pytest Python test runner

CONCEPTS

Files named test_*.py or *_test.py, functions named test_*, are collected; plain assert is all you need.

Collect find tests
↓ filter -k -m
Setup fixtures
↓ run
Call test body
↓ teardown
Report pass / fail

SELECT TESTS

pytest *<dir>* / *<file.py>*
Run a directory or file

pytest *<file.py>*::*<test>*
Run one test function

pytest -k "a and not b"
Select by name expression

pytest -m *<marker>*
Select by marker

RUN CONTROL

pytest -x
Stop at first failure

pytest --lf
Rerun only last failures

OUTPUT

pytest -v / -vv
More detail per test

pytest -s
Do not capture stdout

DEBUGGING

pytest --pdb
Open pdb at a failure

FIXTURES

@pytest.fixture
Declare a fixture

FIXTURES (cont.)

def test_x(*<fixture>***):**
Request a fixture by argument name

yield *<value>*
Setup before, teardown after

conftest.py
Directory-wide shared fixtures

Fixture with teardown

```
@pytest.fixture
def db():
    conn = connect()
    yield conn
    conn.close()
```

PARAMETRIZE AND MARKS

@pytest.mark.parametrize("a,b", [(1, 2), (3, 4)])
Run a test per data row

ASSERTIONS AND BUILT-INS

assert *<expr>*
Plain assert with rich diffs

with pytest.raises(*<Exc>***):**
Expect an exception

tmp_path
Temporary directory per test

monkeypatch
Patch attrs, env, dicts

CONFIG

[tool.pytest]
pyproject table (pytest 9+)

RECIPES

Table-driven test

```
@pytest.mark.parametrize("s,n", [
    ("1", 1),
    (" 2 ", 2),
    pytest.param("x", 0, id="bad"),
])
def test_parse(s, n):
    assert parse(s) == n
```