**HW 05 - Static Code Analysis**

SSW-567 – SW Testing, Qual. Assur. & Maint

Srikanth Uppada

*“I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.”*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Assignment Description**:

The objective of this assignment is to apply the techniques from the lecture to static testing of your Triangles program.    Specifically:

* You will run a static code analyzer on your code, e.g. Pylint, identify and fix any problems reported by the static code analyzer;
* You will run a code coverage tool on your code, e.g. Coverage.py, and extend your test cases to demonstrate at least 80% code coverage;

In this assignment, you will need to download and install the tools that you will need for static code analysis and code coverage.  You will then run those tools locally on your laptop to get the results.

Any changes that you make to your programs should be pushed up to GitHub.

## **Deliverables:**

Submit a report with the following information :

1. The GitHub URL containing the code that was analyzed
2. The name and output of the static code analyzer tool you used;
3. The name and output of the code coverage tool you used;
4. Identify both your original test cases and new test cases that you created to achieve at least 80% code coverage.
5. Attach screen shots of the output of the static code analyzer as well as code coverage.  You should show a screen shot of the analysis results both before and after any changes that you make to your programs:
   1. Static code analysis report on original program
   2. Code coverage report before any changes to the program
   3. Static code analysis report after you have made changes to eliminate issues
   4. Code coverage after any changes to the programs (coverage should be > 80%)

**Summary**:

To identify any static code issues on my previous triangle program, I have used the tool PyLint. This tool has identified few code and coding standard issues. Majorly the coding standard and format issues were reflected. Once all of identified issues were resolved, I was able to produce a error free program. During source code development I didn’t expect to have any static code issues. The tool really helped to identify static code issues, where sometimes a human eye cannot catch the simplest issues. However the tool Coverage.py is another useful code coverage tool, It enables the tester to identify any missed out test cases. In this way all of the possible test inputs can be driven over a function or a program.

1. GitHub URL: <https://github.com/sriksrik7/SSW-567/tree/master/HW05>

Github Repo: <https://github.com/sriksrik7/SSW-567>

Folder: HW05

1. The static code analysis tool **PyLint** is used to identify code issues.
2. **Coverage.py** tool is used to identify code coverage
3. The existing test cases for the triangle program has already generated a report of 100% coverage. To full fill the objective of this assignment, I have intentionally remove few test cases (1 t 8 and 15 to 21). Assuming these tests are not available, here is my original test cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sl # | Test cases | Inputs | | | Output |
| Side A | Side B | Side C |
| 1 | Test Case7 | 15 | 17 | 18 | Right |
| 2 | Test Case8 | 2 | 5 | 6 | Scalene |
| 3 | Test Case9 | 201 | 15 | 24 | InvalidInput |
| 4 | Test Case10 | 50 | 201 | 40 | InvalidInput |
| 5 | Test Case11 | 70 | 120 | 201 | InvalidInput |
| 6 | Test Case12 | -1 | 5 | 6 | InvalidInput |
| 7 | Test Case13 | 8 | -6 | 3 | InvalidInput |
| 8 | Test Case14 | 16 | 13 | -4 | InvalidInput |

