



pytest



## Простота написання тестів

Для написання тестів достатньо створити функції, імена яких починаються з `test_`.

**`pytest` автоматично знаходить тестові функції, класи або модулі за певними правилами:**

- Імена файлів мають починатися або закінчуватися на `test` (наприклад, `test_sample.py`).
- Імена тестових функцій мають починатися з `test_`.

**`assert`**



**Викликаються на глобальному рівні, перед і після виконання всього тестового модуля.**

```
def setup():  
    print ("basic setup into module")  
def teardown():  
    print ("basic teardown into module")
```

**Викликаються один раз на модуль.**

```
def setup_module(module):  
    print ("module setup")  
def teardown_module(module):  
    print ("module teardown")
```

**Викликаються для кожної тестової функції в модулі.**

```
def setup_function(function):  
    print ("function setup")  
def teardown_function(function):  
    print ("function teardown")
```

## Декоратор @pytest.fixture

```
@pytest.fixture
def sample_data():
    return {"name": "John", "age": 30}


def test_sample_data(sample_data):
    assert sample_data["name"] == "John"
    assert sample_data["age"] == 30
```

```
@pytest.fixture
def setup_and_teardown():
    # Підготовка
    print("Setup")
    resource = {"connection": "db_connection"}
    yield resource
    # Очищення
    print("Teardown")
```



# conftest.py

```
# conftest.py
@pytest.fixture
def shared_data():
    return {"shared": True}
```



У pytest "scope" визначає, як довго існує фікстура, і коли вона створюється та зникає.

- function (за замовчуванням): фікстура створюється для кожної тестової функції.
- **@pytest.fixture(scope="class")**: фікстура створюється один раз для кожного тестового класу.
- **@pytest.fixture(scope="module")**: фікстура створюється один раз для кожного модуля (файлу з тестами).
- **@pytest.fixture(scope="package")**: фікстура створюється для всього пакету тестів.
- **@pytest.fixture(scope="session")**: фікстура створюється один раз для всієї тестової сесії.

## Коли використовувати різні score?

### **function:**

Коли фікстура має бути унікальною для кожного тесту.  
Наприклад, створення тимчасових файлів або даних.

### **class:**

Коли потрібно повторно використовувати ресурси для тестів одного класу.  
Наприклад, ініціалізація об'єкта, який використовується кількома тестами в класі.

### **module:**

Коли потрібно зберегти ресурс у межах одного модуля.  
Наприклад, відкриття з'єднання з базою даних.

### **session:**

Коли фікстура створює глобальний ресурс, який використовується всіма тестами.  
Наприклад, підключення до API, ініціалізація конфігурації або створення тимчасового каталогу.



```
@pytest.mark.parametrize(arguments, values)
```



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

```
(1, 2, 3), # Перший набір даних
```

```
(3, 5, 8), # Другий набір даних
```

```
(-1, -1, -2) # Третій набір даних
```

```
])
```

```
def test_addition(x, y, expected):
```

```
    assert x + y == expected
```







**pytest.raises()**

```
@pytest.mark.parametrize("a, b, exception", [  
    (1, 0, ZeroDivisionError),  
    ("1", 2, TypeError)  
])  
def test_exceptions(a, b, exception):  
    with pytest.raises(exception):  
        result = a / b
```

**@pytest.mark.skip**

**@pytest.mark.skip(reason="Test is not available")**

**@pytest.mark.skipif**

**@pytest.mark.skipif(condition, reason="Test is not available")**

**@pytest.mark.xfail**

**Очікувано буде fail**



**@pytest.mark**

**@pytest.mark.smoke**

**@pytest.mark.regression**

**pytest -m smoke**

## Запуск

**pytest test\_file.py**

**pytest test\_file.py::test\_function\_name**

**pytest -v** або **--verbose** вмикає детальний вивід для кожного тесту

**pytest -q** або **--quiet** зменшує обсяг виводу

**pytest -k "pattern"** запускає тільки тести, що відповідають певному шаблону в назві.

**pytest -m marker** запускає тільки тести з вказаною міткою

**pytest -n** КІЛЬКІСТЬ ПОТОКІВ

