1. Assignment Description

This assignment applies the testing techniques of static code testing and analysis. The objective of this assignment is to apply the techniques from the lecture to static testing of the classify Triangles program.

2. Author- Ryan Davis

3. Summary

Results:

GitHub Repository containing triangle classification: <https://github.com/ryry91021/ssw567/tree/main/hw/hw4/hw4b>

* Initial Code Analysis:
  + Coverage (Coverage.py):A close-up of a paper

    Description automatically generated
  + Static Code Analysis (Pylint): A black text on a white background

    Description automatically generated
* Original Code:

Original testing Code:

import unittest

from classify\_triangles import classify\_triangle

class testTriangle(unittest.TestCase):

def test\_isEquilateral(self):

self.assertEqual(classify\_triangle(500, 500, 500), "equilateral")

def test\_isIsosceles(self):

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

def test\_isScalene(self):

self.assertEqual(classify\_triangle(15, 34, 32), "scalene")

def test\_isRight(self):

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

def test\_negativeLengthError(self):

with self.assertRaises(ValueError):

classify\_triangle(-1, 2, 5)

def test\_improperTriangle(self):

with self.assertRaises(ValueError):

classify\_triangle(1000, 1, 1)

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

unittest.main()

Original classify triangle:

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

if any(side <= 0 for side in [a, b, c]):

raise ValueError("Lengths must be positive")

if not (a + b > c and a + c > b and b + c > a):

raise ValueError("This is not a working triangle")

else:

if a==b==c:

return "equilateral"

elif a==b!=c or a!=b==c or a==c!=b:

return "isosceles"

elif ((a\*\*2)+(b\*\*2)==(c\*\*2)) or ((b\*\*2)+(c\*\*2)==(a\*\*2)) or ((a\*\*2)+(c\*\*2)==(c\*\*2)):

return "right"

elif a!=b!=c:

return "scalene"

return

* Changes applied:
  + Classify\_triangle.py:
    - Removed trailing whitespace.
    - Added a final newline at the end of the file.
    - Added a function docstring.
    - Removed unnecessary else after raise.
    - Replaced elif with if where return already exits the function.
    - Ensured all return statements have a return value
  + Test\_classify\_triangles.py:
    - Covered different placements of the two equal sides in isosceles triangles.
    - Added more unique cases for scalene triangles.
    - Included more known Pythagorean triples for right triangles.
    - Checked behavior when invalid side lengths, including negative and zero values, are given.
    - Ensured that improper triangles violating the triangle inequality are correctly rejected.
    - Checked if floating-point numbers work correctly.
* Post-change code analysis:
  + Coverage: A close-up of a white background

    Description automatically generated
  + Static Code Analysis:A white background with black text

    Description automatically generated

Reflection:

* What I learned:
  + How to use Coverage in order to determine that test cases are properly testing the function
  + How to use static code analysis in order to improve functions and code.
* What went well:
  + Coverage and static code analysis with Pylint successfully addressed issues in the code.
* Challenges faced:
  + Determining which file to run coverage and pylint on was confusing, however I was able to learn that coverage is ran on the test cases, and pylint can be ran on both to find issues with each program.
  + I was able to attain an improvement in both coverage and improved the code’s rating in static code analysis.

5. Honor Pledge

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

Signed: *Ryan Davis*