session中设置cookie操作



实战：



re模块的使用：

从一个大文本中匹配需要的信息的过程

两个知识点支撑：

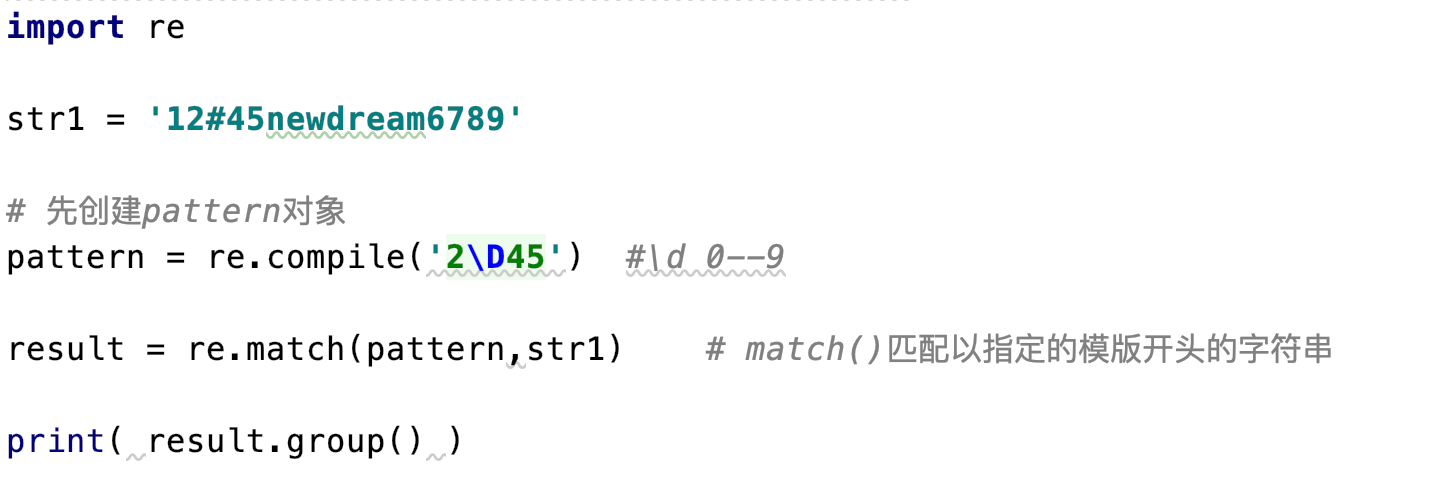
第一个知识点 正则表达式中元字符的理解

第二个知识点 re模块中方法的使用

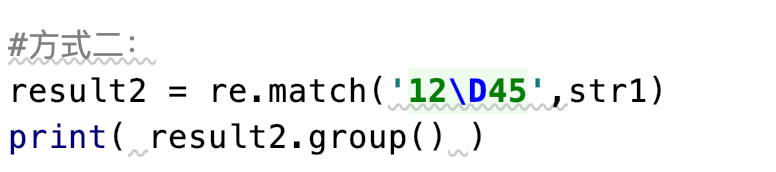
在python中使用正则的方式：

方式一：先创建pattern对象 ，再利用函数进行匹配

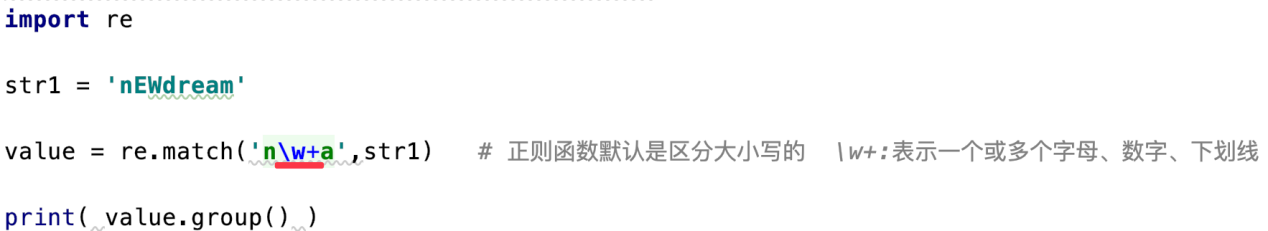
pattern可以理解为一个匹配模式 字符匹配模版对象 ：从一个字符串找出能匹配模版的子字符串



