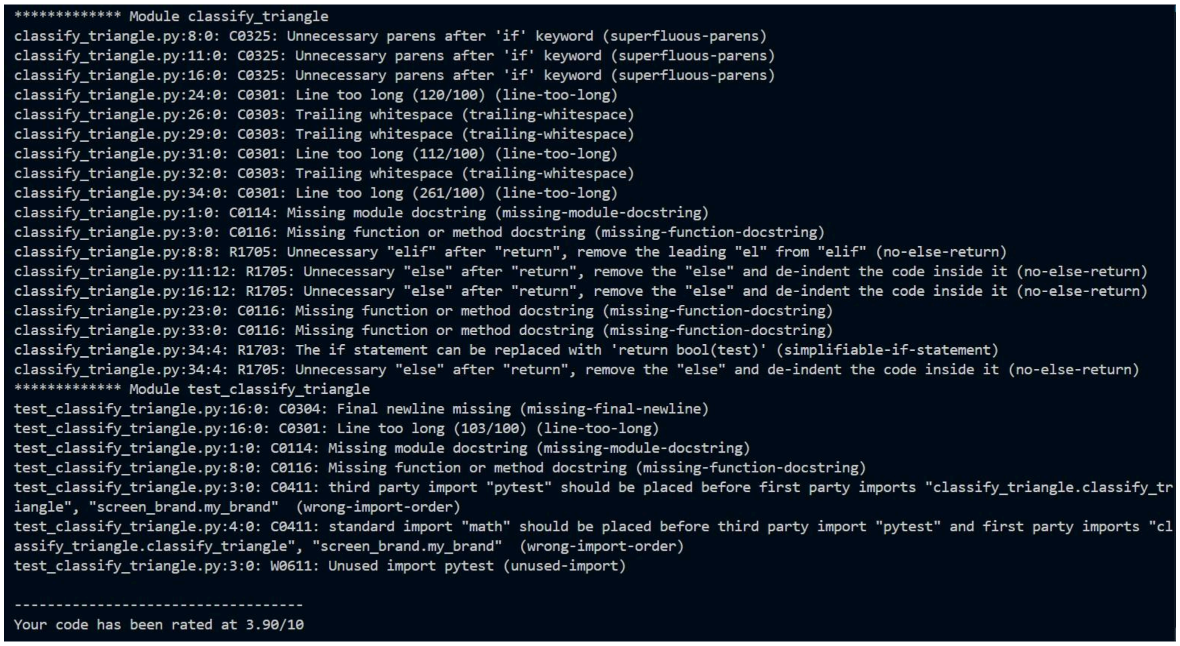
Jason McCauley
10/25/2024
CWID: 20006660
I pledge my honor that I have abided by the Stevens Honor System.

1. Here is the link to my GitHub URL containing that code that was analyzed. More specifically, I analyzed the code in folder “HW01 -- Testing Triangle Classification”, and ignored the “screen_brand.py” file, as we are primarily concerned with running these libraries on the functional code for identifying the type of triangle along with the corresponding unit tests.
   https://github.com/jasonmccauley/SSW_567

