LAB - Hướng dẫn sử dụng thư viện Pytest

I. Cài đặt thư viện Pytest

Vào cmd gõ: py – m pip install pytest

1. Chạy các file test và các phương thức test

Theo mặc định thì pytest sẽ chỉ nhận dạng được các file có tên bắt đầu với test_ hoặc kết thúc với _test. Còn đối với các phương thức thì yêu cầu tên của các phương thức bắt đầu bằng "test", những phương thức có tên khác sẽ bị bỏ qua.

Dưới đây là ví dụ về đặt tên file pytest hợp lệ và không hợp lệ:

test_username.py --- hợp lệ

username_test.py --- hợp lệ

testusername --- không hợp lệ

usernametest --- không hợp lệ

Và ví dụ về tên các phương thức pytest hợp lệ và không hợp lệ:

def test_method(): --- hợp lệ

def testmethod(): --- hợp lệ

def method(): ---không hợp lệ

2. Chạy những file test được chỉ định

Chúng ta sẽ sử dụng pytest markers bằng cách định nghĩa các marker trên mỗi phương thức, cách khai báo @pytest.mark.<name>

Ví dụ tạo 2 file test test_1.py và test_2.py có nội dung như bên dưới, sau đó chỉ chạy những test có mark tên example1.

Nội dung file test_1.py:

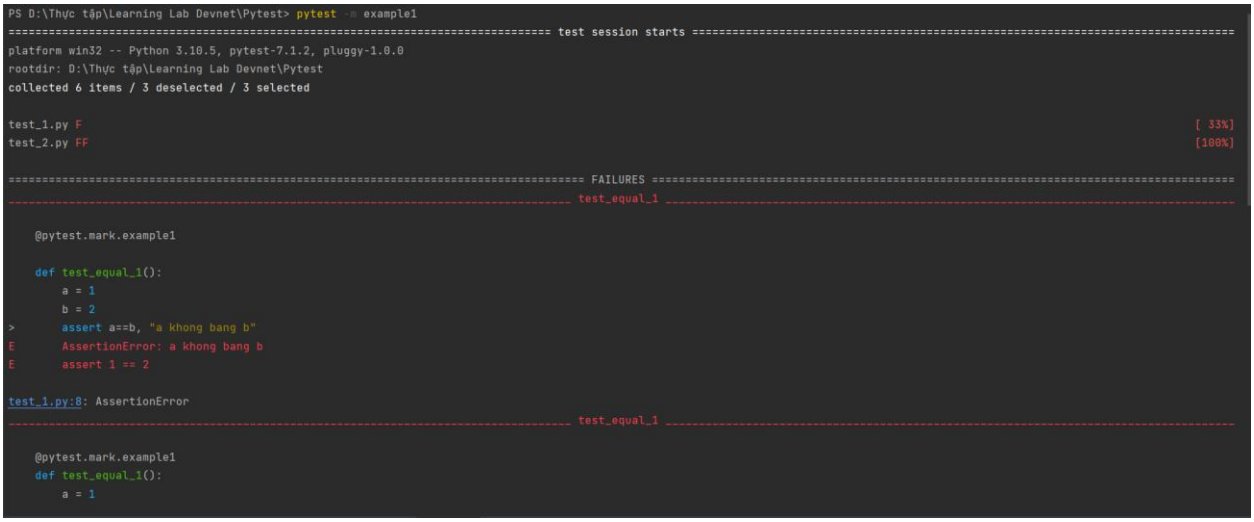
```
import pytest


@pytest.mark.example1
def test_equal_1():
    a = 1
    b = 2
    assert a==b, "a không bằng b"
@pytest.mark.example2
def test_equal_2():
    a = 1
    b = 2
    assert a==b, "a không bằng b"
```

Nội dung file test_2.py:

```
import pytest


@pytest.mark.example1
def test_equal_1():
    a = 1
    b = 2
    assert a==b, "a không bằng b"
```

```
@pytest.mark.example1

def test_equal_2():

    a = 1

    b = 2

    assert a==b, "a không bằng b"
```

