Фикстура

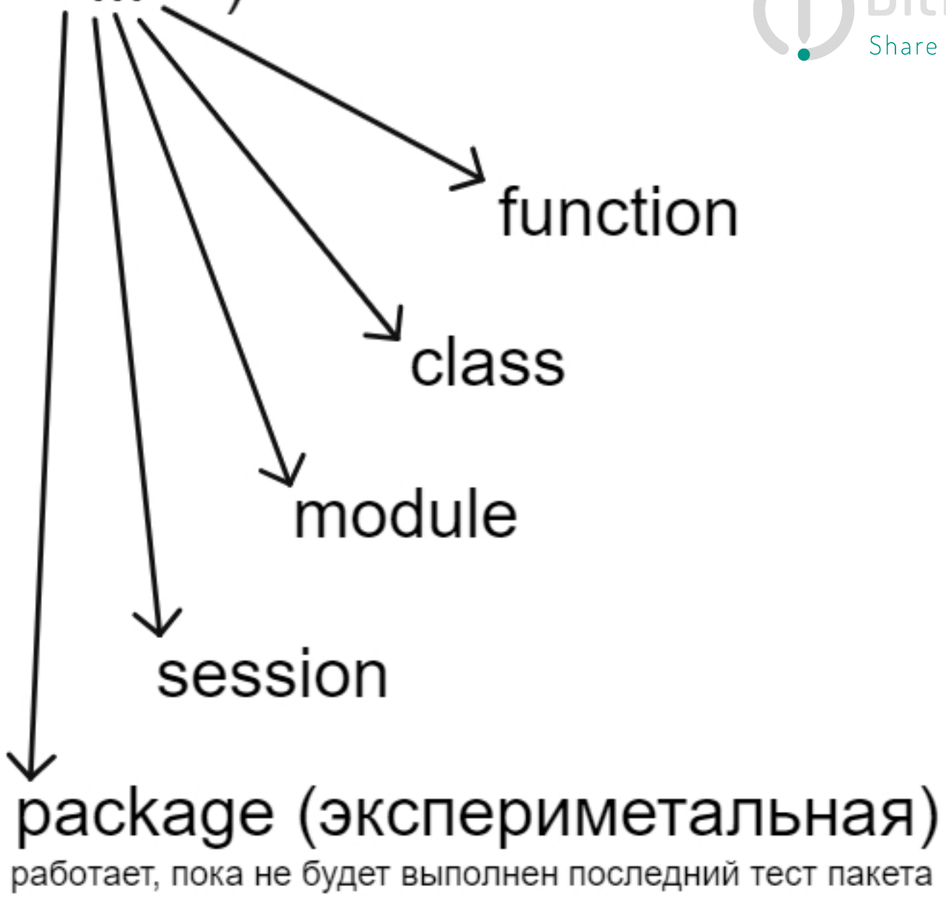
Фикстура — это ресурс или объект, который можно рассматривать как набор условий или предопределенное состояние, необходимое тесту для правильного выполнения, зачастую фикстуры создаются, чтобы генерировать какие-то данные еще до теста и возвращать их для использования в тесте или перед тестом,

Фикстуры позволяют тестовым функциям легко получать предварительно инициализированные объекты и работать с ними, не заботясь об импорте/установке/очистке.

Например:
1. Создавать подключение к базе данных перед тестом и отключаться от нее после завершения теста
2. Инициализировать драйвер-браузера и закрывать сессию после завершения теста
3. Авторизовываться перед запуском теста и не тратить время на логин
4. Создавать новый аккаунт перед тестом, использовать его в тесте и по завершению теста удалять его т.д

Поиск фикстур начинается с тестовых классов, затем они иущется в тестовых модулях и в файлах conftest.py, и, в последнюю очередь, во встроенных и сторонних плагинах.

@pytest.fixture(scope= ...)




```
@pytest.fixture(scope="function")


def browser():
        print("\nstart browser for test..")
        browser = webdriver.Chrome()

        yield browser

        print("\nquit browser..")
        browser.quit()
```

При запросе фикстуры функцией сначала инициализируются фикстуры с самой широкой областью действия - session и module, а затем - фикстуры более низкого уровня с областями class или function.

В рамках одной тестовой функции порядок создания фикстур с одинаковой областью действия зависит от очередности вызова этих фикстур и установленных между ними зависимостей.

При этом фикстуры с параметром autouse = True инициализируются прежде явно объявленных фикстур того же уровня.

Если вы используете оператор yield вместо return, то весь код после yield выполняет роль «уборщика»

conftest.py

Для хранения часто употребимых фикстур
и хранения глобальных настроек
нужно использовать файл conftest.py,
который должен лежать в директории верхнего уровня в вашем проекте с тестами.
Можно создавать дополнительные файлы conftest.py в других директориях,
но тогда настройки в этих файлах будут применяться только к тестам в под-директориях.

PyTest автоматически находит и подгружает файлы conftest.py,
которые находятся в директории с тестами.

pytest.ini


```
[pytest]
markers =
        smoke: marker for smoke tests
        regression: marker for regression tests
```

```
@pytest.mark.smoke
def test_.....
```

```
@pytest.mark.regression
def test_.....
```

```
pytest -s -v -m smoke test_fixture8.py

pytest -s -v -m "not smoke" test_fixture8.py

pytest -s -v -m "smoke or regression" test_fixture8.py
```

@pytest.mark.parametrize

BitPaper
Share Knowledge

```
@pytest.mark.parametrize('language', ["ru", "en-gb"])
def test_guest_should_see_login_link(browser, language):
    link = f"http://selenium1py.pythonanywhere.com/{language}/"
```

yTest позволяет запустить один и тот же тест с разными входными параметрами.
ля этого используется декоратор @pytest.mark.parametrize()

@pytest.mark.parametrize() нужно передать
араметр, который должен изменяться, и список значений параметра.

самом тесте наш параметр тоже нужно передавать в качестве аргумента.

PyTest будет запускать тест
с разными параметрами,
указанными в списке.

Количество параметров
- это количество запусков те

Обратите внимание, что внутри декоратора имя параметра оборачивается в кавычки,
в списке аргументов теста кавычки не нужны.

conftest.py (addoption)


```
import pytest
from selenium import webdriver

def pytest_addoption(parser):
    parser.addoption('--browser_name', action='store', default=None,
                     help="Choose browser: chrome or firefox")


@pytest.fixture(scope="function")
def browser(request):
    browser_name = request.config.getoption("browser_name")
    if browser_name == "chrome":
        browser = webdriver.Chrome()
    elif browser_name == "firefox":
        browser = webdriver.Firefox()
    else:
        raise pytest.UsageError("--browser_name should be chrome or firefox")

    yield browser
```

```
pytest -s -v --browser_name=chrome test_parser.py
pytest -s -v --browser_name=firefox   test_parser.py
```

Локаторы

1.By.id
2.By.name
3.By.className
4.By.TagName
5.By.LinkText
6.By.PartialLinkText
7.By.cssSelector
8.By.XPath

1.link
2.id
3.name
4.dom
5.css
6.xpath