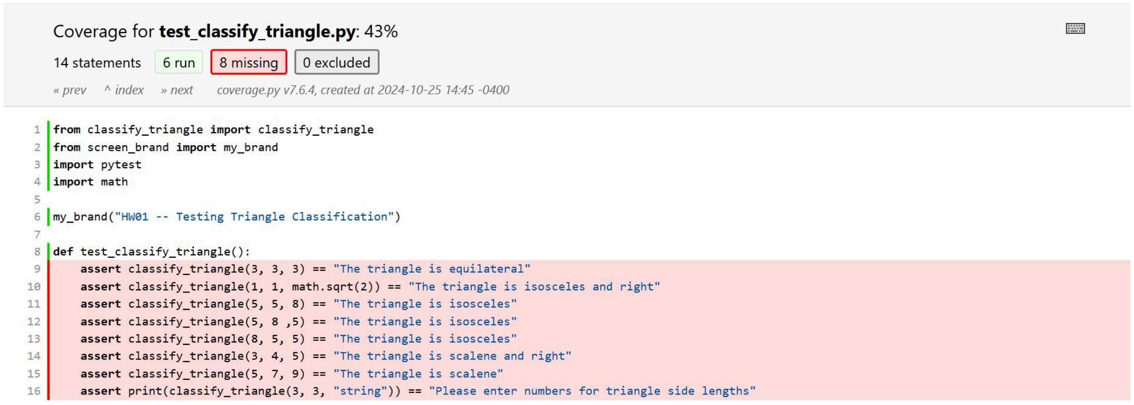
2. For this assignment, I used the static code analyzer tool “Pylint”. Below is the output before making any changes to my code



```
************* Module classify_triangle
classify_triangle.py:8:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
classify_triangle.py:11:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
classify_triangle.py:16:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
classify_triangle.py:24:0: C0301: Line too long (120/100) (line-too-long)
classify_triangle.py:26:0: C0303: Trailing whitespace (trailing-whitespace)
classify_triangle.py:29:0: C0303: Trailing whitespace (trailing-whitespace)
classify_triangle.py:31:0: C0301: Line too long (112/100) (line-too-long)
classify_triangle.py:32:0: C0303: Trailing whitespace (trailing-whitespace)
classify_triangle.py:34:0: C0301: Line too long (261/100) (line-too-long)
classify_triangle.py:1:0: C0114: Missing module docstring (missing-module-docstring)
classify_triangle.py:3:0: C0116: Missing function or method docstring (missing-function-docstring)
classify_triangle.py:8:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
classify_triangle.py:11:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
classify_triangle.py:16:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
classify_triangle.py:23:0: C0116: Missing function or method docstring (missing-function-docstring)
classify_triangle.py:33:0: C0116: Missing function or method docstring (missing-function-docstring)
classify_triangle.py:34:4: R1703: The if statement can be replaced with 'return bool(test)' (simplifiable-if-statement)
classify_triangle.py:34:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
************* Module test_classify_triangle
test_classify_triangle.py:16:0: C0304: Final newline missing (missing-final-newline)
test_classify_triangle.py:16:0: C0301: Line too long (103/100) (line-too-long)
test_classify_triangle.py:1:0: C0114: Missing module docstring (missing-module-docstring)
test_classify_triangle.py:8:0: C0116: Missing function or method docstring (missing-function-docstring)
test_classify_triangle.py:3:0: C0411: third party import "pytest" should be placed before first party imports "classify_triangle.classify_tr
iangle", "screen_brand.my_brand"  (wrong-import-order)
test_classify_triangle.py:4:0: C0411: standard import "math" should be placed before third party import "pytest" and first party imports "cl
assify_triangle.classify_triangle", "screen_brand.my_brand"  (wrong-import-order)
test_classify_triangle.py:3:0: W0611: Unused import pytest (unused-import)

-----------------------------------
Your code has been rated at 3.90/10
```

3. For this assignment, I used the Python code coverage tool “coverage”. Below is the output before making any changes to my code