方式二：不需要创建pattern对象，直接利用函数进行匹配

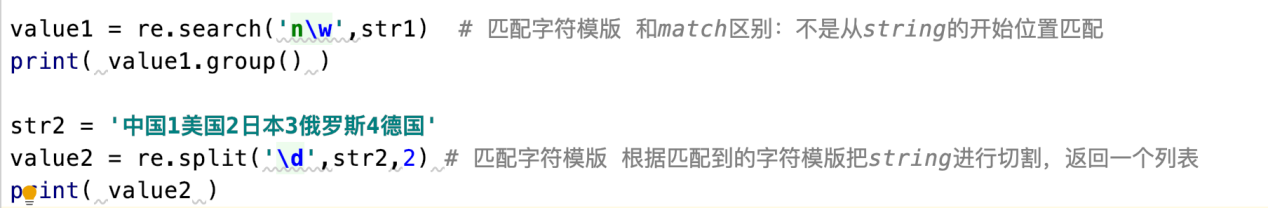


原字符 量词 配合使用



Search():匹配字符模版 和match区别：不是从string的开始位置匹配

Split(): 根据匹配到的字符模版把string进行切割，返回一个列表

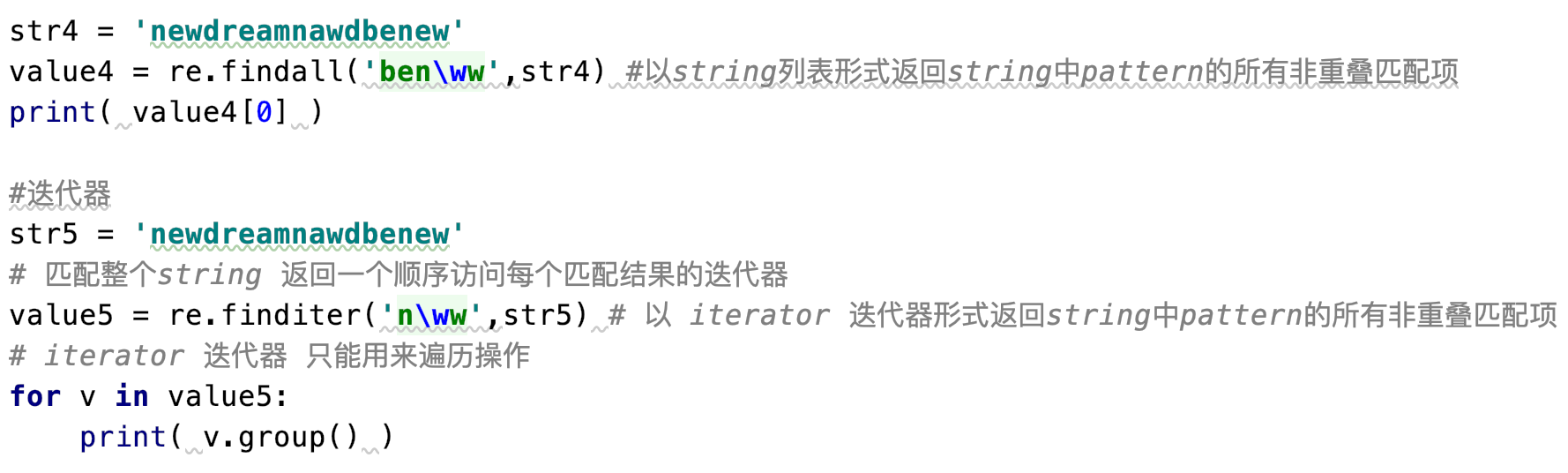


def split(pattern, string, maxsplit=0, flags=0)

