Sure! Here's a **short and practical crash course on** pytest — enough to get you productive quickly.

## ✅ What is pytest?

pytest is a Python testing framework that's:

* Simple to write
* Powerful for real-world projects
* Widely used in open-source and production code

## 🛠️ 1. ****Installation****

bash

CopyEdit

pip install pytest

## 📄 2. ****Write a test****

Create a file like test\_math.py:

python

CopyEdit

def add(x, y):

return x + y

def test\_add():

assert add(2, 3) == 5

✅ Tests:

* Are functions that start with test\_
* Use assert — no need for unittest.TestCase

## ▶️ 3. ****Run the tests****

In the terminal:

bash

CopyEdit

pytest

It finds all files named test\_\*.py or \*\_test.py, and runs all test\_ functions.

## 🔄 4. ****Parametrize tests****

Test many inputs/outputs easily:

python

CopyEdit

import pytest

@pytest.mark.parametrize("x, y, expected", [

(1, 2, 3),

(3, 5, 8),

(10, -2, 8),

])

def test\_add(x, y, expected):

assert x + y == expected

## 🧪 5. ****Fixtures for setup****

python

CopyEdit

import pytest

@pytest.fixture

def sample\_list():

return [1, 2, 3]

def test\_sum(sample\_list):

assert sum(sample\_list) == 6

Fixtures are reusable setups for your tests.

## ⚠️ 6. ****Testing exceptions****

python

CopyEdit

def divide(x, y):

return x / y

def test\_zero\_division():

with pytest.raises(ZeroDivisionError):

divide(1, 0)

## 📁 7. ****Directory layout****

css

CopyEdit

my\_project/

├── src/

│ └── my\_code.py

└── tests/

└── test\_my\_code.py

Make sure to include \_\_init\_\_.py or use PYTHONPATH to resolve imports.

## 🔧 8. ****Useful commands****

bash

CopyEdit

pytest -v # Verbose output

pytest tests/test\_utils.py # Run specific file

pytest -k "add or sum" # Run tests matching keywords

pytest -x # Stop after first failure

## 🧰 9. ****Advanced tools****

* coverage → See what code your tests cover:

bash

CopyEdit

pip install pytest-cov

pytest --cov=src/

* tox / nox → Test across environments

## 🚀 10. ****When you're ready for more****

* pytest documentation
* Plugins: pytest-mock, pytest-django, pytest-xdist, etc.

tell me about plugin pytest-mock

**ChatGPT said:**

The pytest-mock plugin is a handy **pytest wrapper around unittest.mock**, making mocking easier and cleaner in tests using pytest.

**✅ What it is:**

* pytest-mock provides a **mocker fixture**, which simplifies mocking and patching.
* It's built on top of unittest.mock and integrates **natively with pytest**.

**🔧 Installation:**

bash

CopyEdit

pip install pytest-mock

**💡 Why use pytest-mock?**

* Avoids manual @patch or with patch(...) syntax.
* Automatically handles cleanup after the test.
* Makes tests **more readable** and **shorter**.

**🧪 Basic Example:**

Suppose you have this function:

python

CopyEdit

# file: mymath.py

import math

def compute\_sqrt(x):

return math.sqrt(x)

Here’s a test using pytest-mock:

python

CopyEdit

# file: test\_mymath.py

from mymath import compute\_sqrt

def test\_compute\_sqrt(mocker):

mock\_sqrt = mocker.patch('math.sqrt', return\_value=10)

result = compute\_sqrt(25)

assert result == 10

mock\_sqrt.assert\_called\_once\_with(25)

**✨ Features of mocker:**

| **Feature** | **Example** |
| --- | --- |
| Mock a function | mocker.patch('module.function') |
| Spy on a function (wraps it) | mocker.spy(obj, 'method') |
| Create a mock object | m = mocker.Mock() |
| Reset call history | mock.reset\_mock() |
| Assert call counts | mock.call\_count == 1 |

**🧼 Cleanup:**

pytest-mock ensures all patches are **automatically undone after each test**, avoiding side effects.

**🧰 Optional config in pytest.ini:**

ini

CopyEdit

[pytest]

addopts = --strict-markers

**🧠 Summary:**

* pytest-mock = easier mocking with pytest.
* Use mocker.patch(...) instead of @patch(...).
* Automatically cleans up.
* Perfect for clean, modular, and maintainable unit tests.

Would you like examples with mocker.spy() or mocking class methods?

------------------------------------------------------------------------------------------------------------------

pytest-xdist: Parallel and Distributed Testing for pytest

----------------------------------------------------------------------------