# Unittest

## 什么是 UnitTest 框架？

概念：UnitTest 是 Python 自带的一个单元测试框架，用它来做单元测试

## 为什么使用 UnitTest 框架？

能够组织多个用例去执行；

提供丰富的断言方法；

能够生成测试报告；

## UnitTest 核心要素

TestCase；

TestSuite；

TestRunner；

TestLoader；

Fixture；

## 一、TestCase

1 、 导 包 ：import unittest

2、定义测试类：新建测试类必须继承 unittest.TestCase；

3、定义测试方法：测试方法名称命名必须以 test 开头

import unittest  
  
class my\_test*(*unittest.TestCase*)*:  
  
 def test\_011*(*self*)*:  
 print*(*'test 01'*)* def test\_022*(*self*)*:  
 print*(*'tset 02'*)* def test\_003*(*self*)*:  
 print*(*'test 003'*)*

## 二、TestSuite

多条测试用例集合在一起，就是一个 TestSuite

### Makesuite 把类添加到套件中

**#只是把测试用例添加到了测试套件中，并不能执行用例**

把整个测试类里面的方法添加到测试套件中

import unittest  
import test\_01 #导入test\_01模块，  
#只是把测试用例添加到了测试套件中，并不能执行用例  
suite=unittest.TestSuite*()*#创建测试套件对象  
suite.addTest*(*test\_01.my\_test*(*'test\_11111'*))*#测试对象.addtest.模块.类（’方法名称‘）  
suite.addTest*(*test\_01.my\_test*(*'test\_22222'*))*#把整个类里面的方法添加到测试套件中  
suite.addTest*(*unittest.makeSuite*(*test\_01.my\_test*))*

## 三、TextTestRunner

用来执行测试套件中的用例

import unittest  
import test\_01 #导入test\_01模块，  
#只是把测试用例添加到了测试套件中，并不能执行用例  
suite=unittest.TestSuite*()*#创建测试套件对象  
suite.addTest*(*unittest.makeSuite*(*test\_01.my\_test*))*#把整个类里面的方法添加到测试套件中  
runner=unittest.TextTestRunner*()*#创建testrunner对象  
runner.run*(*suite*)*#把测试套件放到 runner里面，便可执行 套件中的用例

Ran 3 tests in 0.000s

OK

test 111111

tset22222

test 33333

问题：使用 suite.addtest(unittest.makeSuite(className))导入多条测试用例，

addtest()需要调用多次。

## 四、TestLoader

#把当前路径下 所有以my开头的模块内、 以test开头的测试用例装进测试套件中

import unittest  
suite=unittest.TestLoader*()*.discover*(*'.',"my\*.py"*)*#把当前路径下 所有以my开头的模块内、 以test开头的测试用例装进测试套件中，  
runner=unittest.TextTestRunner*()*#创建testrunner对象  
runner.run*(*suite*)*#把测试套件放到 runner里面，便可执行 套件中的用例

Ran 2 tests in 0.000s

OK

tset22222

test 33333

六、TestSuite 与 TestLoader 区别

TestSuite 需要手动添加测试用例（可以添加测试类，也可以添加测试类中某个测试方法）；

TestLoader 搜索指定目录下指定开头.py 文件，并添加测试类中的所有的测试方法，不能指定添加测试方法

## 小结

- 所有的TestCase最终都是用TextTestRunner来执行的

- TextTestRunner执行的是TestSuite

- 一个TestSuite中可以有多个TestCase

框架

步骤

1. 导入unittest
2. 导入参数化包 from parmeterzed import parmeterzed
3. 创建 类 继承 unittest. TestCase类

## Fixture

* 可以在测试用例执行执行之前自动调用指定的函数,在测试用例执行之后自动调用指定的函数
* 控制级别
  + 方法级
    - 每个方法执行前和执行后都自动调用函数
  + 类级
    - 不管类中有多少方法,一个类执行前后都自动调用函数
  + 模块级
    - 不管一个模块(一个模块就是一个py文件)中有多少类,模块执行前后自动调用函数

