**Report: Static Code Analysis and Code Coverage for Triangle Program**

**Static Code Analysis and Code Coverage for Triangle Program**

**Summary**

**This report details the application of**

**Pylint for static code analysis and Coverage.py for code coverage testing on a Python program that classifies triangles. Initially, the static analysis revealed several issues related to code style and documentation, including missing docstrings and redefined variable names. After fixing these problems, the Pylint score was significantly improved from**

**7.50/10 to 9.62/10. The primary goal for code coverage was to exceed 80%. By configuring the coverage tool to exclude non-testable, command-line interface code, we successfully achieved**

**100% coverage of the core application logic.**

**GitHub Repository URL**

**https://github.com/Kvekarial/Stevens-SSW567/HW03**

**1. Static Code Analysis**

**Tool Used: Pylint**

**Command Executed: python -m pylint triangle.py**

**Analysis Before Changes**

The initial Pylint analysis of the triangle.py file identified several areas for improvement.

* **Key Issues Identified:**  
  A screenshot of a computer program

  AI-generated content may be incorrect.

**Analysis After Changes**

After refactoring the code to address the issues above, the Pylint score improved to

**9.62/10**. The only remaining suggestion was a minor style issue,

A computer code with white text

AI-generated content may be incorrect.

**2.** Code Coverage Analysis

**Commands Executed:**

* **To run tests and collect data: python -m coverage run -m pytest test\_triangle.py**
* **To generate a report: python -m coverage report -m**
* **Initial Test Cases & Final Coverage**
* The initial test suite was already comprehensive, covering all logical branches of the classify\_triangle function. However, the initial report showed less than 100% coverage because it included the main() function and the if \_\_name\_\_ == "\_\_main\_\_": block, which are not executed by the unit tests.
* To reflect the true coverage of the testable logic, the code was modified with # pragma: no cover comments. This standard practice tells the tool to ignore the non-testable, interactive parts of the script. After this change, the final report confirmed **100% code coverage** for all relevant application logic.

A screen shot of a computer

AI-generated content may be incorrect.

**Screen shot of Coverage HTML Report:** A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.