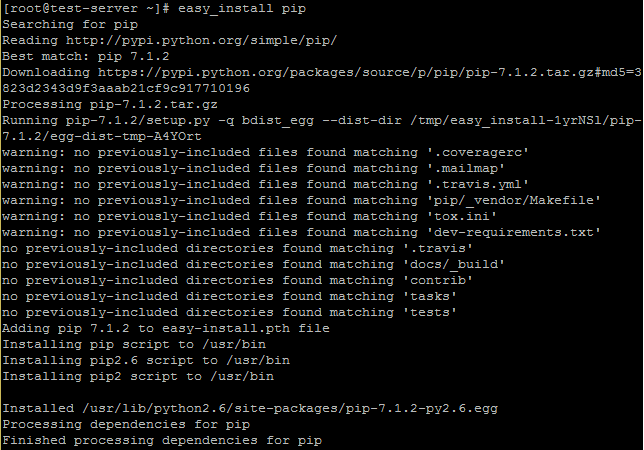
**pip使用**

pip 是一个安装和管理 Python 包的工具 , pip的安装需要setuptools 或者 distribute，distribute是setuptools的取代(Setuptools包后期不再维护了)。

安装pip

easy\_install pip



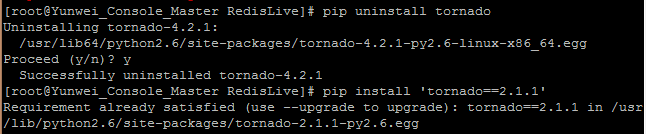
安装特定版本的package，通过使用==, >=, <=, >, <来指定一个版本号。

pip install 'tornado==2.1.1'

升级包到当前最新的版本，可以使用-U 或者 –upgrade pip install -U 包名称

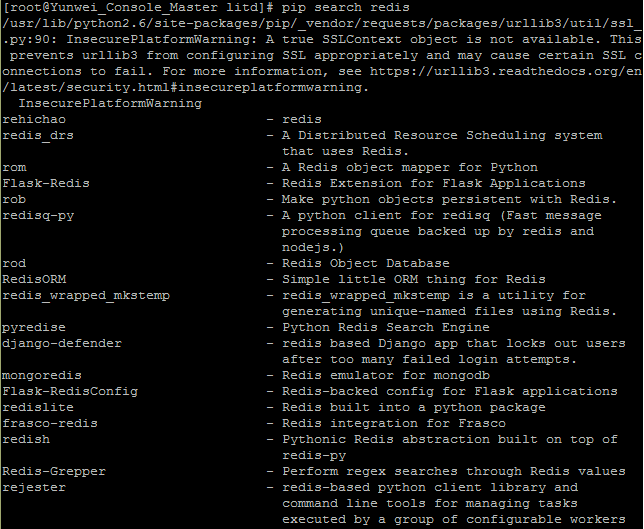
pip install -U python-dateutil

卸载包pip uninstall 包名称

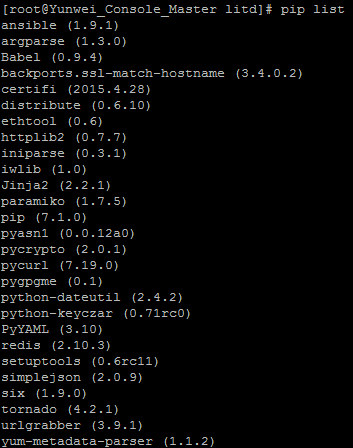


查询包pip search "包名称"

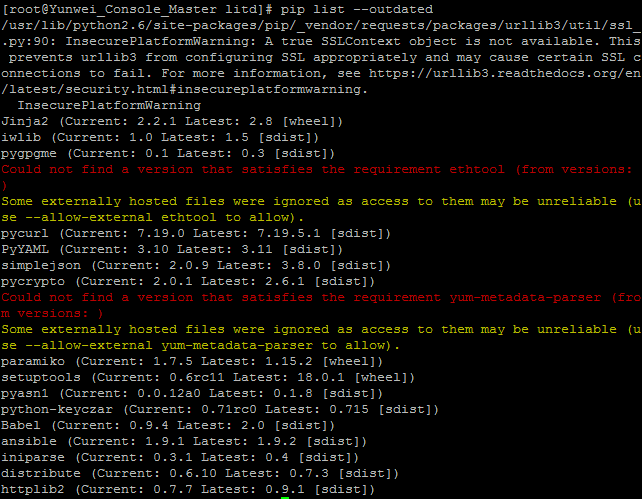
pip search redis



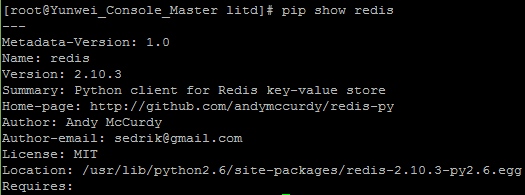
列出安装的packages pip freeze或者pip list



获取过期的库：pip list –outdated



查看某个库的信息：pip show redis



**psutil**

psutil是python实现ps、netstat的shell命令的模块，使用它很简单，yum安装的版本过低(0.6.1)，没有pids()等函数，源代码编译安装较高版本psutil-2.2.1