### 方法级别 setUp，tearDown

import unittest  
  
class my\_test*(*unittest.TestCase*)*:  
  
 def setUp*(*self*)*:  
 print*(*'方法运行前调用'*)* def aa11111*(*self*)*:  
 print*(*'test 111111'*)* def test\_22222*(*self*)*:  
 print*(*'tset22222'*)* def test\_333333*(*self*)*:  
 print*(*'test 33333'*)* def tearDown*(*self*)*:  
 print*(*'方法结束后自动调用'*)*

### 类级别 setUpClass，tearDownClass

类级别方法需要加classmethod

import unittest  
  
class my\_test*(*unittest.TestCase*)*:  
  
 @classmethod  
 def setUpClass*(*cls*)*:  
 print*(*' setUpClass 每个类运行一次'*)* @classmethod  
 def tearDownClass*(*cls*)*:  
 print*(*'tearDownClass 类运行结束后运行一次'*)* def setUp*(*self*)*:  
 print*(*'方法运行前调用'*)* def aa11111*(*self*)*:  
 print*(*'test 111111'*)* def test\_22222*(*self*)*:  
 print*(*'tset22222'*)* def test\_333333*(*self*)*:  
 print*(*'test 33333'*)* def tearDown*(*self*)*:  
 print*(*'方法结束后自动调用'*)*

## 参数化

参数化的三种等效方式

1. 参数直接写进实参里
2. 通过变量 直接写变量名
3. 通过 return 返回 当参数

import unittest  
from parameterized import parameterized  
class my\_test*(*unittest.TestCase*)*:  
  
 @parameterized.expand*([(*1,2,7*)*,*(*5,2,7*)*,*(*4,4,9*)])* def test11111*(*self,a,b,c*)*:  
 # print('test 111111')  
 d=a+b  
 self.assertEqual*(*c,d*)*

## 跳过方法skip

@unittest.skip  
def testa*(*self*)*:  
 print*(*'跳过此方法'*)*

FAILED (failures=2, skipped=1)

