reference: <https://docs.pytest.org/en/latest/fixture.html>

tutorial video: <https://www.youtube.com/watch?v=8mp_1Jt-xHQ>

[what is a fixture]

Imagine…

1. all your testing function have the same repetitive code, and it is tedious to code them manually
2. all your testing function need the same object to be created, and it is time waste to create them for each test function

Both cases above can be solved by using pytest fixture, which is the function that test\_function take as argument. The fixture function will be executed before the test\_function() does, so it can be used as…

1. the test\_function preparation
2. the object generating function for repeating test process

* example

@pytest.fixture # the fixture decorator to annotate fixture function.

def some\_fixture()

return [1,0] # can return a object

def test\_f1(some\_fixture) # the fixture function should be test function argument

assert some\_fixture[0] == 1 # the argument will be the object that fixture function return

[check available fixture]

>> pytest --fixtures <test\_XXX>.py

[conftest.py]

reference: <https://stackoverflow.com/questions/34466027/in-pytest-what-is-the-use-of-conftest-py-files>

Fixture can be defined in the test file or, more generally, defined in conftest.py file.

In this way, fixture function can be shared

the test file don’t need to import module from conftest.py.

* conftest example:

test\_XXX.py

def test\_OOO(some\_fixture):

…..

conftest.py

import pytest

@pytest.fixture

def some\_fixture():

print(“some\_fixture running”)

the system will automatically import conftest fixture

* how the conftest file cover the file tree, when there is multiple confest.py
* the conftest file will cover the same directory and its all sub-directory
* the sub-direcotory can also have conftest file that adding or overwriting fixtures.
* the test file will take fixture that whichever is closest to it. (upward search from same directory)

root/conftest.py

root/test\_1.py

root/sub/conftest.py

root/sub/sub2/conftest.py

root/sub/sub2/test\_2.py

[[fixture scope]](https://docs.pytest.org/en/latest/fixture.html#scope-sharing-a-fixture-instance-across-tests-in-a-class-module-or-session)

the fixture scope is the setting saying what range can this fixture object be shared.

For example, if the fixture scope is “module”, it means that fixture object can be shared in a same test module. So no matter how many functions using that fixture in the same module, it will be only instantiated once.

The all scope setting : function(default), class, module, package, session

function 🡺 instantiate every time

module 🡺 instantiate once for each test file

session 🡺 instantiate once for each “pytest” execution

* Notice:

Because the fixture may be shared in mulitple test function, the change on fixture in one test function can affect the other test function.

[[fixture finalization]](https://docs.pytest.org/en/latest/fixture.html#fixture-finalization-executing-teardown-code)

Sometime when a fixture is no longer needed, we may want to do something about it.(Like object destructor) How can we do that?

* example

@pytest.fixture

def some\_fixture():

<do fixture thing>

yield <object> # here use “yeild” instead of “return”

# the system will executed following part when this fixture object is no longer needed

<do fixture closing thing>