Coverage for **test_classify_triangle.py**: 43%

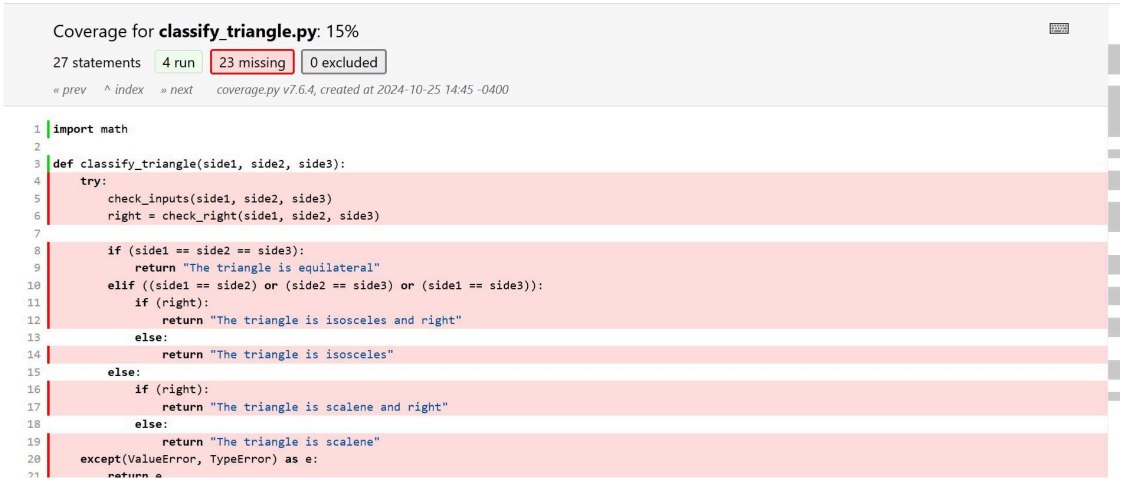
14 statements | 6 run | 8 missing | 0 excluded

« prev ^ index » next coverage.py v7.6.4, created at 2024-10-25 14:45 -0400

```
from classify_triangle import classify_triangle
from screen_brand import my_brand
import pytest
import math

my_brand("HW01 -- Testing Triangle Classification")

def test_classify_triangle():
    assert classify_triangle(3, 3, 3) == "The triangle is equilateral"
    assert classify_triangle(1, 1, math.sqrt(2)) == "The triangle is isosceles and right"
    assert classify_triangle(5, 5, 8) == "The triangle is isosceles"
    assert classify_triangle(5, 8 ,5) == "The triangle is isosceles"
    assert classify_triangle(8, 5, 5) == "The triangle is isosceles"
    assert classify_triangle(3, 4, 5) == "The triangle is scalene and right"
    assert classify_triangle(5, 7, 9) == "The triangle is scalene"
    assert print(classify_triangle(3, 3, "string")) == "Please enter numbers for triangle side lengths"
```



Coverage for **classify_triangle.py**: 15%

27 statements | 4 run | 23 missing | 0 excluded

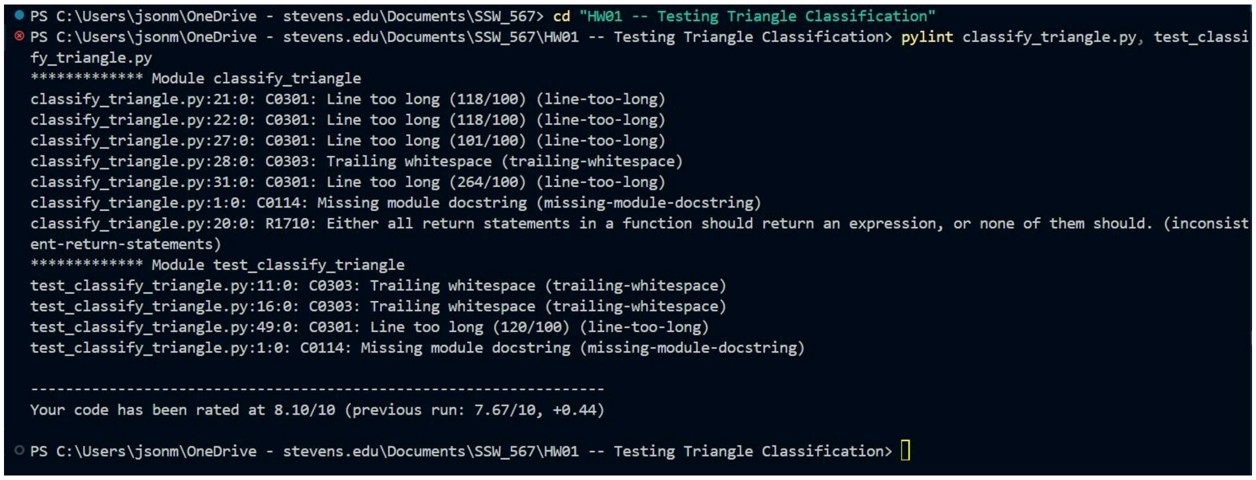
« prev ^ index » next coverage.py v7.6.4, created at 2024-10-25 14:45 -0400

```
import math

def classify_triangle(side1, side2, side3):
    try:
        check_inputs(side1, side2, side3)
        right = check_right(side1, side2, side3)

        if (side1 == side2 == side3):
            return "The triangle is equilateral"
        elif ((side1 == side2) or (side2 == side3) or (side1 == side3)):
            if (right):
                return "The triangle is isosceles and right"
            else:
                return "The triangle is isosceles"
        else:
            if (right):
                return "The triangle is scalene and right"
            else:
                return "The triangle is scalene"
    except(ValueError, TypeError) as e:
        return e
```

