**1. Introduction to pytest**

**Pytest** is a testing framework in Python. It’s widely used because it’s:

* **Simple and powerful**: Easy syntax, handles complex scenarios.
* **Feature-rich**: Supports fixtures, parameterization, plugins, etc.
* **Popular for automation**: Can be used for API, UI, or unit testing.

**2. Installation**

Start by installing pytest. Open the terminal/command prompt and run:

pip install pytest

Verify the installation:

pytest --version

If pytest version is displayed, it’s successfully installed.

**3. Writing Your First Test**

Create a file named test\_example.py:

python

# test\_example.py

def test\_addition():

assert 1 + 1 == 2

def test\_subtraction():

assert 2 - 1 == 1

Run the test:

pytest

* **Output:**
  + ✅ If test passes: PASSED
  + ❌ If test fails: FAILED

**4. Naming Conventions**

* Test file names should start or end with test (e.g., test\_example.py).
* Test functions should start with test\_.

This ensures pytest identifies and runs your test cases.

**5. Pytest Assertions**

Pytest uses assert to validate conditions. If the condition is True, the test passes.

**Examples:**

python

def test\_assertions():

assert 1 == 1 # Pass

assert "hello" in "hello world" # Pass

assert 3 > 2 # Pass

assert [1, 2, 3] == [1, 2, 3] # Pass

**6. Running Specific Tests**

Run a specific file:

pytest test\_example.py

Run a specific function in a file:

pytest test\_example.py::test\_addition

**7. Detailed Output**

Use the -v option for more detailed output:

pytest -v

**8. Pytest Markers (Tagging Tests)**

Markers are used to group or tag tests.

**Example:**

python

import pytest

@pytest.mark.sanity

def test\_login():

assert True

@pytest.mark.regression

def test\_logout():

assert True

Run tests with a specific marker:

pytest -m sanity

**9. Fixtures (Setup and Teardown)**

Fixtures are used for setting up preconditions for tests (e.g., initializing variables, database connections).

**Example:**

python

import pytest

@pytest.fixture

def setup():

print("\nSetup before each test")

return "Gowtham"

def test\_fixture\_example(setup):

print("Running the test")

assert setup == "Gowtham"

Run the test and observe the setup runs before the test.

**10. Homework**

1. Write a new test file test\_calculations.py with tests for addition, subtraction, multiplication, and division.
2. Use -v to see detailed test results.
3. Experiment with markers and try running only selected tests.

**1. Verify Marker Registration in pytest.ini**

Your pytest.ini should look like this:

ini

[pytest]

markers =

sanity: sanity tests

regression: regression tests

function: function tests

Make sure the file is saved in the **same folder** where you are running the tests (pytestpractise folder in your case).

**2. Check the Command**

Run the following to ensure markers are working:

bash

pytest -v -s -m "sanity"

* Use quotes around the marker name, as sometimes it avoids parsing issues.

**3. Confirm File and Function Names**

* Ensure your test file starts with test\_ (e.g., test\_firsttest.py).
* Ensure all test functions start with test\_.

**4. Clean Cache**

Sometimes pytest caches old configurations. Clear it using:

bash

pytest --cache-clear

Then, re-run the test.

**5. Debug Marker Recognition**

Run the following command to see all the markers pytest recognizes:

bash

pytest --markers

* Verify that your markers (sanity, regression, function) appear in the list.

**6. If the Issue Persists**

If all of the above steps fail:

1. Share the exact content of pytest.ini and the command you are running.
2. Double-check the directory structure.

Finally issue solved , the way to run in below   
  
