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/testing_triangle_classification>

* 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()

* 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 interact with APIs in Python using requests.
  + The importance of writing unit tests before implementation.
  + How to set up CircleCI for continuous integration.
* What went well:
  + Unit tests confirmed that the function handles various cases.
  + Using CircleCI helped automate testing and ensured the code was always working.
* Challenges faced:
  + Encountered GitHub API rate limits, requiring token authentication.
  + Travis-CI not displaying GitHub as an option, leading to using CircleCI instead.
  + Had issues with package installations (requests, dotenv) on CircleCI.

4. Detailed Results

GitHub Repository- <https://github.com/ryry91021/ssw567/tree/main/hw/hw4/GitHubApi567-hw4a>

GitHub API Calls

* Retrieve repositories:
  + https://api.github.com/users/{username}/repos
* Retrieve commits for a repository:
  + https://api.github.com/repos/{username}/{repo}/commits

Implemented Code

* The program retrieves repositories and counts commits using requests and JSON parsing.

Example Output:

Repo: ProjectA, Number of commits: 14

Repo: ProjectB, Number of commits: 32

5. Honor Pledge

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

Signed: *Ryan Davis*