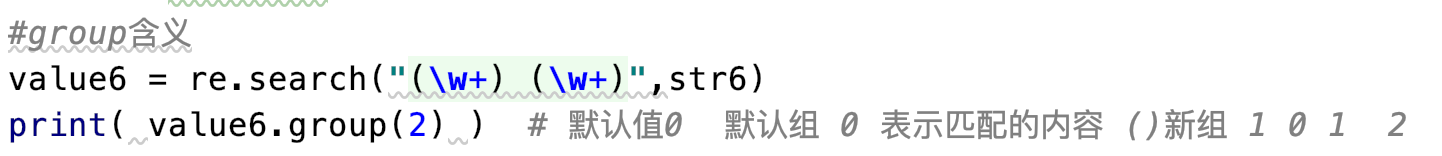
Maxsplit :指定切割次数

Findall():以string列表形式返回string中pattern的所有非重叠匹配项

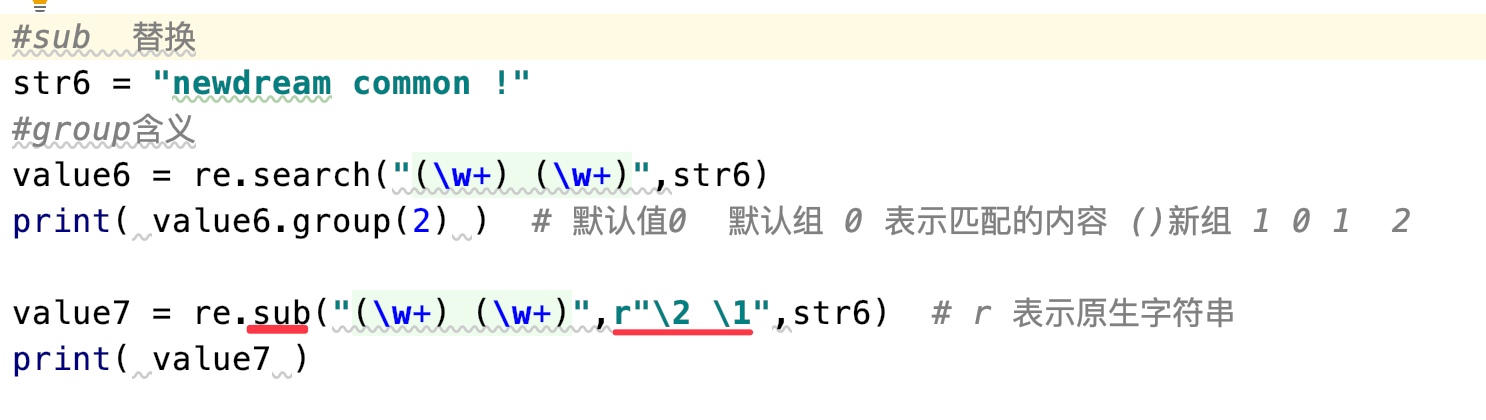
finditer():以 iterator 迭代器形式返回string中pattern的所有非重叠匹配项



