Fixtures

◇ What are Fixtures in Pytest?

A **fixture** in `pytest` is a special function that provides **setup and teardown logic** for your tests.

Think of it as a way to **prepare the environment** a test needs before running, and optionally **clean it up afterward**.

Instead of writing the same setup code inside every test, you define it **once** as a fixture and reuse it.

◇ Functions of Fixtures

1. **Setup (Arrange things before the test runs)**
    a. Example: creating a database connection, instantiating a class, preparing sample data.
2. **Provide data or objects to tests**
    a. You can `return` or `yield` values from a fixture, and your tests automatically get them as function arguments.
3. **Teardown (Clean up after the test)**
    a. Example: closing files, clearing memory, resetting a database.
    b. Done with `yield` → the code after `yield` runs **after** the test finishes.
4. **Reusability**
    a. Fixtures can be shared across multiple test functions, modules, or even entire projects.
    b. Reduces duplication and keeps tests clean.
5. **Scoping**
    a. Fixtures can run:
        i. **per test** (`scope="function"`, default)
        ii. **per class** (`scope="class"`)
        iii. **per module** (`scope="module"`)
        iv. **per session** (`scope="session"`)

This lets you control how often the fixture is executed.

◇ Example: Simple Fixture

```
import pytest

@pytest.fixture
def sample_list():
    return [1, 2, 3]

def test_append(sample_list):
    sample_list.append(4)
    assert sample_list == [1, 2, 3, 4]

def test_length(sample_list):
    assert len(sample_list) == 3
```

Here:

- `sample_list` is a **fixture**.
- Each test gets a **fresh copy** of `[1, 2, 3]`.
- No cleanup is needed → simple `return`.

◇ Example: Fixture with Teardown

```
@pytest.fixture
def db():
    db = {"users": []}  # setup
    yield db
    db["users"].clear()  # teardown

def test_add_user(db):
    db["users"].append("Alice")
    assert "Alice" in db["users"]
```

```
def test_is_empty(db):
    assert db["users"] == []
```

Here:

- Before each test → new dict `{"users": []}`.
- After each test → `db["users"].clear()` ensures no leftovers.

☑ Why are Fixtures Important?

- Keep tests **clean, DRY (Don't Repeat Yourself)**.
- Ensure tests are **independent** (no pollution between them).
- Handle **setup/teardown automatically**.
- Enable testing in **complex environments** (databases, APIs, files, Docker containers).

👉 In short: Fixtures are the **foundation of clean, reusable, and reliable tests** in `pytest`.

Would you like me to also compare **fixtures vs. setup/teardown methods (`unittest` style)** so you see why fixtures are more powerful?