Unit Test Matrix

There are three different types of unit tests in this project based on machine learning industry standards.

In this section, we'll provide you with a simple framework for understanding how the same test can serve different validation purposes depending on the context, **AND** also help you identify unit tests that must be excluded/removed from your test suite before testing a seeded / golden response.

- A - Pass to Fail Unit test -> **INVALID** unit tests with wrong expectations. REMOVED before testing the seeded response / golden response. **Makes a correct solution fail, hence removed before testing either the seeded response or the reference solution/golden response.**
- B - Fail To Pass unit tests -> **VALID** unit tests that, upon investigation, make the seeded response **FAIL** and justify a **TRUE** response failure.
- C - Pass To Pass unit tests -> **VALID** unit tests that would make the reference solution PASS.

For example, when justifying that the "Seed response" contains a "true failure", the valid unit test would be a "Fail-to-Pass" test. When justifying that the provided program/code solution is the correct "Reference solution", the unit test that was initially a "Fail-to-Pass "test now becomes a "Pass-to-Pass" test, meaning the response needs to pass the unit test!

Pretty cool, right?

Test Classification Matrix

TEST TYPE	SEED RESPONSE TEST RESULT	GOLDEN RESPONSE TEST RESULT	TEST VALIDITY	ACTION REQUIRED
Pass to Pass ✅ Valid test, correct expectations	**PASS**	**PASS**	**VALID ✓**	KEEP IN / ADD TO golden test suite
Fail to Pass ✅ Valid test, demonstrates model failure	**FAIL**	**PASS**	**VALID ✓**	KEEP IN / ADD TO golden test suite
Pass to Fail ❌ Invalid test, wrong expectations	*N/A - Removed*	*N/A - Removed*	⚠ **INVALID** ⚠	**REMOVE** from test suite

Pass - Test succeeds

Fail - Test fails

Invalid - Test has issues

N/A - Removed

- Baseline: The same valid unit tests can be either a Fail-to-Pass or Pass-to-Pass Unit test, depending on the response tested.

- Remember: For justifying a model failure, you need at least one Fail-to-Pass unit test, and for a reference solution, you ONLY need Pass-to-Pass unit tests!

- Pass-to-Fail unit tests are always removed as they would make a good solution fail, and vice versa, a faulty solution pass.

✅ Basic Edge Cases

Please make sure your test suite covers the following basic edge cases where applicable:

- Null or empty inputs
- Zero values
- Maximum and minimum numeric limits (e.g., INT_MAX, INT_MIN)
- Off-by-one boundaries (e.g., index 0, length - 1)
- Invalid data types

⚠️ Constraints

If the prompt includes **specific constraints**, you **must** test edge cases that target those constraints.

They're considered **critical functionality** and need to be validated.

⛔ Avoid

Do not include unit tests that test trivialities or tautologies (conditions that are always true) such as

- `if __name__ != "__main__": pass`
- `def test_pass(self): pass`
- `def test_true(self): self.assertFalse(False is not False)`
- `def test_zero_addition(self): self.assertEqual(0 + 0, 0)`