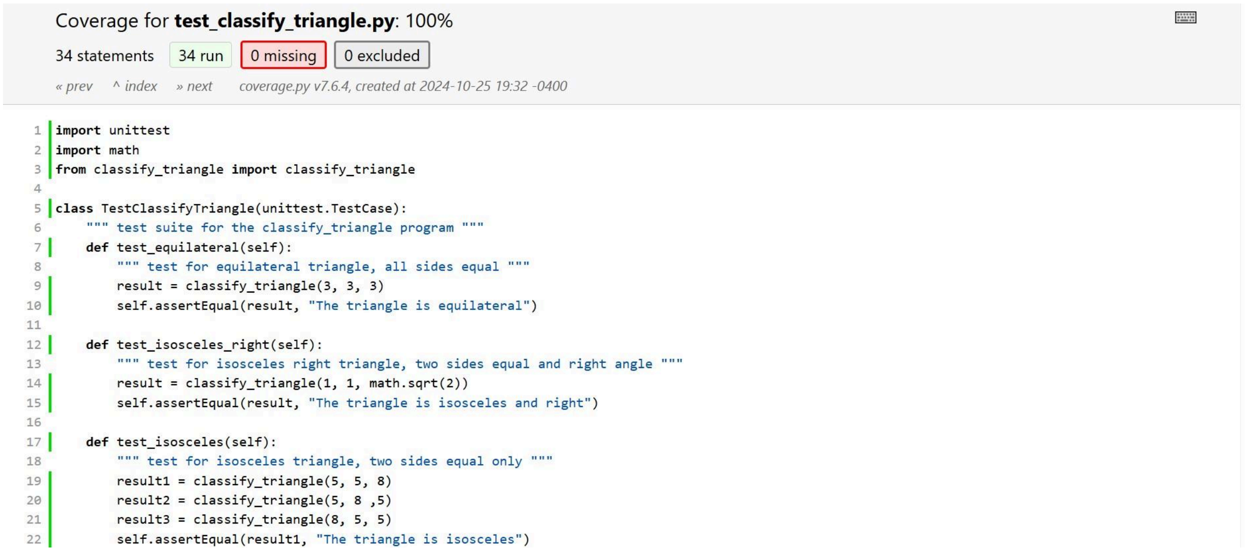
4. Below is the output for Pylint and coverage after making the necessary changes to my code



```
PS C:\Users\jsonm\OneDrive - stevens.edu\Documents\SSW_567> cd "HW01 -- Testing Triangle Classification"
PS C:\Users\jsonm\OneDrive - stevens.edu\Documents\SSW_567\HW01 -- Testing Triangle Classification> pylint classify_triangle.py, test_classi
fy_triangle.py
************* Module classify_triangle
classify_triangle.py:21:0: C0301: Line too long (118/100) (line-too-long)
classify_triangle.py:22:0: C0301: Line too long (118/100) (line-too-long)
classify_triangle.py:27:0: C0301: Line too long (101/100) (line-too-long)
classify_triangle.py:28:0: C0303: Trailing whitespace (trailing-whitespace)
classify_triangle.py:31:0: C0301: Line too long (264/100) (line-too-long)
classify_triangle.py:1:0: C0114: Missing module docstring (missing-module-docstring)
classify_triangle.py:20:0: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsist
ent-return-statements)
************* Module test_classify_triangle
test_classify_triangle.py:11:0: C0303: Trailing whitespace (trailing-whitespace)
test_classify_triangle.py:16:0: C0303: Trailing whitespace (trailing-whitespace)
test_classify_triangle.py:49:0: C0301: Line too long (120/100) (line-too-long)
test_classify_triangle.py:1:0: C0114: Missing module docstring (missing-module-docstring)

------------------------------------------------------------------
Your code has been rated at 8.10/10 (previous run: 7.67/10, +0.44)

PS C:\Users\jsonm\OneDrive - stevens.edu\Documents\SSW_567\HW01 -- Testing Triangle Classification>
```



