一、Unittest

Unittest是Python标准库中自带的单元测试框架，Unittest有时候也被称为PyUnit，就像JUnit是Java语言的标准单元测试框架一样，Unittest则是Python语言的标准单元测试框架。
Unittest支持自动化测试，测试用例的初始化、关闭和测试用例的聚合等功能，它有一个很重要的特性：它是通过类(class)的方式，将测试用例组织在一起。
示例：

```
import unittest
import logging
from src.coupon.list_consumed_coupon import ListConsumedCoupon


class CouponInterface(unittest.TestCase):
    ''' 优惠券接口 '''

    def test_list_consumed_coupon_no_login(self):
        ''' 我的优惠券，未登录，返回码校验 '''
        logging.info("test_list_consumed_coupon")
        lc = ListConsumedCoupon()
        response = lc.list_consumed_coupon_param_no_cookie()
        # 校验接口返回码是否为0
        self.assertEquals(response["code"], 401, msg="接口返回码错误，code = " + str(response["code"]))
```

执行结果：

```
run_coupon.py .                                                          [100%]

========================== 1 passed in 1.78 seconds ==========================
Process finished with exit code 0
```

注：unittest有一个关联模块unittest2，但unittest2仅适用于Python 2.4-2.6。这是由于从Python 2.7开始，unittest增加一些新的特性。为了在老的版本中支持这些特性，所以提供了unittest2这个库。但对于Python 2.7及之后的版本，unittest是唯一的。本次示例中使用的为python2.7。

二、Pytest

Pytest是Python的另一个第三方单元测试库。它的目的是让单元测试变得更容易，并且也能扩展到支持应用层面复杂的功能测试。
pytest的特性有：

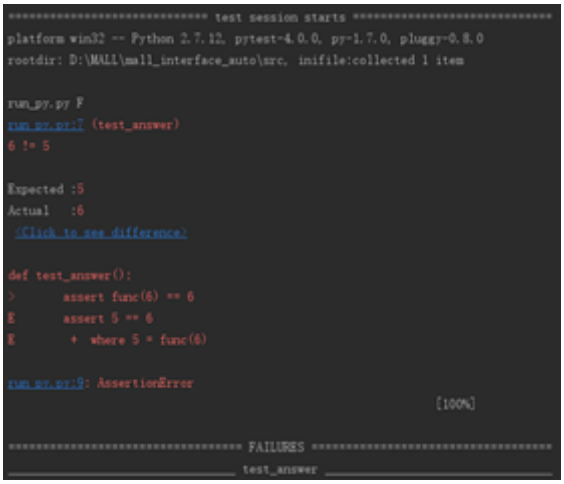
- 支持用简单的assert语句实现丰富的断言，无需复杂的self.assert*函数
- 自动识别测试模块和测试函数
- 模块化夹具用以管理各类测试资源
- 对 unittest 完全兼容，对 nose基本兼容
- 支持Python3和PyPy3
- 丰富的插件生态，已有300多个各式各样的插件，社区繁荣

示例：

```python
def func(x):
    return x - 1

def test_answer():
    assert func(6) == 6
```

执行结果：



三、Unittest vs Pytest

	unittest	pytest
用例编写规则	1)测试文件必须先import unittest 2)测试类必须继承unittest.TestCase 3)测试方法必须以“test_”开头 4)测试类必须要有unittest.main()方法	1)测试文件名必须以“test_”开头或者"_test"结尾（如：test_ab.py） 2)测试方法必须以“test_”开头 3)测试类命名以"Test"开头
用例分类执行	默认执行全部用例，也可以通过加载testsuit，执行部分用例	可以通过@pytest.mark来标记类和方法，pytest.main加入参数("-m")可以只运行标记的类和方法
用例前置和后置	提供了setUp/tearDown，只能针对所有用例	pytest中的fixture显然更加灵活。可以任意自定义方法函数，只要加上@pytest.fixture()这个装饰器，那么被装饰的方法就可以被使用
参数化	需依赖ddt库	使用@pytest.mark.parametrize装饰器
断言	很多断言格式(assertEqual、assertIn、assertTrue、assertFalse)	只有assert一个表达式，用起来比较方便
报告	使用HTMLTestRunnerNew库	有pytest-HTML、allure插件
失败重跑	无此功能	pytest支持用例执行失败重跑，pytest-rerunfailures插件

总结：总体来说，unittest用例格式复杂，兼容性无，插件少，二次开发方便。pytest更加方便快捷，用例格式简单，可以执行unittest风格的测试用例，无须修改unittest用例的任何代码，有较好的兼容性。pytest插件丰富，比如flask插件，可用于用例出错重跑，还有xdist插件，可用于设备并行执行，效率更高。