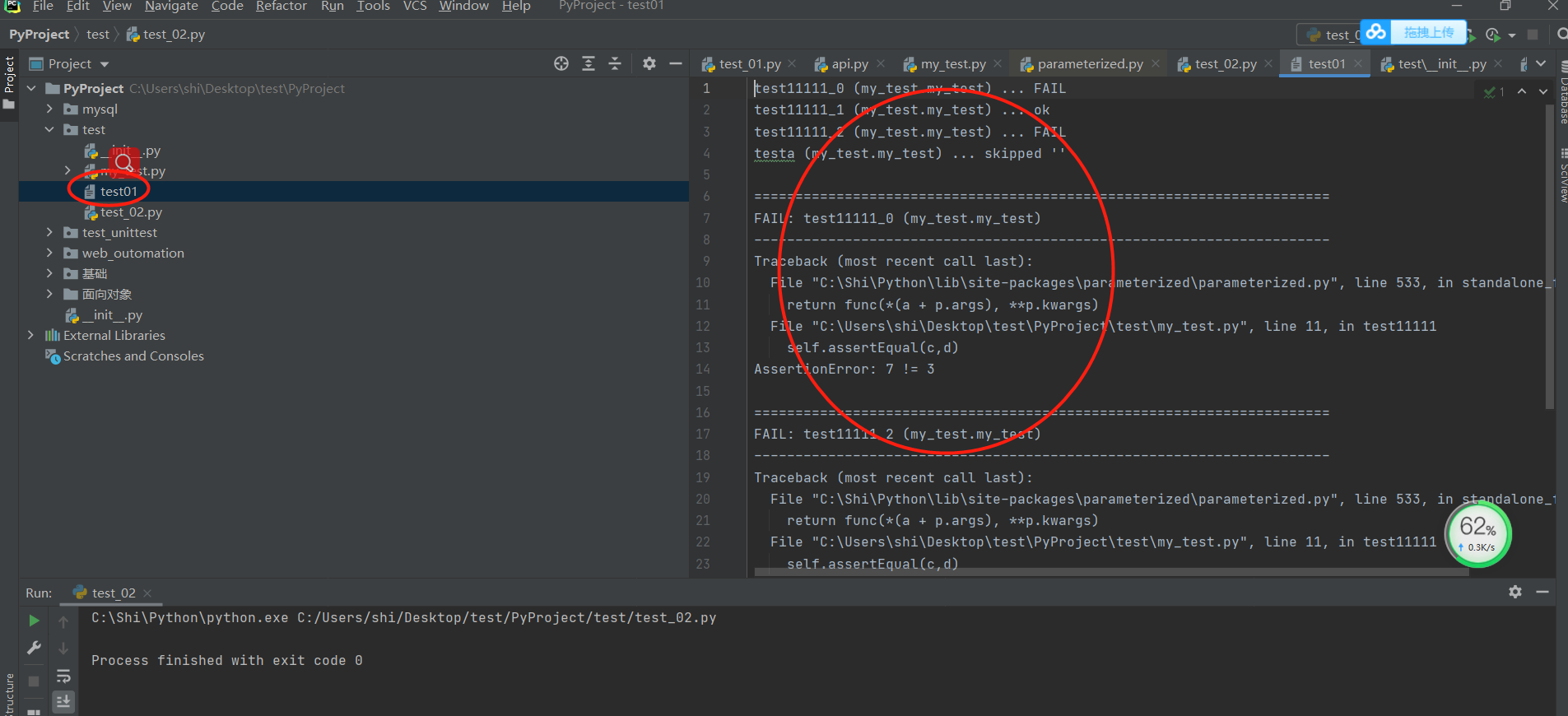
## Unittest 自带测试报告

* 在实例化TextTestRunner对象的时候,需要写参数

stream=file, verbosity=2  
file代表用open打开的一个文件  
verbosity=2,固定

* 第一步:用open,w方式打开测试报告文件
* 第二步:实例化TextTestRunner对象
* 第三步调用对象的run方法执行测试套件
* 第四步:关闭open打开的文件

import unittest  
suite=unittest.TestLoader*()*.discover*(*'.',"my\*.py"*)*#把当前路径下 所有以my开头的模块内、 以test开头的测试用例装进测试套件中，  
file=open*(*'test01',"w",encoding='utf8'*)*#打开文件读取  
runner=unittest.TextTestRunner*(*stream=file,verbosity=2*)*#创建testrunner对象  
runner.run*(*suite*)*#把测试套件放到 runner里面，便可执行 套件中的用例



## HTML测试报告

* 把文件HTMLTestRunner.py拷贝到项目目录下
* 在代码中导入模块from HTMLTestRunner import HTMLTestRunner
* 调用HTMLTestRunner(stream=file, title="我的第一个html测试报告")
  + 第一个参数是用open打开的文件, 打开的文件扩展名一定是.html
  + open打开文件的时候,用wb,不用指定字符集
* 调用runner对象的run方法执行测试套件
* 关闭open打开的文件

import unittest  
from HTMLTestRunner import HTMLTestRunner  
suite=unittest.TestLoader*()*.discover*(*'.',"my\*.py"*)*#把当前路径下 所有以my开头的模块内、 以test开头的测试用例装进测试套件中，  
file=open*(*'test888.html',"wb",*)*#打开文件读取  
# runner=unittest.TextTestRunner(stream=file,verbosity=2)#创建testrunner对象  
runner=HTMLTestRunner*(*stream=file,title='我的第一个测试文件'*)*runner.run*(*suite*)*#把测试套件放到 runner里面，便可执行 套件中的用例