Coverage for **test_classify_triangle.py**: 100%

34 statements | 34 run | 0 missing | 0 excluded

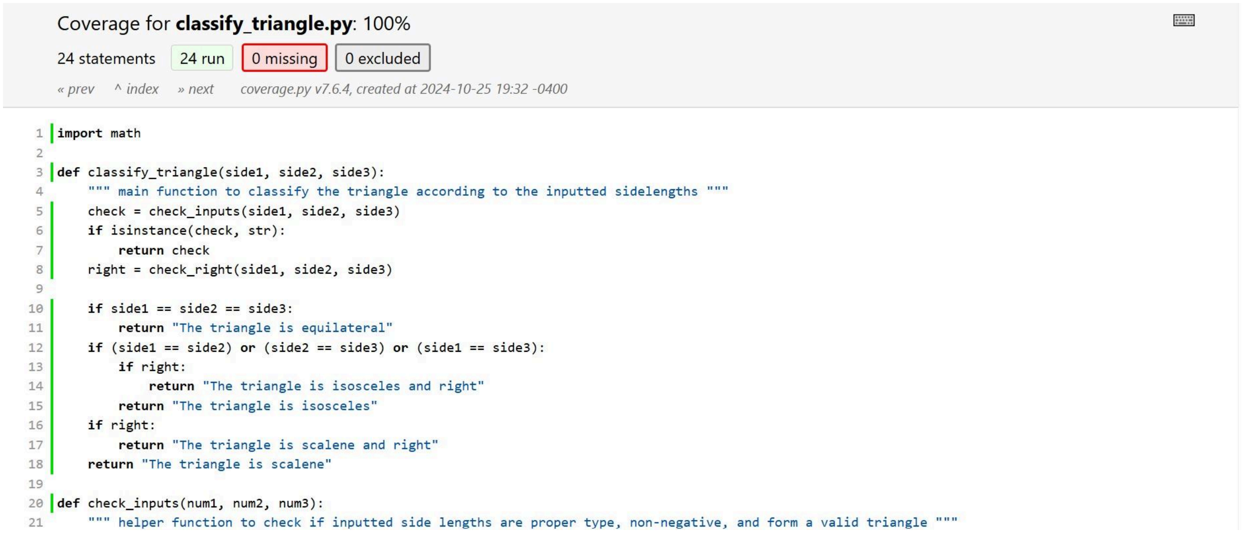
« prev ^ index » next coverage.py v7.6.4, created at 2024-10-25 19:32 -0400

```python
import unittest
import math
from classify_triangle import classify_triangle

class TestClassifyTriangle(unittest.TestCase):
    """ test suite for the classify_triangle program """
    def test_equilateral(self):
        """ test for equilateral triangle, all sides equal """
        result = classify_triangle(3, 3, 3)
        self.assertEqual(result, "The triangle is equilateral")

    def test_isosceles_right(self):
        """ test for isosceles right triangle, two sides equal and right angle """
        result = classify_triangle(1, 1, math.sqrt(2))
        self.assertEqual(result, "The triangle is isosceles and right")

    def test_isosceles(self):
        """ test for isosceles triangle, two sides equal only """
        result1 = classify_triangle(5, 5, 8)
        result2 = classify_triangle(5, 8 ,5)
        result3 = classify_triangle(8, 5, 5)
        self.assertEqual(result1, "The triangle is isosceles")
```