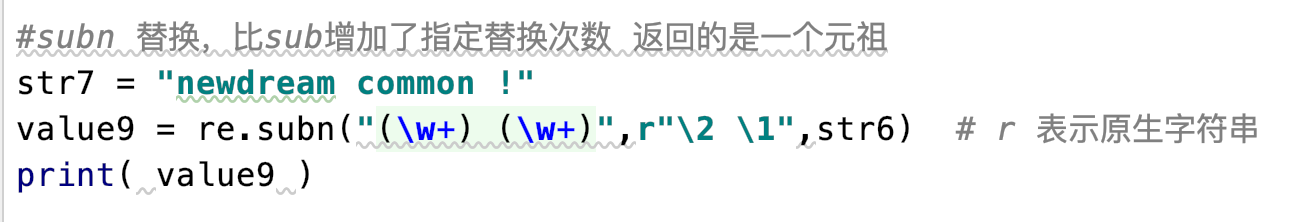
Group(): 一个括号表示1组



Sub()：替换功能



Subn()：替换功能 可以指定替换次数 返回的是元祖



正则表达式应用

扩展：爬虫程序 正则解析网站就是属于爬虫

线性接口自动化测试框架

Unittest + requests

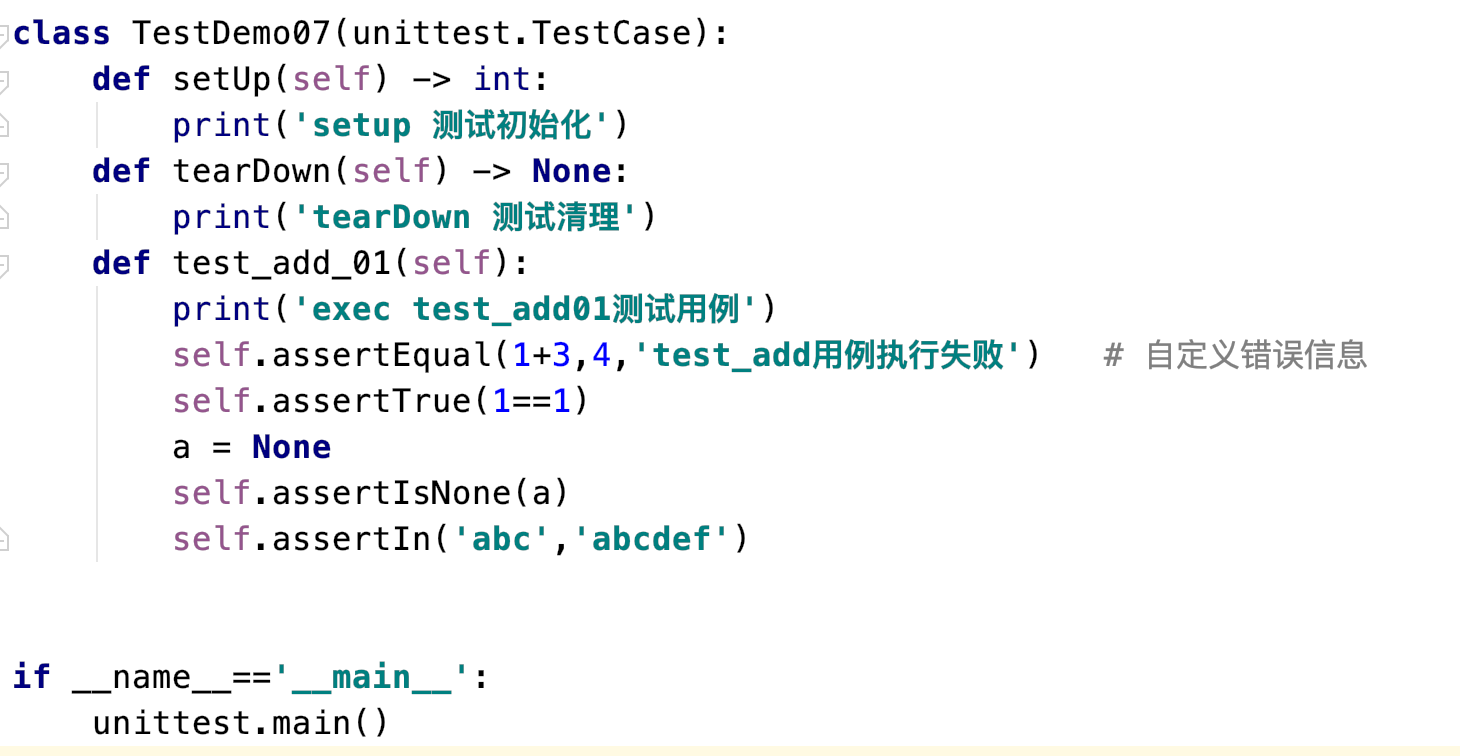
unittest内容

unittest是python内置的单元测试框架，具备编写用例、组织用例、执行用例、功能，可以结合selenium进行UI自动化测试，也可以结合appium、requests等模块做其它自动化测试。

使用unittest前需要了解该框架的五个概念:即test case,test suite,testLoader，test runner,test fixture。



Unittest的断言



assertEqual(a, b) a == b

assertNotEqual(a, b) a != b

assertTrue(x) bool(x) is True

assertFalse(x) bool(x) is False

assertIs(a, b) a is b

assertIsNot(a, b) a is not b

assertIsNone(x) x is None

assertIsNotNone(x) x is not None

assertIn(a, b) a in b

assertNotIn(a, b) a not in b

assertIsInstance(a, b) isinstance(a, b)

assertNotIsInstance(a, b) not isinstance(a, b)

assertGreater(a, b) a > b

assertGreaterEqual(a, b) a >= b

assertLess(a, b) a < b

assertLessEqual(a, b) a <= b

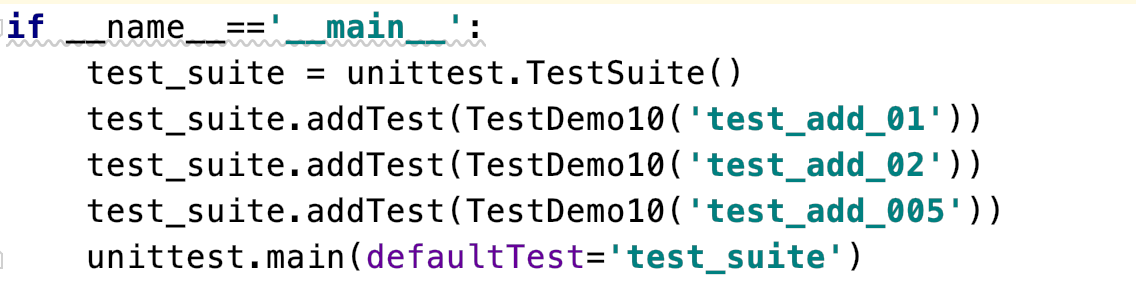
unittest用例执行顺序

两种方法设置：

第一种方法：通过默认的规则（字符串根据ascii码比较大小，升序执行）推荐第一种



第二种方法：suite()里面方法名 addTest()



忽略用例：让某些用例此次不执行

在执行测试脚本的时候，可能会有某几条用例本次不想执行，但又不想删也不想注释，unittest通过忽略部分测试用例不执行的方式，分无条件忽略和有条件忽略,通过装饰器实现所描述的场景。提供的装饰器如下：

