📚 Pytest Library – Testing in Python

📌 Introduction

`pytest` is a **powerful testing library** in Python used to write **unit tests**, **functional tests**, and even **complex test scenarios**. It helps developers **find bugs** early and ensures that the code behaves correctly.

✅ Why Use Pytest?

- **Automatic test discovery** – Finds all test files and functions without needing extra configuration.
- **Simple syntax** – Easy to write and understand.
- **Clear output** – Shows you exactly where and why a test failed.
- **Scalable** – Works for both small scripts and large applications.

🛠️ Installation

Install pytest using pip:

```
pip install pytest
```

📂 File & Function Naming Rules

To use pytest properly:

- Test file name should start with: `test_`
- Test function name should start with: `test_`

Example:
File → `test_calculator.py`
Function → `test_add()`

✍ Example Code

1. Your main code file: `calculator.py`

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

---

2. Your test file: `test_calculator.py`

```python
from calculator import add, subtract

def test_add():
    assert add(2, 3) == 5
    assert add(-1, 1) == 0

def test_subtract():
    assert subtract(5, 3) == 2
    assert subtract(10, 10) == 0
```

```
assert subtract(10, 10) == 0
```

▶ Running the Tests

Open your terminal in the project folder and type:

```
pytest
```

Pytest will:

- Find all files starting with `test_`
- Run all functions starting with `test_`
- Show which tests passed or failed

✹ If a Test Fails

Pytest gives a **clear error** showing:

- Which test failed
- What was expected
- What was actually returned

This helps you fix the issue quickly.

□ Benefits of Using Pytest

Feature	Benefit
Fast & automatic	No need to write test runners
Easy debugging	Clear and readable error messages
Modular structure	Split tests in multiple files
Plugins available	For advanced features

□ Real-World Usage

- Used in **professional Python development**
- Helps in **Test Driven Development (TDD)**
- Commonly used in frameworks like **Django**, **Flask**, and **FastAPI**

✓ Conclusion

`pytest` makes Python testing:

- **Easy to start**
- **Powerful to grow**
- **Reliable for production**

If you're serious about writing **bug-free** and **efficient** code, learning pytest is a must!