今天的学习内容包括：python的类、模块、collection、代码风格、装饰器、迭代器以及一些Django框架的使用入门

创建一个类

class 类名(继承自):

python 支持多继承

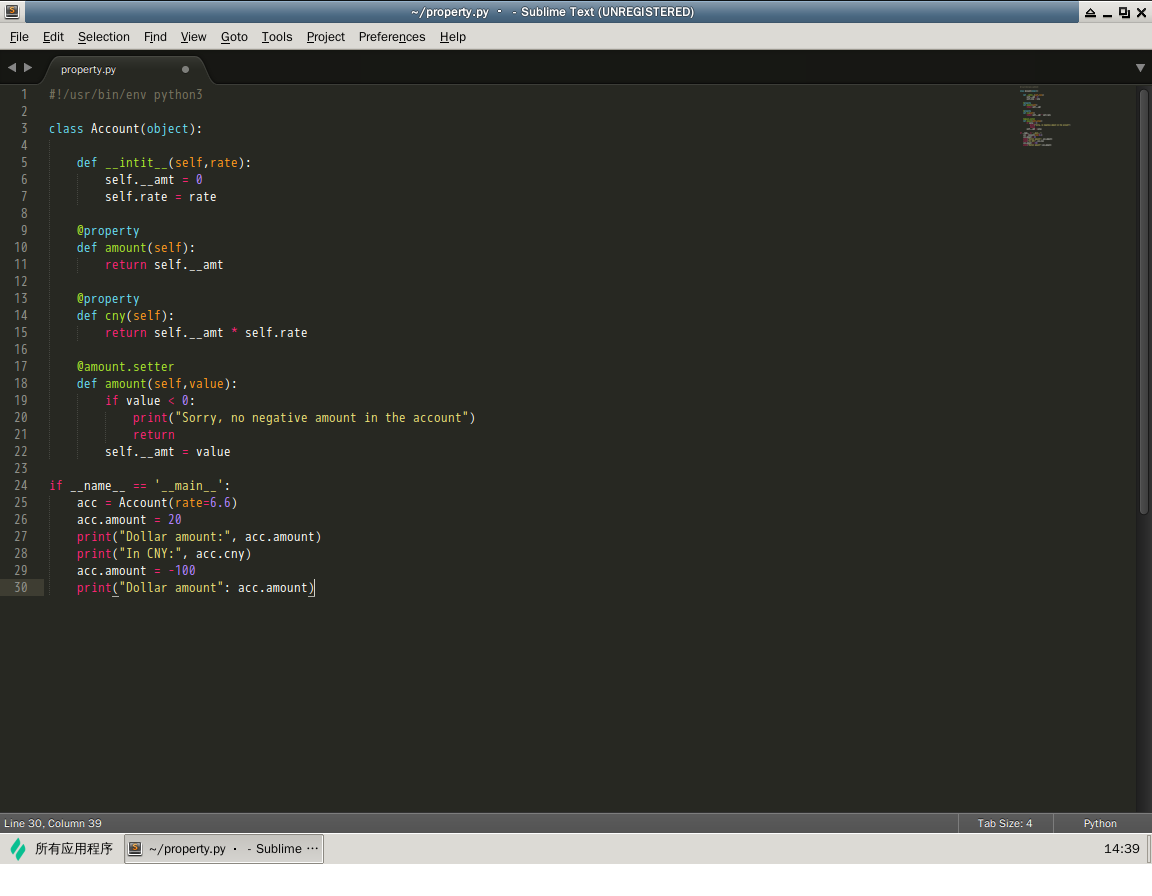
构造函数 def **init**(self,pa,pb):

注意这里需要传入一个self

创建实例 类名(参数)