@unittest.skip(reason)：强制跳转。reason是跳转原因

@unittest.skipIf(condition, reason)：condition为True的时候跳转@unittest.skipUnless(condition, reason)：condition为False的时候跳转@unittest.expectedFailure：标记该测试预期为失败，如果该测试方法运行失败，则该测试不算做失败



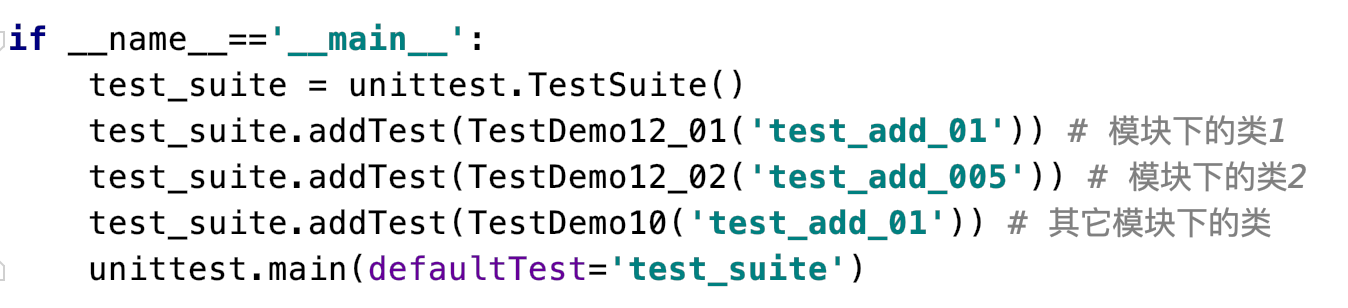
构建测试套件

在实际项目中，随着项目进度的开展，测试类会越来越多，可是直到现在我们还只会一个一个的单独运行测试类，这在实际项目实践中肯定是不可行的，在unittest中可以通过测试套件来解决该问题。

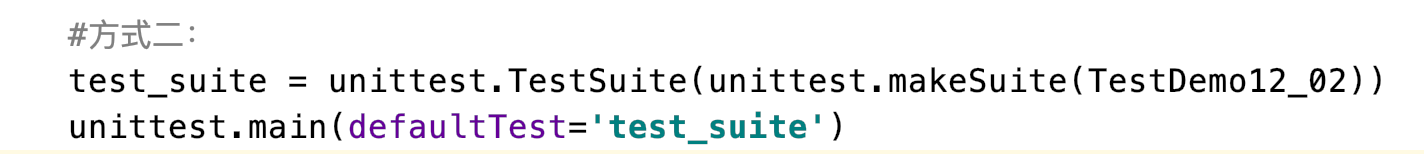
测试套件（Test Suite）是由多个测试用例（Test Case）组成的，当然也可以由多个子测试套件组成。

