1. Assignment Description

The objective of this assignment is to (a) develop a set of tests for an existing triangle classification program, (b) use those tests to find and fix defects in that program, and (c) report on your testing results for the Triangle problem. Starting with an existing implementation of the classify triangle program (Triangle.py) and a starter test program (TestTriangle.py), the task involves:

1. Enhancing the set of test cases in TestTriangle.py.
2. Running tests against the original Triangle.py implementation.
3. Creating a test report for the initial implementation.
4. Updating the logic in Triangle.py to fix all bugs found.
5. Running the same test set on the improved classifyTriangle() function.
6. Creating a final test report on the improved logic.

2. Author

Yash SanjayKumar Soni

3. Summary

Results Summary

- Initial implementation passed 8 out of 9 test cases.
- After improvements, all 9 test cases passed successfully.
- Major bug fixed: Incorrect classification of isosceles triangles.
- Additional edge cases and invalid inputs were handled in the improved version.

	Test case 1	Test case 2
Tests planned	9	12
Tests Executed	9	12
Tests Passed	8	12
Defects Found	1	0
Defects Fixed	1	0

Reflection

This job taught me the value of extensive test case design, as well as iterative testing and improvement. The process of finding edge situations and potential faults in the initial implementation helped me improve my analytical skills. What worked well was the systematic approach of creating a strong test suite before attempting to repair the code. This strategy enabled a clear grasp of the current difficulties and gave a reliable means of verifying improvements. One problem was identifying when the test suite was comprehensive enough. I overcome this by taking into account a variety of triangle kinds, edge cases, and incorrect inputs. If I had to perform this project again, I would include more randomized tests to discover unexpected edge cases.

4. Honor Pledge

“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.”

1. Test report for the initial implementation:

Test ID	Input	Expected Results	Actual Result	Pass or Fail
1	3,4,5	Right	Right	Pass
2	5,3,4	Right	Right	Pass
3	1,1,1	Equilateral	Equilateral	Pass
4	2,2,3	Isosceles	Isoceles	Fail
5	3,4,6	Scalene	Scalene	Pass
6	201,3,4	InvalidInput	InvalidInput	Pass
7	-1,2,3	InvalidInput	InvalidInput	Pass
8	1.5,2,3	InvalidInput	InvalidInput	Pass
9	1,1,3	NotATriangle	NotATriangle	Pass

2. Test Report for improved Implementation:

Test ID	Input	Expected Result	Description
1	3, 4, 5	Right	Right triangle (basic case)
2	5, 3, 4	Right	Right triangle (different order)
3	1, 1, 1	Equilateral	Smallest equilateral triangle

Test ID	Input	Expected Result	Description
4	5, 5, 3	Isosceles	Isosceles triangle (two equal sides)
5	4, 6, 6	Isosceles	Isosceles triangle (different order)
6	8, 8, 10	Isosceles	Larger isosceles triangle
7	5, 6, 7	Scalene	Scalene triangle (small)
8	10, 11, 13	Scalene	Scalene triangle (larger)
9	-1, 2, 3	InvalidInput	Negative input
10	1, 1, 201	InvalidInput	Input exceeds maximum
11	0, 0, 0	InvalidInput	All zeros
12	1.5, 2, 3	InvalidInput	Non-integer input
13	"a", "b", "c"	InvalidInput	Non-numeric input
14	1, 1, 3	NotATriangle	Sum of two sides equals third
15	2, 4, 2	NotATriangle	Sum of two sides less than third
16	8, 15, 17	Right	Larger right triangle
17	100, 100, 100	Equilateral	Large equilateral triangle
18	200, 200, 200	Equilateral	Maximum size equilateral triangle
19	200, 200, 201	InvalidInput	Exceeds maximum on one side

Output:

Data Inputs Description:

1. Valid Triangle Inputs:
   - Right Triangles: (3,4,5), (5,3,4), (8,15,17)
     These represent classic right triangles, including different orderings of sides.

   - Equilateral Triangles: (1,1,1), (100,100,100), (200,200,200)
     Testing smallest possible, a larger, and the maximum size equilateral triangle.

   - Isosceles Triangles: (5,5,3), (4,6,6), (8,8,10)
     Various isosceles triangles with different side lengths and orderings.

   - Scalene Triangles: (5,6,7), (10,11,13)
     Different sizes of scalene triangles where no sides are equal.

2. Invalid Inputs:
   - Negative Numbers: (-1,2,3)
     Testing the handling of negative side lengths.

   - Exceeding Maximum: (1,1,201), (200,200,201)
     Testing values that exceed the specified maximum of 200.

   - Zero Values: (0,0,0)
     Checking how the function handles all zero inputs.

   - Non-Integer Values: (1.5,2,3)
     Testing the function's response to floating-point inputs.

   - Non-Numeric Inputs: ("a","b","c")
     Verifying the handling of string inputs.

3. Not A Triangle Inputs:
   - (1,1,3), (2,4,2)

Cases where the sum of two sides is less than or equal to the third side, violating the triangle inequality theorem.

4. Boundary Cases:
   - (200,200,200)
     Testing the upper limit of valid inputs.
   - (1,1,1)
     Testing the lower limit of valid inputs.

These inputs were carefully selected to cover a wide range of scenarios:
- All types of triangles (Right, Equilateral, Isosceles, Scalene)
- Various orderings of side lengths
- Edge cases (minimum and maximum valid values)
- Invalid inputs (negative, zero, non-integer, out-of-range)
- Cases that don't form valid triangles

The goal of this diverse set of inputs is to thoroughly test the classifyTriangle() function, ensuring it correctly handles both valid and invalid cases across the entire range of possible inputs. This comprehensive approach helps identify potential bugs or edge cases that might be missed with a more limited set of test cases.