Python-单元测试 Nosetests

单元测试是用来对一个模块、一个函数或者一个类来进行正确性检验的测试工作。 unittest 是Python必备的一个单元测试框架。

TODO https://www.bbsmax.com/A/x9J2j0yWJ6/

Noestests介绍

nostests优点:

- 编写测试更容易。nose可以自动识别继承于unittest.TestCase的测试单元，并执行测试，而且，nose也可以测试非继承于unittest.TestCase的测试单元。nose提供了丰富的API便于编写测试代码。
- 执行测试更容易。 只要遵循一些简单的规则去组织你的类库和测试代码，nose是可以自动识别单元测试的。执行测试是非常耗资源的，但是，一段第一个测试模块被加载后，nose就开始执行测试。
- 建立测试环境更容易。
- 做你想做的事情更容易。 nose拥有很多内置的插件帮助进行暑输出抓取、错误查找、代码覆盖、文档测试（doctest）等等。同样，如果你不喜欢这些内置插件提供的功能或者这些插件不能满足你的项目结构，你可以自定义开发插件来完成你想要做的事情。

unittest核心工作原理

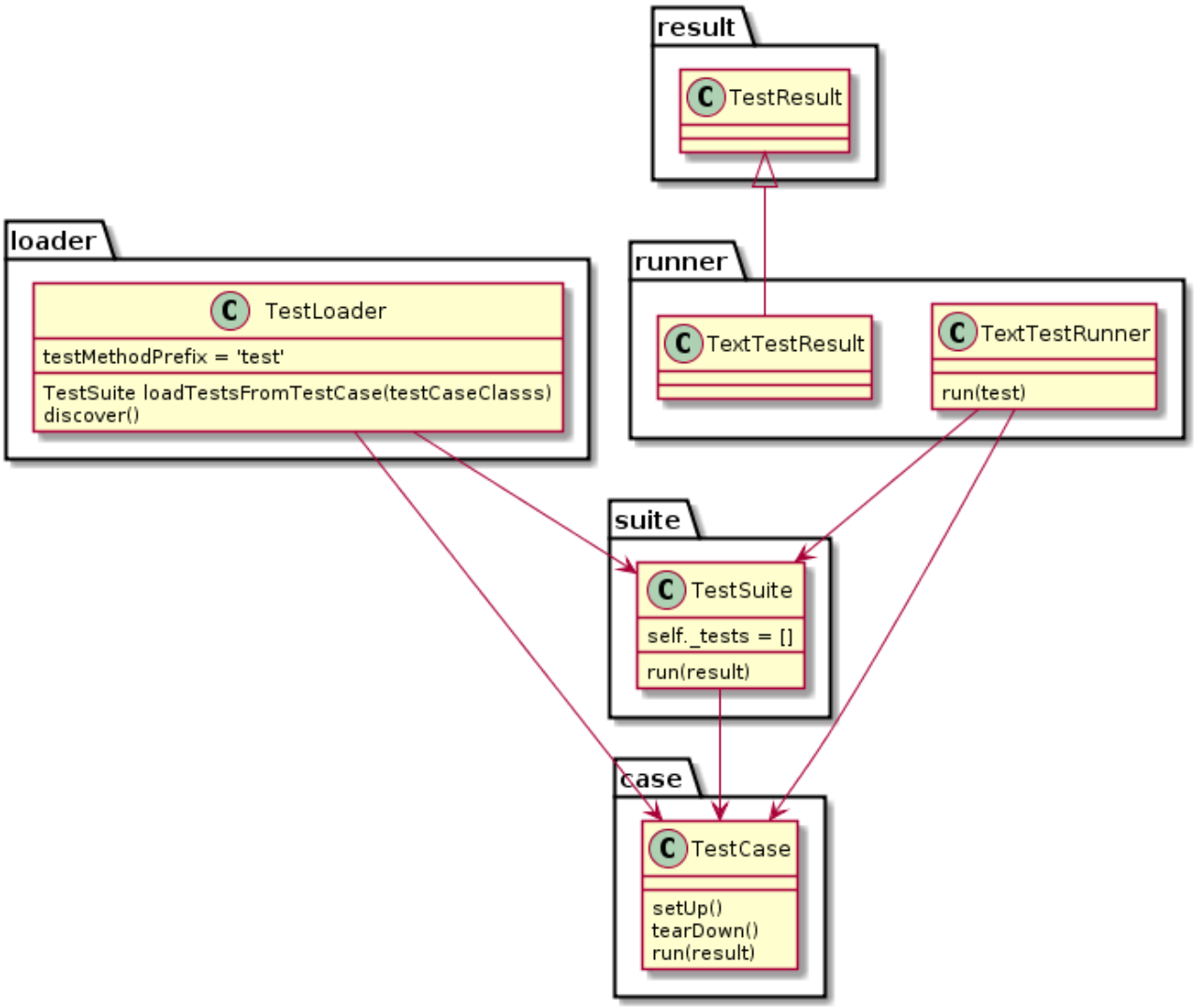
unittest中最核心的四个概念

- **test case:** 测试用例。一个完整的测试流程。
- **test suite:** 多个测试用例集合。带有顺序性
- **test runner:** 执行测试用例
- **test fixture:** 负责测试环境的创建和销毁

unittest关键类说明

- TestCase 也就是测试用例
- TestSuite 多个测试用例集合在一起，就是TestSuite
- TestLoader是用来加载TestCase到TestSuite中的

- TestRunner是来执行测试用例的,测试的结果会保存到TestResult实例中，包括运行了多少测试用例，成功了多少，失败了多少等信息



Noestest使用

安装部署

```
pip install nose
```

插件

TODO 如何配置插件

使用pip安装所需要的插件，然后通过nosetests命令行配置插件。 如果通过nose.main()或者nose.run()执行测试，可以将要使用的插件关键字参数的列表传递进去

```
# 显示已安装插件
nosetests --plugins
```

```
# -v 显示插件详细信息

(env) [scfan@scfan project]$  nosetests --plugins -v
Plugin capture
  score: 1600
  Output capture plugin. Enabled by default. Disable with ``-s`` or
  ``--nocapture``. This plugin captures stdout during test execution,
  appending any output captured to the error or failure output, should
  the test fail or raise an error.

Plugin failuredetail
  score: 1600
  Plugin that provides extra information in tracebacks of test
  failures.

Plugin xunit
  score: 1500
  This plugin provides test results in the standard XUnit XML format.

Plugin deprecated
  score: 1000
  Installs a DEPRECATED error class for the DeprecatedTest exception.
  Enabled by default.

Plugin skip
  score: 1000
  Plugin that installs a SKIP error class for the SkipTest exception.
  When SkipTest is raised, the exception will be logged in the skipped
  attribute of the result, 'S' or 'SKIP' (verbose) will be output, and
  the exception will not be counted as an error or failure.

Plugin multiprocess
  score: 1000
  Run tests in multiple processes. Requires processing module.

Plugin logcapture
  score: 500
  Log capture plugin. Enabled by default. Disable with --nologcapture.
  This plugin captures logging statements issued during test
  execution, appending any output captured to the error or failure
  output, should the test fail or raise an error.

Plugin coverage
  score: 200
  Activate a coverage report using Ned Batchelder's coverage module.
.....
```