Based on above test cases, the coverage of the code is 76 %

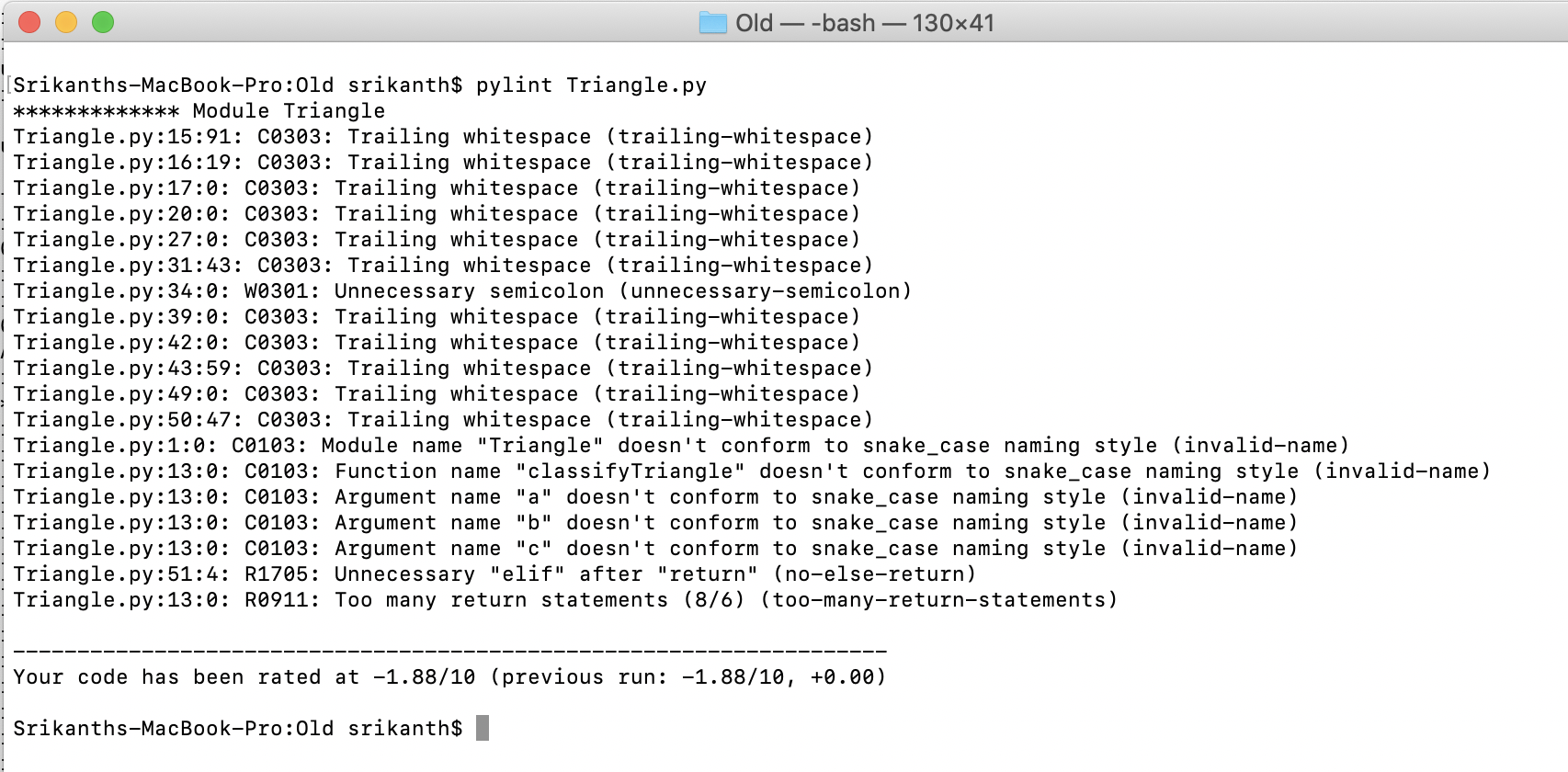


To achieve 100% code average, following additional new tests cases are introduced.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sl # | Test cases | Inputs | | | Output |
| Side A | Side B | Side C |
| 1 | Test Case1 | 3 | 4 | 5 | Right |
| 2 | Test Case2 | 5 | 3 | 4 | Right |
| 3 | Test Case3 | 1 | 1 | 1 | Equilateral |
| 4 | Test Case4 | 2 | 2 | 1 | Isoceles |
| 5 | Test Case5 | 5 | 7 | 7 | Isoceles |
| 6 | Test Case6 | 9 | 3 | 9 | Isoceles |
| 7 | Test Case15 | 0.5 | 1.0 | 2.35 | InvalidInput |
| 8 | Test Case16 | 0 | 199 | 199 | InvalidInput |
| 9 | Test Case17 | 199 | 0 | 199 | InvalidInput |
| 10 | Test Case18 | 199 | 199 | 0 | InvalidInput |
| 11 | Test Case19 | 0 | 12 | 24 | NotATriangle |
| 12 | Test Case20 | 6 | 2 | 3 | NotATriangle |
| 13 | Test Case21 | 1 | 3 | 2 | NotATriangle |

After introducing these additional test cases, the code coverage has bumped up to 100%



1. A. Static code analysis report using PyLint on original code:

B. Static Code analysis report using PyLint on updated/new code: