像风一样的自由

人生苦短，我也用python!!!

python单元测试unittest

标签： 单元测试 python 测试 module import 工作

2011-12-26 23:41 44453人阅读 评论(7) 收藏 举报

分类： python应用（42）

单元测试作为任何语言的开发者都应该是必要的，因为时隔数月后再回来调试自己的复杂程序时，其实也是很崩溃的事情。虽然会很快熟悉内容，但是修改和调试将是一件痛苦的事情，如果你在修改了代码后出现问题的话，而单元测试可以帮助我们很快准确的定位到问题的位置，出现问题的模块和单元。所以这是一件很愉快的事情，因为我们知道其它修改 地方仍然是正常工作的，而我们目前的唯一问题就是搞定眼前这个有点问题的“家伙”。所以工作会在轻松中开始，并且很快将会结束，因为你已经知道很多信息了。

单元测试自然是对程序中最小的可测试模块--函数来进行测试；因为单元测试的对象是函数，也就是说你得被测试对象一定要有输出结果，哪怕就是异常输出，也得有输出，以便单元测试模块能够捕获返回值，并且与预期值进行比较，从而得出测试通过与否。

单元测试的加载方式有2种：一种是通过unittest.main()来启动单元测试的测试模块；一种是添加到testsuite集合中再加载所有的被测试对象，而testsuit里存放的就是单元测试的用例，下面分别列出了2种方法的使用。

1.1 测试模块中的函数：

被测模块：

```python
#!/usr/bin/env python
#encoding: utf-8

def sum( x, y):
        return x+y


def sub( x, y):
        return x-y
```

单元测试模块：

```python
#!/usr/bin/env python
#encoding: utf-8

import unittest
import myclass

class mytest(unittest.TestCase):

    ##初始化工作
    def setUp(self):
        pass
```

```python

    #退出清理工作
    def tearDown(self):
        pass

    #具体的测试用例，一定要以test开头
    def testsum(self):
        self.assertEqual(myclass.sum(1, 2), 2, 'test sum fail')


    def testsub(self):
        self.assertEqual(myclass.sub(2, 1), 1, 'test sub fail')


if __name__ =='__main__':
    unittest.main()
```

测试结果：【F表示一个fail， F前的点表示一个通过，有E的话表示程序自身异常】

```
[python]

.F
======================================================================
FAIL: testsum (__main__.mytest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\xiaowu\workspace\mypython\unitTest.py", line 19, in testsum
    self.assertEqual(myclass.sum(1, 2), 2, 'test sum fail')
AssertionError: test sum fail

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)
```

1.2 测试模块类中的函数：

被测模块：

```
[python]

#!/usr/bin/env python
#encoding: utf-8

class myclass:
    def __init__(self):
        pass


    def sum(self, x, y):
        return x+y


    def sub(self, x, y):
        return x-y
```

单元测试模块：

```
[python]

#!/usr/bin/env python
#encoding: utf-8

import unittest
import myclass

class mytest(unittest.TestCase):

    ##初始化工作
    def setUp(self):
        self.tclass = myclass.myclass()    ##实例化了被测试模块中的类

    #退出清理工作
    def tearDown(self):
        pass

    #具体的测试用例，一定要以test开头
    def testsum(self):
        self.assertEqual(self.tclass.sum(1, 2), 3)
```

```
20.
21.
22. if __name__ =='__main__':
23.     unittest.main()
```

运行结果：

```
[python]
01. .
02. ----------------------------------------------------------------------
03. Ran 1 test in 0.000s
04.
05. OK
```

这种方式执行单个测试文件时使用-v参数可以获得更多的测试结果信息。如：mytest.py -v

2 加载测试套件

好吧，在运用测试套件进行单元测试之前，我想还是稍微研究一下unittest模块的内容有哪些，其大概的运行方式是什么样的。而后在给出根据各种情况如何制定单元测试套件。