可以用关键字del 来删除一个实例，也可以等垃圾回收机制来回收它

装饰器 这部分实验楼没细讲，只是给了一个实例



利用装饰器 @property和@参数.setter 实现了java中的getter和setter

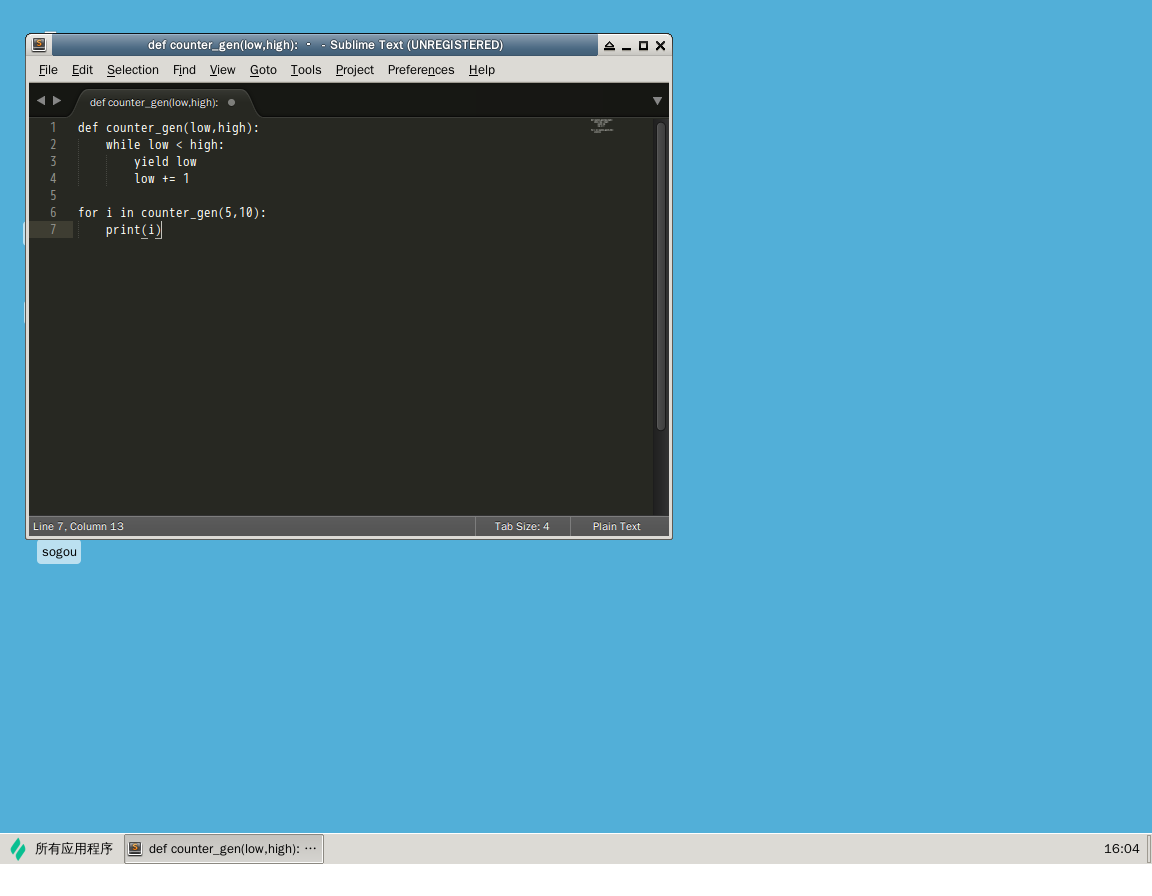
疑问： \_\_开头的参数是私有的？

迭代： 一个类要支持迭代，需要实现以下两个函数：

 \_\_**iter\_\_**()，返回迭代器对象自身。这用在 for 和 in 语句中。

**\_\_next\_\_**()，返回迭代器的下一个值。如果没有下一个值可以返回，那么应该抛出 StopIteration 异常。

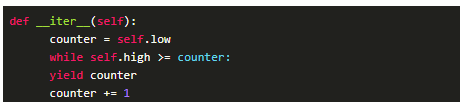
生成器： 这通过在函数中使用 yield 关键字完成 貌似用法与与迭代一致？



在 While 循环中，每当执行到 yield 语句时，返回变量 low 的值并且生成器状态转为挂起。在下一次调用生成器时，生成器从之前冻结的地方恢复执行然后变量 low 的值增一。生成器继续 while 循环并且再次来到 yield 语句...