nosetests参数

```
-V,—version
```

输出nose的版本

-p,–plugins

输出可获取的插件列表。

-v=DEFUALT,–verbose=DEFAULT

使用更多的verbose

--verbosity=VERBOSITY

设置verbosity;--verbosity=2与-v设置一致

-q,–quiet=DEFAULT

使用更少的verbose

-c=FILES,–config=FILES

设置配置文件。可以设置很多次，然后将所有的配置文件合并。

-w=WHERE,--where=WHERE

设置查找的根目录。

-py3where=WHERE

顾名思义，针对python3.x以上设置查找路径。

-m=REGEX,--match=REGEX,–testmatch=REGEX

设置用于自动化收集用例的正则表达式。

--tests=NAMES

执行这些测试。

--debug-log=FILE

设置调试的日志文件路径。

--logging-config=FILE,–log-config=FILE

设置日志文件的配置文件。

-I=REGEX，–ignore-files=REGEX

设置自动收集测试用例时忽略的正则表达式。

-e=REGEX,–exclude=REGEX

排除要执行的测试用例的正则表达式

```
-i=REGEX,—include=REGEX

    包含要执行的测试用例的正则表达式

-x,—stop

  执行测试发生错误后，停止执行测试。

--noexe

    不查找可以执行文件。

-a=ATTR,--attr=ATTR

  只执行包含ATTR属性的测试用例。

-A=EXPR,—eval-attr=EXPR

  只执行属性与EXPR匹配的测试用例。

-s,—nocapture

    不抓取标准输出(stdout)

--nologcapture

    禁止使用日志插件

--logging-format=FORMAT

    打印语句的自定义格式

--logging-datefmt=FORMAT

  日志的日期时间格式

--logging-filter=FILTER

  日志语句的过滤器。

--logging-clear-handlers

  清除日志的其他handlers

--logging-level=DEFUALT

    设置日志的等级

--with-coverage

    开启coverage插件

--cover-package=PACKAGE
```

```
    限定coverage所在包
--cover-erase
   在执行之前 清除上次coverage统计结果
--cover-testes
   在coverage报告中包含测试模块
--cover-html
   产生html的coverage报告
--cover-html-dir=DIR
   设置存储html的目录
--cover-xml
    产生xml的coverage报告
--cover-xml-file=FILE
    设置存储coverage报告的xml文件
--cover-config-file=DEFUALT
   设置coverage的配置文件
-pdb
    当测试失败或产生错误是进入调试模式
--pdb-failures
   当测试失败时进入调试模式
--pdb-errors
   当测试产生错误时进入调试模式
--with-doctest
开启doctest插件
--doctest-tests
   在测试模块中查询doctests
--with-profile
    开启profile插件
```

```
--profile-sort=SORT

  设置profiler 输出排序

--profile-stats-file=FILE

  设置统计所在的文件地址

--with-id

  开启TestId插件

--processes=NUM

  开始测试处理器的个数

--processes-timeout=SECONDS

  设置超时时间。

--with-xuint

  开始Xunit插件

--xuint-file=FILE

  设置XUnit报告所在的xml文件

--all-modules

   开启AllModules插件

--collect-only

  开启只收集测试功能。只收集测试用例及输出测试名字，而不执行测试
```

Nosetests语法说明

TODO 详细说明

扩展 nose-printlog

可以在控制台输出日志，也可以将日志显示在caplog中

```
# 安装插件
pip install nose-printlog
# 运行命令样例
nosetests -sv  test_data_load.py   --logging-format="%(asctime)s:%(name)s:%
(levelname)s: %(message)s"  --with-printlog
```

附件

用HTMLTestRunner输出漂亮的HTML报告

参考链接：CSDN博客unittest单元测试

tools_unittest_HTMLTestRunner.py 下载链接

测试代码

tools_mathfunc.py

```
(env) [scfan@WOM tools]$ cat tools_mathfunc.py
#! -*- coding:utf-8 -*-
u"""
    常见数学类函数

- add
- minux
- divide
- multi

"""

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

def minus(x,y):
    return x-y

def divide(x,y):
    return x/y

def multi(x,y):
    return x * y
```

test_unittest_htmlreport.py

```
# coding=utf-8
u"""
    单元测试unittest 生成HTML报告测试

运行命令:
cd /home/scfan/pro/server && python -m  pro.tests.test_unittest_htmlreport

"""
import unittest
from ..tools.tools_unittest_HTMLTestRunner import HTMLTestRunner
from ..tools.tools_mathfunc import *
```

```
class TestMathFunc(unittest.TestCase):

    def test_add(self):
        self.assertEqual(3, add(1, 2))
        self.assertNotEqual(3, add(2, 2))

    def test_minus(self):
        self.assertEqual(1, minus(3, 2))

    def test_multi(self):
        self.assertEqual(6, multi(3, 2))

    def test_divide(self):
        self.assertEqual(2, divide(6, 3))
        self.assertEqual(2.5, divide(5, 2))

if __name__ == '__main__':
    suite = unittest.TestSuite()

    tests = [TestMathFunc("test_add"), TestMathFunc("test_minus"),
TestMathFunc("test_divide")]
    suite.addTests(tests)

    with open('HTMLReport.html', 'w') as f:
        runner = HTMLTestRunner(stream=f,
                                title = 'MathFunc Test Report',
                                description='generated by HTMLTestRunner.',
                                verbosity=2
                                )
        runner.run(suite)
```

参考资源

参考链接：

- Python官方unittest文档
- CSDN博客unittest单元测试
- CSDN博客unittest单元测试B
- Python多个单元测试软件说明
- Nose全面使用介绍

扩展:

- Python测试框架对比
- 软件通用测试流程和设计方法详解