Sau khi lưu lại 2 file trên, chúng ta sẽ chạy những test có mark tên example1 bằng cách vào đường dẫn nơi đặt 2 file vừa tạo và gõ: **pytest –m example1**



```
test_2.py:13: AssertionError
  D:\Thực tập\Learning Lab Devnet\Pytest\test_2.py:9: PytestUnknownMarkWarning: Unknown pytest.mark.example1 - is this a typo?  You can register custom marks to avoid this warning - fo
r details, see https://docs.pytest.org/en/stable/how-to/mark.html
    @pytest.mark.example1

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
=========================================================================== short test summary info ============================================================================
FAILED test_1.py::test_equal_1 - AssertionError: a khong bang b
FAILED test_2.py::test_equal_1 - AssertionError: a khong bang b
FAILED test_2.py::test_equal_2 - AssertionError: a khong bang b
======================================================================== 3 failed, 3 deselected, 4 warnings in 0.27s ========================================================================
```

Theo như trên hình thì test_1.py có 1 chức năng bị Fail, test_2 có 2 phương thức bị Fail, chỉ ra bị sai ở chỗ nào của phương thức.

Ngoài ra, pytest có cung cấp các mark được định nghĩa sẵn, chúng ta sẽ vào cmd gõ: **pytest - -markers**

```
PS D:\Thực tập\Learning Lab Devnet\Pytest> pytest --markers
@pytest.mark.filterwarnings(warning): add a warning filter to the given test. see https://docs.pytest.org/en/stable/how-to/capture-warnings.html#pytest-mark-filterwarnings

@pytest.mark.skip(reason=None): skip the given test function with an optional reason. Example: skip(reason="no way of currently testing this") skips the test.

@pytest.mark.skipif(condition, ..., *, reason=...): skip the given test function if any of the conditions evaluate to True. Example: skipif(sys.platform == 'win32') skips the test if w
e are on the win32 platform. See https://docs.pytest.org/en/stable/reference/reference.html#pytest-mark-skipif

@pytest.mark.xfail(condition, ..., *, reason=..., run=True, raises=None, strict=xfail_strict): mark the test function as an expected failure if any of the conditions evaluate to True.
Optionally specify a reason for better reporting and run=False if you don't even want to execute the test function. If only specific exception(s) are expected, you can list them in rai
ses, and if the test fails in other ways, it will be reported as a true failure. See https://docs.pytest.org/en/stable/reference/reference.html#pytest-mark-xfail

@pytest.mark.parametrize(argnames, argvalues): call a test function multiple times passing in different arguments in turn. argvalues generally needs to be a list of values if argnames
specifies only one name or a list of tuples of values if argnames specifies multiple names. Example: @parametrize('arg1', [1,2]) would lead to two calls of the decorated test function,
 one with arg1=1 and another with arg1=2.see https://docs.pytest.org/en/stable/how-to/parametrize.html for more info and examples.

@pytest.mark.usefixtures(fixturename1, fixturename2, ...): mark tests as needing all of the specified fixtures. see https://docs.pytest.org/en/stable/explanation/fixtures.html#usefixtu
res

@pytest.mark.tryfirst: mark a hook implementation function such that the plugin machinery will try to call it first/as early as possible.

@pytest.mark.trylast: mark a hook implementation function such that the plugin machinery will try to call it last/as late as possible.
```

II. Fixtures

Fixtures được dùng để khởi tạo các thông số đầu vào, thay vì trong mỗi đoạn test chúng ta đều phải khai báo các giá trị để đưa vào test thì ta chỉ cần khai báo một lần duy nhất, khi muốn sử dụng vào đoạn test nào thì gọi hàm đã được chúng ta đánh dấu là fixture.

Chúng ta sẽ đi đến ví dụ sau để dễ hình dung hơn:

```
import pytest


@pytest.fixture
```

```
def input_value():

    input = 10

    return input


def test_mod_2(input_value):

   assert input_value %2 == 0


def test_mod_3(input_value):

   assert input_value %3 == 0
```

Lưu file với tên test_mod.py, sau đó vào cmd chạy file: pytest –k mod –v .Đây là một cách chạy các test có tên có chữ “mod”.

Kết quả: