify\_triangle

class TestClassifyTriangle(unittest.TestCase):

    def test\_equilateral(self):

        self.assertEqual(classify\_triangle(3, 3, 3), "Equilateral")

    def test\_isosceles(self):

        self.assertEqual(classify\_triangle(5, 5, 3), "Isosceles")

        self.assertEqual(classify\_triangle(5, 3, 5), "Isosceles")

        self.assertEqual(classify\_triangle(3, 5, 5), "Isosceles")

    def test\_scalene(self):

        self.assertEqual(classify\_triangle(3, 4, 5), "Right")  # Right triangle is also Scalene

        self.assertEqual(classify\_triangle(6, 7, 8), "Scalene")

    def test\_right(self):

        self.assertEqual(classify\_triangle(3, 4, 5), "Right")

        self.assertEqual(classify\_triangle(5, 12, 13), "Right")

        self.assertEqual(classify\_triangle(8, 15, 17), "Right")

    def test\_invalid\_triangle(self):

        self.assertEqual(classify\_triangle(1, 2, 3), "Not a triangle")

        self.assertEqual(classify\_triangle(5, 1, 1), "Not a triangle")

    def test\_zero\_or\_negative\_sides(self):

        self.assertEqual(classify\_triangle(0, 5, 5), "Not a triangle")

        self.assertEqual(classify\_triangle(-1, 4, 5), "Not a triangle")

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

**Output:**

**A screen shot of a computer program

Description automatically generated**

**Challenges:**

Working on this assignment was a valuable learning experience. I started by understanding how triangles are classified based on their sides and then implemented the function step by step.

One challenge I faced was ensuring the function handled invalid triangles correctly, like when the sum of two sides is less than or equal to the third side.

I resolved this by adding a check at the beginning of the function to identify such cases and return "Not a triangle."

Writing the test cases helped me confirm that the function works as expected for all types of triangles.

Overall, the process gave me a better understanding of both Python programming and unit testing.

**Github Repo link:**

https://github.com/Kvekaria1/SE\_567