方式一：

用unittest.TestSuite()实例化测试套件对象后，内部的addTest()方法对测试类内部的测试案例进行逐一添加



方式二：



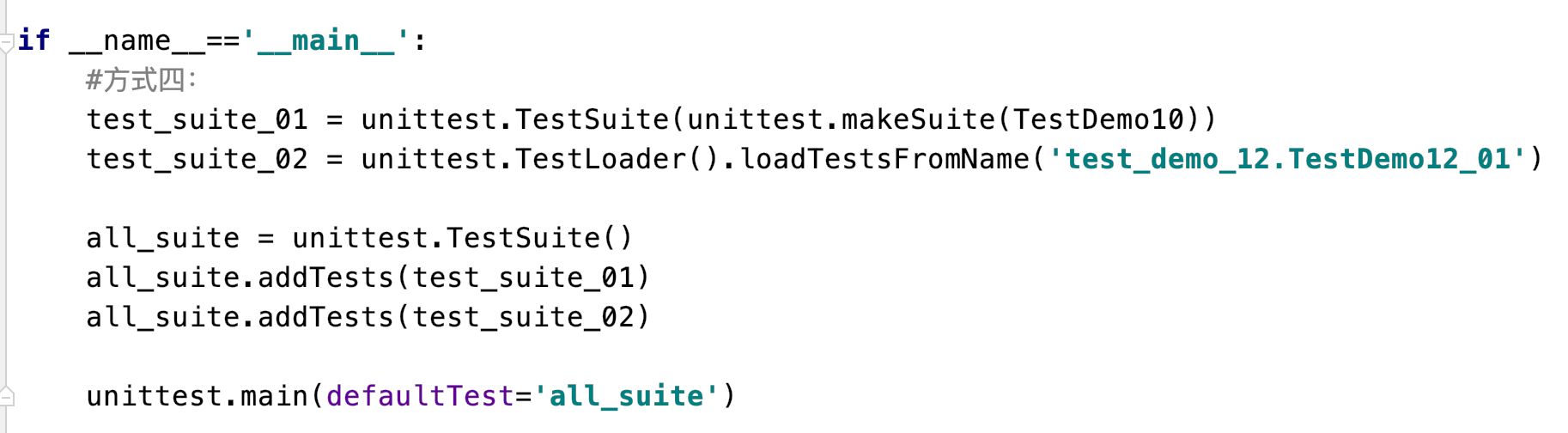
方式三：

unittest提供一个TestLoader类用于自动创建一个测试集并把单个测试放入到测试集中。TestLoader自动运行以test开头的测试方法。



方式四：

用unittest.TestSuite()实例化测试套件对象后，内部的addTests()方法可以把多个子测试集合进行整合到一个大的测试集合中



方式五：

方法如下：

discover（start\_dir, pattern ='test \*.py', top\_level\_dir = None ）

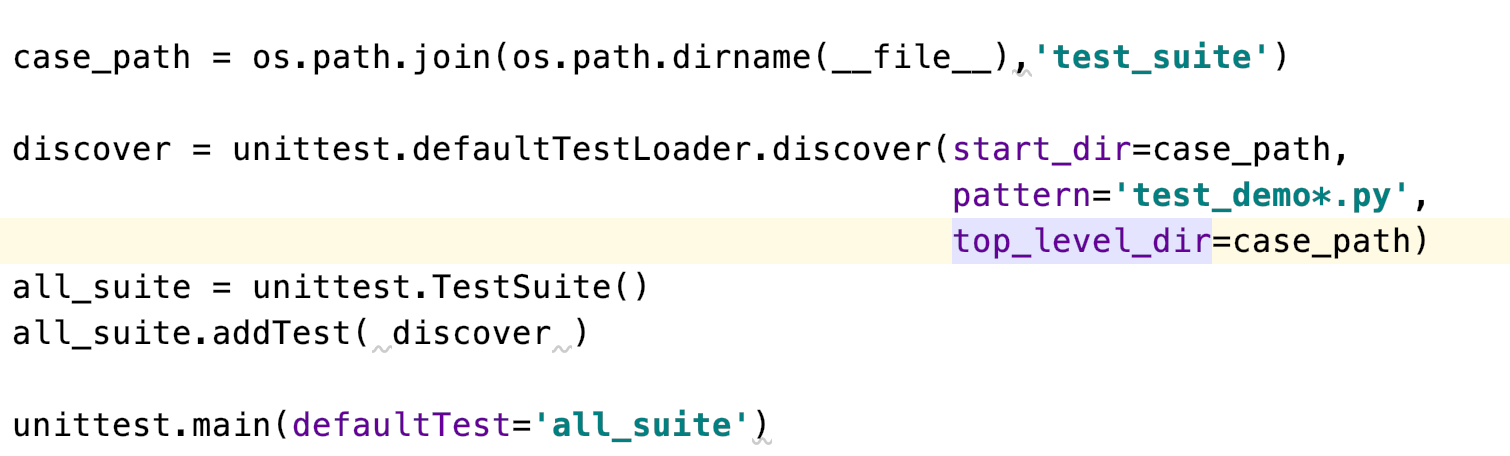
start\_dir：要测试的模块名或测试用例目录；

