单元测试老生常谈了，但是测试要会单元测试还是近几年在国内流行起来的，测试开发岗位那是必备技能了，所以这个单元测试还是要接触下，就算以后的岗位中用不上，但万一面试时问到单元测试，也不至于太狼狈，或者学到这个技能不是不可以往测试开发岗位发展下。我们看下python怎么实现单元测试。

　　python的优点想必大家已经很了解了，现在想到要单元测试，那么肯定有牛人已经探好路子，并研发了这个模块，我们用就好了，没错，就是unittest模块：

　　TestCase 测试用例，就是功能里那样一条条用例

　　TestSuite 多个测试用例集合在一起，就是TestSuite，就是一个功能模块的所有用例放这里了

　　TestLoader是用来加载TestCase到TestSuite中的，这个方法很好玩，可以将一个目录下的所有python文件里的测试用例抠出来

　　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

**1.unittest测试的基本方法**

[复制代码](javascript:void(0);)

1 import unittest #导入unittest模块

2 import HTMLTestRunner,xmlrunner #导入生成报告的模块，html生成的是方便给自己看的，xml生成的报告是以后给jekens用的，能自动识别测试报告内容。

3 class MyTest(unittest.TestCase): #定义一个测试用例的类

4 def testa(self): #定义一个测试用例，注意，测试用例必须test开始，不然不会当做是测试用例

5 '''a''' #描述，会同测试用例标题一并显示在测试报告里

6 self.assertEqual(1,1) #测试用例断言，比较预期结果与实际结果，这里1==1，显然结果是pass

7 def testhaha(self):

8 '''b'''

9 self.assertEqual(2, 1) #测试用例断言，比较预期结果与实际结果，这里2==1，显然结果是Fail

10 def testb(self):

11 '''c'''

12 self.assertEqual(3, 2) #测试用例断言，比较预期结果与实际结果，这里3==1，显然结果是Fail

13

14 class MyTest2(unittest.TestCase):

15 def testc(self):

16 '''d'''

17 self.assertEqual(1,1)

18 def testhaha(self):

19 '''e'''

20 self.assertEqual(2, 1)

21 def testd(self):

22 '''f'''

23 self.assertEqual(3, 2)

24

25 if \_\_name\_\_=='\_\_main\_\_':

26 # unittest.main() #运行所有的测试用例，运行下面的代码时这个要注释

27 suite=unittest.TestSuite() #定义一个测试套件

28 # suite.addTest(MyTest('testa')) #添加测试用例方法一：往测试套件里新增一条测试用例，测试用例只能用一种方法，同时两种会报错

29 # suite.addTest(MyTest('testb'))

30 suite.addTest(unittest.makeSuite(MyTest)) #添加测试用例方法二：往测试套件里新增这个类下的所有测试用例

31 suite.addTest(unittest.makeSuite(MyTest2))

32 # fw=open('test.html','wb') #定义一个文件对象，给后面的HTMLTestRunner生成测试报告用，注意打开方式必须是wb

33 # runner=HTMLTestRunner.HTMLTestRunner(stream=fw,title='Testing',description='miaoshu') #生成测试报告方法一：HTMLTestRunner，需要指定文件对象和标题

34 runner=xmlrunner.XMLTestRunner(output='.') #生成测试报告方法二：xmlrunner，这个方法只要指定测试报告目录就可以

35 runner.run(suite) #运行，注意上述方法我写一起了，运行的话只能运行一种，另一个要注释

[复制代码](javascript:void(0);)

**2.unittest中的setUp和tearDown的应用**

[复制代码](javascript:void(0);)

1 #setUp和tearDown的运用

2 import unittest

3 class MyTest(unittest.TestCase):

4 @classmethod

5 def setUpClass(cls): #类开始前运行，比如在执行这些用例之前需要备份数据库

6 print('1')

7

8 @classmethod

9 def tearDownClass(cls): #类结束后运行，比如在执行这些用例之后需要还原数据库

10 print('0')

11 def setUp(self): #测试用例执行前运行

12 print('a')

13 def tearDown(self): #测试用例执行后运行

14 print('z')

15 def testa(self):

16 print('测试用例1')

17 self.assertEqual(1,1)

18 def testb(self):

19 print('测试用例2')

20 self.assertEqual(1,1)

21 if \_\_name\_\_=='\_\_main\_\_':

22 unittest.main()

[复制代码](javascript:void(0);)