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

Fixtures in conftest.py are shared with every test below that directory, with no import.

SELECT TESTS

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

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

pytest *<file.py>*::*<Class>*::*<method>*
Run one method of a class

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

pytest -m *<marker>*
Select by marker

pytest -m "not slow"
Skip a marker

RUN CONTROL

pytest -x
Stop at first failure

pytest --lf
Rerun only last failures

pytest --ff
Failures first, then the rest

pytest --durations=*<n>*
Show n slowest tests

OUTPUT

pytest -v / -vv
More detail per test

pytest -s
Do not capture stdout

OUTPUT (cont.)

pytest --tb=short
Compact tracebacks

pytest -l
Show locals in tracebacks

pytest -ra
Summary of all but passed

DEBUGGING

pytest --pdb
Open pdb at a failure

breakpoint()
Drop into the debugger here

pytest --setup-show
Show fixture setup and teardown

pytest --fixtures
List available fixtures

FIXTURES

@pytest.fixture
Declare a fixture

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

yield *<value>*
Setup before, teardown after

@pytest.fixture(scope="*<s>***")**
Share across tests

@pytest.fixture(autouse=True)
Apply to every test in scope

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

@pytest.mark.skip
Always skip (reason="<why>")

@pytest.mark.skipif(*<cond>***)**
Skip on a condition (add reason=)

PARAMETRIZE AND MARKS (cont.)

@pytest.mark.xfail
Expected to fail (reason="<why>")

ASSERTIONS AND BUILT-INS

assert *<expr>*
Plain assert with rich diffs

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

pytest.raises(*<Exc>***, match="***<re>***")**
Check the message by regex

pytest.approx(*<x>***)**
Compare floats with tolerance

tmp_path
Temporary directory per test

monkeypatch
Patch attrs, env, dicts

capsys
Capture stdout and stderr

monkeypatch.setattr(*<obj>***, "***<name>***",** *<val>***)**
Replace an attribute

CONFIG

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

testpaths = ["tests"]
Where to look by default

addopts = ["-ra", "-q"]
Options added to every run

markers = ["slow: slow tests"]
Register custom markers

--strict-markers
Error on unregistered markers

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
```