今日课题：Pytest从入门到精通训练营第二次课

技术分享：码尚学院-百里老师　　录播视频加助教微信：mashang-vv

Pytest框架实现一些前后置（固件，夹具）的处理，常用三种。

一、setup/teardown,setup_class/teardown_class所有

为什么需要这些功能？

比如：web自动化执行用例之前，请问需要打开浏览器吗？用例执行后需要关闭浏览器？

```
class TestMashang:

 #这个在所有的用例之前只执行一次
 def setup_class(self):
 print('在每个类执行前的初始化的工作：比如：创建日志对象，创建数据库的连接，创建接口的请求对象。')

 #在每个用例之前执行一次。
 def setup(self):
 print('\n在执行测试用例之前初始化的代码：打开浏览器，加载网页')

  def test_01_baili(self):
  print('\n测试百里')

  def test_02_xingyao(self):
  print('\n测试星瑶')

  def teardown(self):
  print('\n在执行测试用例之后的扫尾的代码：关闭浏览器')

  def teardown_class(self):
  print('在每个类执行后的扫尾的工作：比如：销毁日志对象，销毁数据库的连接，销毁接口的请求对象。')
```

注意：和Unittest不一样，全是小写。

二、使用@pytest.fixture()装饰器来实现部分用例的前后置。

装饰器

@pytest.fixture(scope="",params="",autouse="",ids="",name="")

---

(1)scope表示的是被@pytest.fixture标记的方法的作用域。function(默认)，class，module，package/session.

(2)params：参数化（支持，列表[]，元祖()，字典列表[{},{},{}]，字典元祖({},{},{})

```
import pytest

@pytest.fixture(scope='function',params=['成龙','甄子丹','菜10'])
def my_fixture(request):
 print('前置')
 yield
 print('后置')
 return request.param

class TestMashang1:

 def test_01_baili(self):
 print('\n测试百里')

 def test_02_xingyao(self,my_fixture):
 print('\n测试星瑶')
 print('--------------'+str(my_fixture))
```

params=['成龙','甄子丹','菜10'] 这里params是参数名，有s。

request.param这里是属性名，是没有s的。

(3)autouse=True：自动使用，默认False

(4)ids：当使用params参数化时，给每一个值设置一个变量名。意义不大。

(5)name：给表示的是被@pytest.fixture标记的方法取一个别名。

当取了别名之后，那么原来的名称就用不了了。

三、通过conftest.py和@pytest.fixture()结合使用实现全局的前置应用（比如：项目的全局登录，模块的全局处理）

1.conftest.py文件是单独存放的一个夹具配置文件，名称是不能更改。

2.用处可以在不同的py文件中使用同一个fixture函数。

3.原则上conftest.py需要和运行的用例放到统一层。并且不需要做任何的imprt导入的操作。

总结：
setup/teardown，setup_class/teardown_class 它是作用于所有用例或者所有的类
@pytest.fixtrue() 它的作用是既可以部分也可以全部前后置。
conftest.py和@pytest.fixtrue()结合使用，作用于全局的前后置。

四、断言

assert

```
assert 1==2
```

五、pytest结合allure-pytest插件生成allure测试报告

昨天：pytest-html
今天：allure-pytest

1.下载，解压，配置path路径。
https://github.com/allure-framework/allure2/releases

path路径配置：E:\allure-2.13.7\bin

验证：allure --version

问题：dos可以验证但是pycharm验证失败，怎么办，重启pycharm.

2.加入命令生成json格式的临时报告。
--alluredir ./temp

3.生成allure报告
os.system('allure generate ./temp -o ./report --clean')

allure generate	命令，固定的
./temp	临时的json格式报告的路径
-o	输出output
./report	生成的allure报告的路径
--clean	清空./report 路径原来的报告

六、完成作业