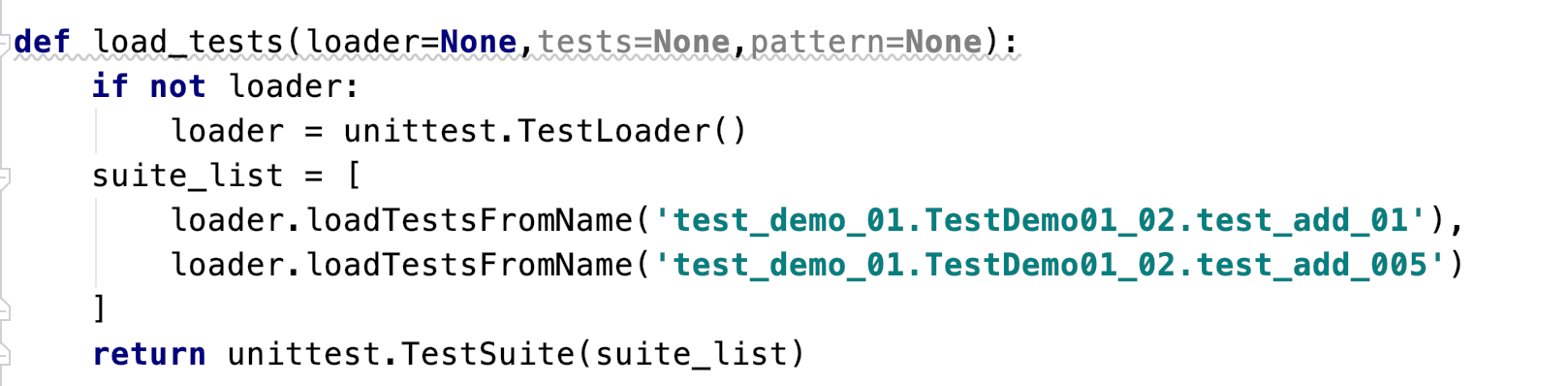
pattern=‘test\*.py’：表示用例文件名的匹配原则，例子中匹配文件名为以“test”开头的“.py”文件，星号“\*”表示任意多个字符；

top\_level\_dir=None：测试模块的顶层目录，如果没有顶层目录，默认为None。

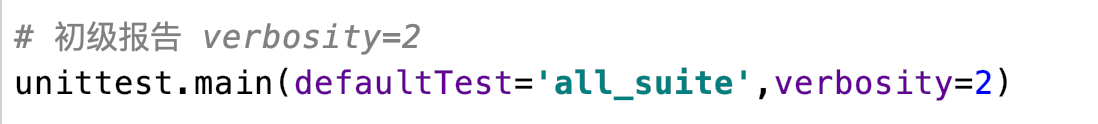
该方法通过从指定的开始目录递归到子目录中查找所有测试模块，并返回包含它们的TestSuite对象，只要与模式匹配测试文件和可导入的模块名称才会被加载。

如果一个测试文件的名称符合pattern，将检查该文件是否包含 load\_tests() 函数，如果 load\_tests() 函数存在，则由该函数负责加载本文件中的测试用例。如果不存在，就会执行loadTestsFromModule()，查找该文件中派生自TestCase 的类包含的 test 开头的方法。