首先，自然是查看unittest模块有哪些成员啦！

```
[python]
01. >>import unittest
02. >>dir(unittest)
03. ['FunctionTestCase', 'TestCase', 'TestLoader', 'TestProgram', 'TestResult', 'Tes
04. tSuite', 'TextTestRunner', '_CmpToKey', '_TextTestResult', '_WritelnDecorator',
05. '__all__', '__author__', '__builtins__', '__doc__', '__email__', '__file__', '__
06. metaclass__', '__name__', '__package__', '__unittest', '__version__', '_makeLoad
07. er', '_strclass', 'defaultTestLoader', 'findTestCases', 'getTestCaseNames', 'mai
08. n', 'makeSuite', 'os', 'sys', 'time', 'traceback', 'types']
```

可以看到其自身的成员也不是很多，大概包括有：

['FunctionTestCase', 'TestCase', 'TestLoader', 'TestProgram', 'TestResult',

'TestSuite','TextTestRunner', '_CmpToKey', '_TextTestResult', '_WritelnDecorator',

'defaultTestLoader','findTestCases', 'getTestCaseNames', 'main', 'makeSuite']

好吧,我们大概看看具体都是干什么的

```
[python]
01. >>memblist = ['FunctionTestCase', 'TestCase', 'TestLoader', 'TestProgram', 'TestResult',\
02. 'TestSuite','TextTestRunner', 'defaultTestLoader','findTestCases', 'getTestCaseNames', \
03. 'main', 'makeSuite']
04. >>for memb in memblist:
05. ..  cur = getattr(unittest, memb)
06. ..  print help(cur)
```

'FunctionTestCase':函数测试用例，即给一个函数作为参数，返回一个testcase实例，可选参数有set-up，tear-down方法

'TestCase'：所有测试用例的基本类，给一个测试方法的名字，返回一个测试用例实例

'TestLoader'：测试用例加载器，其包括多个加载测试用例的方法。返回一个测试套件

loadTestsFromModule(self, module)--根据给定的模块实例来获取测试用例套件

loadTestsFromName(self, name, module=None)

--根据给定的字符串来获取测试用例套件，字符串可以是模块名，测试类名，测试类中的测试方法名，或者一个可调用的是实例对象

这个实例对象返回一个测试用例或一个测试套件

loadTestsFromNames(self, names, module=None) --和上面功能相同，只不过接受的是字符串列表

loadTestsFromTestCase(self, testCaseClass)--根据给定的测试类，获取其中的所有测试方法，并返回一个测试套件

'TestProgram'：命令行进行单元测试的调用方法，作用是执行一个测试用例。其实unittest.main()方法执行的就是这个命令，

而这个类实例时默认加载当前执行的作为测试对象，

原型为 __init__(self, module='__main__', defaultTest=None, argv=None, testRunner=xx, testLoader=xx)

其中module='__main__'就是默认加载自身

'TestResult'：测试用例的结果保存实例，通常有测试框架调用

'TestSuite'：组织测试用例的实例，支持测试用例的添加和删除，最终将传递给testRunner进行测试执行

'TextTestRunner'：进行测试用例执行的实例，其中Text的意思是以文本形式显示测试结果。显示测试名称，即完成的测试结果，其过同执行单元测试脚本时添加-v参数

'defaultTestLoader':其实就是TestLoader

'findTestCases', 'getTestCaseNames'：这个2个就不用解释了

'main': 其实就是TestProgram

'makeSuite'：通常是由单元测试框架调用的，用于生产testsuite对象的实例

至此，我们知道了。其实整个单元测试框架的逻辑出来了，分三步走：第一步testloader根据传入的参数获得相应的测试用例，即对应具体的测试方法，

然后makesuite在把所有的测试用例组装成testsuite，最后把testsiute传给testrunner进行执行。

而我们通常执行的unittest.main()，其实就是unittest.testprom方法，其执行的功能就是上面分析的三步，在第一步中其传入的参数是自身的模块__main__；

