**Python unittest单元测试**

## **一、什么是unittest**

unittest是python内置的单元测试框架，具有编写测试用例，组织用例，用例执行，输出报告等，可以搭建自动化测试。不仅可以用于单元测试，还可以用于web自动化测试

unittest分为5个概念：test case；test suite；testLoader；testrunner；test fixture。

test case(测试用例)：一个完整的测试单元，执行该测试单元可以完成对某一个问题的验证。完整性体现在：测试前的准备(setup),执行测试代码（run），以及测试后环境还原（teardown）。

test suite(测试套件):多个测试用例的集合，测试套件或者测试计划

testLoader():加载test case 到test suite中，其中loadTestFrom\_()方法用于寻找Test case ，并创建它们的实例，然后添加到Test suite中，返回test suite实例。

test runner(测试执行器):执行测试用例，并将测试结果保存到TextTestResult实例中，包括运行了多少次测试用例，成功了多少，失败了多少等信息。

test fixture(测试夹具)：一个测试用例的初始化准备以及环境还原，主要是setUp()和setDown()方法。

1. **工作原理**

通过unittest类的调用分析，框架的原理如下：编写test case ，由testloader加载test case 到test suite中，然后由TextTestRunner来执行test suite 最后将运行结果保存在TextTestResult中。

1. unittest模块的属性

1.unittest.TestCase:所有的测试用例继承这个基本类

Class AbcTest(unittest.TestCase):

2.unittest.main():使用她可以方便的将一个单元测试模块变为可直接运行的测试脚本，main()方法使用TestLoader类来搜索所有包含在该模块中以“test”命名开头的测试方法，并自动执行他们。执行方法的默认顺序是：根据ASCII码的顺序加载测试用例，数字与字母的顺序为：0-9，A-Z，a-z。所以以A开头的测试用例方法会优先执行，以a开头会后执行。

3.unittest.TestSuite()：unittest框架的TestSuite()类是用来创建测试套件的。

4.unittest.TextTextRunner():unittest框架的TextTextRunner()类，通过该类下面的run()方法来运行suite所组装的测试用例，入参为suite测试套件。

5.unittest.defaultTestLoader(): defaultTestLoader()类，通过该类下面的discover()方法可自动更具测试目录start\_dir匹配查找测试用例文件（test\*.py），并将查找到的测试用例组装到测试套件，因此可以直接通过run()方法执行discover。用法如下：

discover=unittest.defaultTestLoader.discover(test\_dir, pattern='test\_\*.py')

6.unittest.skip():装饰器，当运行用例时，有些用例可能不想执行等，可用装饰器暂时屏蔽该条测试用例。一种常见的用法就是比如说想调试某一个测试用例，想先屏蔽其他用例就可以用装饰器屏蔽。

@unittest.skip(reason): skip(reason)装饰器：无条件跳过装饰的测试，并说明跳过测试的原因。

@unittest.skipIf(reason): skipIf(condition,reason)装饰器：条件为真时，跳过装饰的测试，并说明跳过测试的原因。

@unittest.skipUnless(reason): skipUnless(condition,reason)装饰器：条件为假时，跳过装饰的测试，并说明跳过测试的原因。

@unittest.expectedFailure(): expectedFailure()测试标记为失败。

1. **TestCase类的属性**

1.setUp():setUp()方法用于测试用例执行前的初始化工作。如测试用例中需要访问数据库，可以在setUp中建立数据库连接并进行初始化。如测试用例需要登录web，可以先实例化浏览器。

2.tearDown():tearDown()方法用于测试用例执行之后的善后工作。如关闭数据库连接。关闭浏览器。

3.assert\*():一些断言方法：在执行测试用例的过程中，最终用例是否执行通过，是通过判断测试得到的实际结果和预期结果是否相等决定的。

4.assertEqual(a,b，[msg='测试失败时打印的信息']):断言a和b是否相等，相等则测试用例通过。

5.assertNotEqual(a,b，[msg='测试失败时打印的信息']):断言a和b是否相等，不相等则测试用例通过。

6.assertTrue(x，[msg='测试失败时打印的信息'])：断言x是否True，是True则测试用例通过。

7.assertFalse(x，[msg='测试失败时打印的信息'])：断言x是否False，是False则测试用例通过。

8.assertIs(a,b，[msg='测试失败时打印的信息']):断言a是否是b，是则测试用例通过。

9.assertNotIs(a,b，[msg='测试失败时打印的信息']):断言a是否是b，不是则测试用例通过。

10.assertIsNone(x，[msg='测试失败时打印的信息'])：断言x是否None，是None则测试用例通过。

11.assertIsNotNone(x，[msg='测试失败时打印的信息'])：断言x是否None，不是None则测试用例通过。

12.assertIn(a,b，[msg='测试失败时打印的信息'])：断言a是否在b中，在b中则测试用例通过。

13.assertNotIn(a,b，[msg='测试失败时打印的信息'])：断言a是否在b中，不在b中则测试用例通过。

14.assertIsInstance(a,b，[msg='测试失败时打印的信息'])：断言a是是b的一个实例，是则测试用例通过。

15.assertNotIsInstance(a,b，[msg='测试失败时打印的信息'])：断言a是是b的一个实例，不是则测试用例通过。

五**、TestSuite类的属性（组织测试用例时可以用到）**

1.addTest(): addTest()方法是将测试用例添加到测试套件中，如下方，是将test\_baidu模块下的BaiduTest类下的test\_baidu测试用例添加到测试套件。

**suite = unittest.TestSuite()**

**suite.addTest(test\_baidu.BaiduTest('test\_baidu'))**

2.run(): run()方法是运行测试套件的测试用例，入参为suite测试套件。

**runner = unittest.TextTestRunner()**

**runner.run(suite)**

测试思路，案例，见文件：测试用例思路.txt