unittest测试结果展示



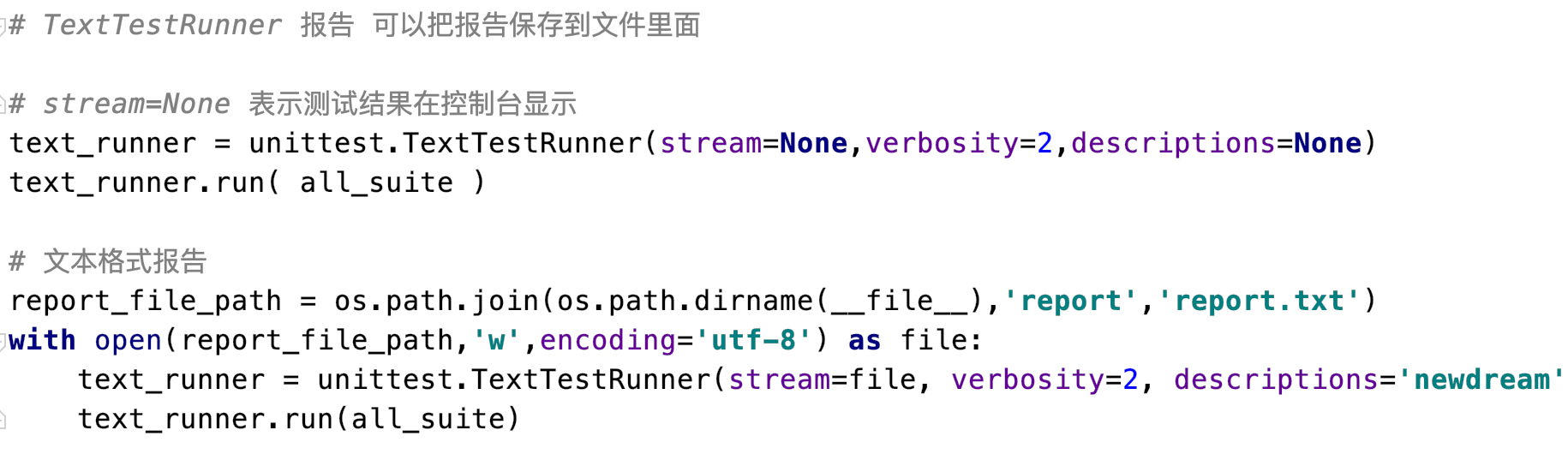
这里的verbosity是一个选项,表示测试结果的信息复杂度，有三个值：

0 (静默模式): 你只能获得总的测试用例数和总的结果 比如 总共100个 失败20 成功80

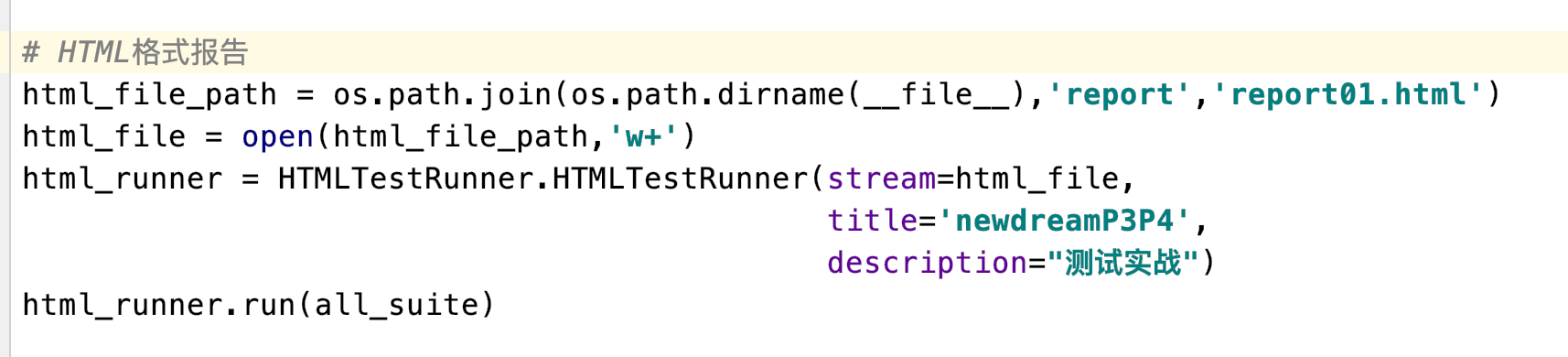
1 (默认模式): 非常类似静默模式 只是在每个成功的用例前面有个“.”

每个失败的用例前面有个 “F”

2 (详细模式):测试结果会显示每个测试用例的所有相关的信息



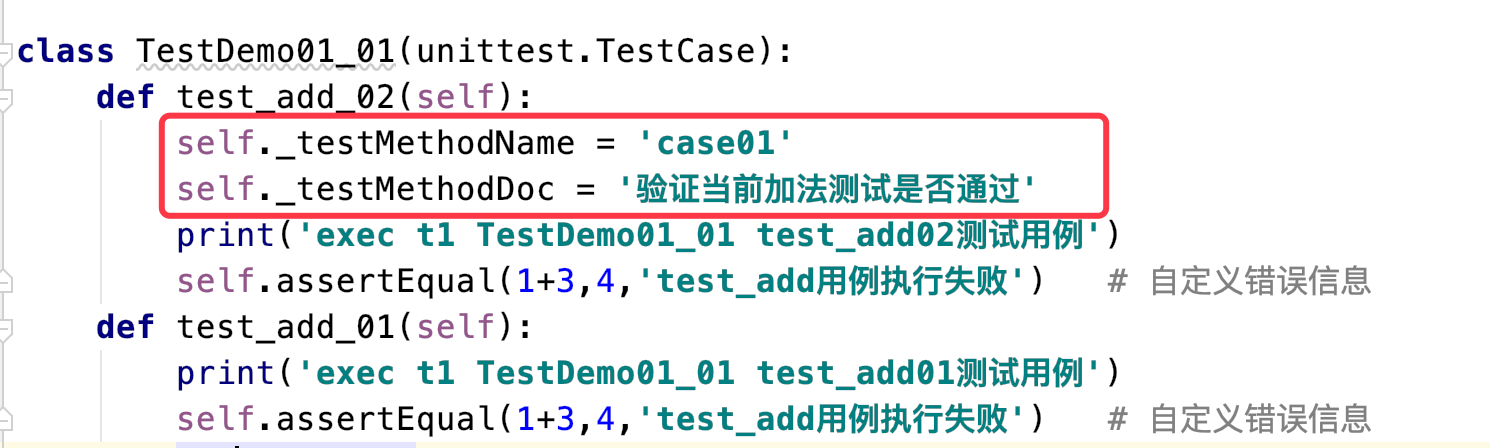
网页格式的报告 需要导入 HTMLTestRunner.py



网页格式报告二： HTMLTestReportCN







带报表的线性接口测试框架