**生成器是一定不能重复循环的**

**记录一个特殊用法，在iter中的循环语句中使用yield，而不写next方法，这样的迭代器可以重复使用**

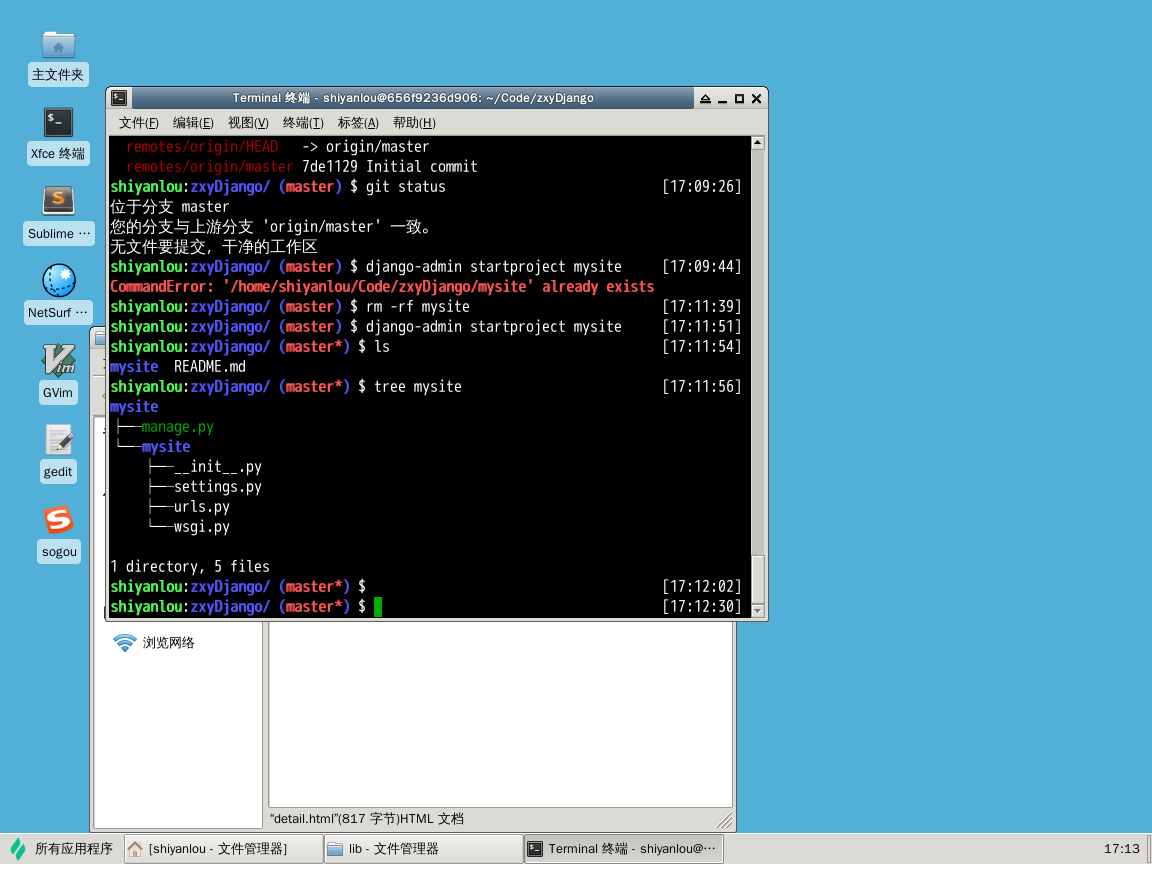


**创建Django项目**

在目标文件夹下执行

django-admin startproject mysite

自动生成了一个基础架构



在mysite下执行

python3 manage.py runserver 运行网站

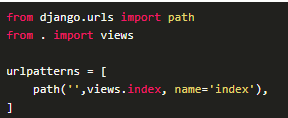
ctrl+c 结束进程

**创建一个可被访问的url**

1.在mysite下执行python3 manage.py startapp lib生成lib文件夹

2.修改lib/views.py 文件 写一个准备被访问的函数，函数通常接受一个httprequest

3.创建lib/urls.py 文件





4.修改mysite/urls.py 添加 path（‘lib/’，include（‘lib.urls’））该函数截取'lib/'的部分，剩下的交给lib.urls

5.python3 manage.py runserver

http://github.com/z503755743/zxyDjango.git