在第二步中把自身模块中的所有测试类中中的测试方法提取出来，并生成测试套件；最后再把测试套件传递给testrunner进行具体的测试。

最后给出一个完整的单元测试组织代码，把该代码放到单元测试用例文件的同一个目录后执行该脚本，即可执行所有的测试用例文件。

【测试用例文件必须为test开头，如：testxxx.py, 当然这个文件本身是一个单元测试的文件】

```python
#!/usr/bin/env python
#encoding: utf-8
#该代码源自深入python
import unittest
import myclass
import re
import os
import sys


def testAllinCurrent():
    path = os.path.abspath(os.path.dirname(sys.argv[0]))
    files = os.listdir(path)
    test = re.compile("test\.py{1}quot;, re.IGNORECASE)
    files = filter(test.search, files)
    filenameToModuleName = lambda f: os.path.splitext(f)[0]
    moduleNames = map(filenameToModuleName, files)
    modules = map(__import__, moduleNames)

    load = unittest.defaultTestLoader.loadTestsFromModule
    return unittest.TestSuite(map(load, modules))

if __name__ == "__main__":
    unittest.main(defaultTest="regressionTest")
```


上一篇 【转载】C语言头文件的使用

下一篇 C语言交换两个变量值不利用额外变量

我的同类文章

python应用（42）

- UMeditor在线web编辑器 -... 2015-06-03 阅读 853
- python实现tail -f命令功能 2014-08-12 阅读 4046
- python发送邮件sendmail-... 2014-05-31 阅读 3847
- python实现测试脚本的关... 2013-06-08 阅读 2320
- python抓网页资源小脚本 2012-11-19 阅读 1174
- python类型比较的3种方式 2012-10-22 阅读 8148
- 代码中加入如下内容，可... 2014-08-27 阅读 1517
- python比较2个xml内容 2014-07-31 阅读 1667
- python循环遍历文件操作 2013-06-27 阅读 3579
- webpy中配置发送邮件服务 2013-01-30 阅读 1365
- python中telnetlib模块的使用 2012-10-22

阅读 23489

更多文章

参考知识库

Python知识库

8368 关注 | 812 收录

猜你在找

Python自动化开发实战视频课程-全新基础篇
软件测试基础
《C语言/C++学习指南》单步调试
Python 零基础到实战
微信公众平台开发入门
Python单元测试框架-unittest
Python Unittest 自动化单元测试框架Demo
python-unittest模块单元测试
python 单元测试assert 或者 unittestTestCase
Python 中 unittest单元测试的学习

查看评论

7楼 sinat_33540202 2015-12-28 14:38发表

楼主还在吗？求解答
我还是不明白你写的第二段 2 加载测试套件 里面那个完整的单元测试组织代码 是怎么加载了测试套件的， 我复制了代码到python里运行 这两行都报错呢
test = re.compile("test\.py{1}quot;, re.IGNORECASE)
unittest.main(defaultTest="regressionTest")

6楼 sbcelso 2015-10-19 17:17发表

内容祥实，谢谢分享！！！

5楼 clover1107 2015-09-08 17:47发表

楼主你好啊！看到你python方面的技术博客。我是麦子学院的讲师顾问，目前我们在建设这方面的课程，不知道你是否有兴趣合作视频课程呢？我的qq:2318019782 方便的话可以详谈一下，期待你的回复。

4楼 白熊花田 2015-07-20 10:39发表


Thks

3楼 杰瑞26 2014-01-20 22:36发表


好文章，简单易懂，收藏了。

2楼 as 2013-03-05 10:24发表


我按你给的例子写了一个测试脚本，
如果我在一个testCase子类中写了两个方法testa和testb, 其中testa的执行要依赖于testb产生的结果，那么我如何使testb在testa之前执行呢？ 好像unittest.main()默认按字母优先顺序执行.

1楼 as 2013-02-23 15:02发表


在测试你的第一个例子（测试模块类中的函数）时我除了你给出的报错信息之外还报出如下的信息，请问是什么原因？

```
[python]
01. Traceback (most recent call last):
02.   File "D:/Python Test/unit_test.py", line 26, in <module>
```

```
    unittest.main()
  File "D:\Python27\lib\unittest\main.py", line 95, in __init__
    self.runTests()
  File "D:\Python27\lib\unittest\main.py", line 231, in runTests
    sys.exit(not self.result.wasSuccessful())
SystemExit: True
```