Coverage for **classify_triangle.py**: 100%

24 statements | 24 run | 0 missing | 0 excluded

« prev ^ index » next coverage.py v7.6.4, created at 2024-10-25 19:32 -0400

```python
import math

def classify_triangle(side1, side2, side3):
    """ main function to classify the triangle according to the inputted sidelengths """
    check = check_inputs(side1, side2, side3)
    if isinstance(check, str):
        return check
    right = check_right(side1, side2, side3)

    if side1 == side2 == side3:
        return "The triangle is equilateral"
    if (side1 == side2) or (side2 == side3) or (side1 == side3):
        if right:
            return "The triangle is isosceles and right"
        return "The triangle is isosceles"
    if right:
        return "The triangle is scalene and right"
    return "The triangle is scalene"

def check_inputs(num1, num2, num3):
    """ helper function to check if inputted side lengths are proper type, non-negative, and form a valid triangle """
```

5. As seen in the screenshots above, I received a rating of 3.90/10 when running Pylint on my original code, which is not quite good. In my classify_triangle.py I had unnecessary parentheses in my if-statements and missing docstrings to explain the purpose of each function. Additionally, a few of the elif- and else-statements were able to be replaced with if-statements, due to the return statements they each contain. For my test_classify_triangle.py file, I was missing docstrings there as well to explain the purpose of each function, and the order of my imports were not considered proper. I addressed each of these concerns when revisiting my code, and as seen, I received a new rating of 8.10/10. The only areas I was losing points for now were lines being too long, specifically at a couple of the if-statements and returns, which I couldn't do much about. Additionally, I lost points for having a line of space between each of my functions, which is more of a stylistic and readability preference than anything.
6. As seen in the screenshots above, the coverage for my original test_classify_triangle.py file was 43%, and the coverage for my classify_triangle.py file was 15%, both of which were suspiciously bad. After looking at the coverage report, I quickly realized that this

was because my test_classify_triangle.py file was somehow never being called, and therefore the code inside the classify_triangle.py file was never being reached. Because, technically, no unit tests were failing, the terminal outputted this as OK. My first step in addressing this was reorganizing my test_classify_triangle.py file to contain all of the test functions within one class. Furthermore, each function would now be testing an individual block of the functional code, such as identifying a scalene right triangle, or an equilateral right triangle. I also added a few additional functions to test_classify_triangle.py to ensure that the validation functions were working as intended, and returning the proper statement, for instance, when negative numbers were entered as side lengths. Doing this, however, did help me realize a logical error in my code – inside of my classify_triangle function, I called the check_inputs function to validate the inputs. The return statements in the check_inputs function would only return the issue, for instance, when a negative number was entered as a side length, to the classify_triangle function, and the rest of the code would still proceed. To address this, I stored the call to the check_inputs function, and if that stored value was of the type string, meaning that an error was handled, that stored value would be returned to where the classify_triangle function was called, which in our case, is the test_classify_triangle.py file. After putting these changes into effect, as seen in the images above, both the classify_triangle.py and test_classify_triangle.py files had 100% coverage.