Assignment Description: HW 07b: Testing a Legacy Systems

Author: Emmanuel Okoro: https://github.com/endiesworld/hw-06b

Summary

Summary of Results

- Two test runs were performed against classifyTriangle(a, b, c):
  - Test Run 1 (before fix): 28 tests executed, 10 passed, 18 failed (including 2 cases that previously raised TypeError conditions).
  - Test Run 2 (after fix): 28 tests executed, 28 passed.
- High-level defects found and fixed:
  - Incorrect triangle inequality check (valid triangles marked as NotATriangle).
  - Equilateral logic ignored c.
  - Scalene logic compared a != b twice and missed a != c.
  - Right-triangle check used linear terms instead of squares and didn't handle permutations.
  - Spelling mismatch: returned Isoceles instead of Isosceles.
  - Input validation order caused TypeError with non-integer inputs (e.g., '3').

A compact matrix of the two runs is below; counts reflect assertion-level cases listed in the CSV files.

	Test Run 1 (Before Fix)	Test Run 2 (After Fix)
Tests Planned	28	28
Tests Executed	28	28
Tests Passed	10	28
Defects Found	6	0
Defects Fixed	0	6

Artifacts

- Initial run table: TEST_REPORT_TABLE.csv
- After-fix run table: TEST_REPORT_TABLE_AFTER_FIX.csv
- Narrative summary: SUMMARY_RESULTS.md

Reflection

- What I learned

  - Well-chosen tests quickly surface multiple, diverse defects, even in a short function.
  - Ordering of validation matters: type checks must precede numeric comparisons.
  - Permutation coverage is important for commutative inputs (triangle sides), especially for Right-triangle detection.
  - A small set of classic triples and boundary inputs provides strong confidence.

- - What worked

  - Combining equivalence partitioning with boundary value analysis gave broad, meaningful coverage with a small set.
  - CSV-based tabular reports made it easy to review expectations vs. actuals across many cases.

- What didn't

  - Relying on only the starter tests would have missed several defects.
  - Minimal tests without permutations can mask order-dependent bugs.

Honor Pledge

I affirm that I have abided by the academic integrity policies for this assignment. The work submitted here is my own. Where I used any external resources, I have acknowledged them appropriately.

Detailed Results

Techniques Used

- Equivalence Partitioning
  - Valid triangles vs. non-triangles vs. invalid inputs (type/range).
  - Within valid triangles: Equilateral, Isosceles, Scalene, Right.
- Boundary Value Analysis
  - Range boundaries: 0, 1, and 200; out-of-range: 201.
  - Triangle inequality edges: cases where x + y == z and just above/below.
- Permutation Coverage for Right Triangles
  - (3,4,5) and (5,12,13) triples with all permutations.
- Negative/Robustness
  - Non-integer types (float/string) and non-triangles (sum of two <= third).

Assumptions and Constraints

- Input domain: integers only; values must be in [1, 200].
- Order of sides should not affect the classification.
- Labels used: InvalidInput, NotATriangle, Right, Equilateral, Isosceles, Scalene.
- Right-triangle determination uses the Pythagorean theorem on sorted sides.

Data Inputs Used

- See TEST_REPORT_TABLE.csv for initial run and TEST_REPORT_TABLE_AFTER_FIX.csv for the after-fix run.
- Representative examples:

    - Right: (3,4,5) and (5,12,13) with permutations
    - Equilateral: (1,1,1), (200,200,200)
    - Isosceles: (2,2,3), (10,10,15), permutations
    - Scalene: (2,3,4), (7,8,9)
    - Non-triangle: (1,1,3), (2,3,5), (10,1,1)
    - Invalid input: (0,1,1), (-1,2,3), (201,10,10), (3.0,4,5), ('3',4,5)

Explanation of the Results

- Initial implementation failures stemmed from:

    - Mis-specified triangle inequality conditions.
    - Incorrect equality checks and misspelling of Isosceles.
    - Right-triangle logic not using squares and being order-sensitive.
    - Type/range validations performed in the wrong order.

- - Fixes applied in Triangle.py:

    - Type validation first, then range checks.
    - Sort sides and apply correct triangle inequality (x + y > z).
    - Pythagorean check (x*x + y*y == z*z) for Right classification.
    - Correct Equilateral/Isosceles/Scalene logic and labels.

- - After fixes, all 28 planned tests pass, giving strong confidence that the function now meets the specification across expected partitions and boundaries.