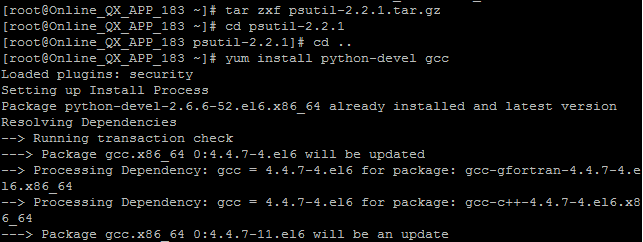
下载链接：https://pypi.python.org/packages/source/p/psutil/psutil-2.2.1.tar.gz

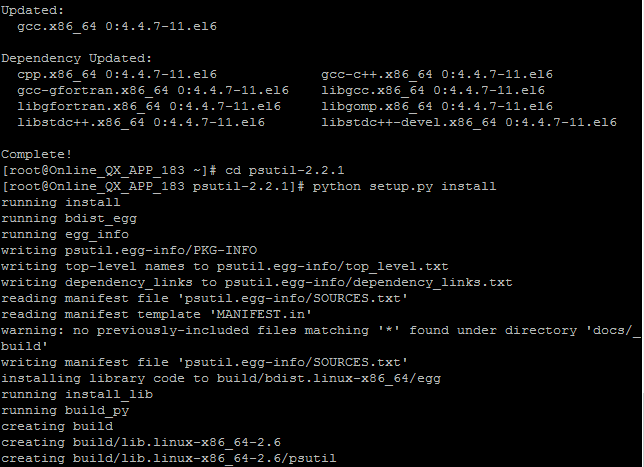
tar zxf psutil-2.2.1.tar.gz

yum install python-devel gcc -y

cd psutil-2.2.1

python setup.py install

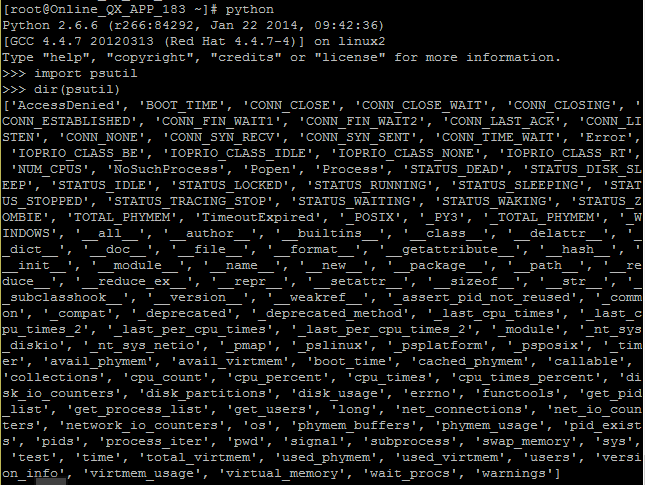




查看psutil所有的函数

import psutil

dir(psutil)

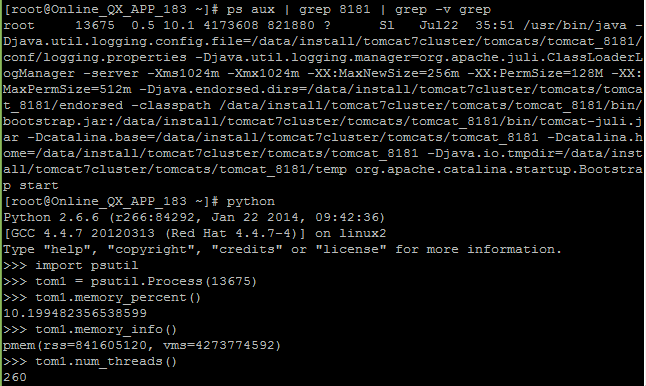


查看JVM的内存使用、threads数量

psutil.Process(pid) ：实例化一个进程

psutil.Process(pid).memory\_percent() ：进程的内存利用率

psutil.Process(pid).num\_threads() ：进程的线程数



**pyinotify**

什么是inotify：

Inotify是一个事件驱动的通知机制，Inotify 提供一个简单的API，使用最小的文件描述符，并且允许细粒度监控。与 inotify 的通信是通过系统调用实现。可用的函数如下所示：

inotify\_init 是用于创建一个inotify实例的系统调用，并返回一个指向该实例的文件描述符。

inotify\_init1 与inotify\_init相似，并带有附加标志。如果这些附加标志没有指定，将采用与inotify\_init相同的值。

inotify\_add\_watch 增加对文件或者目录的监控，并指定需要监控哪些事件。标志用于控制是否将事件添加到已有的监控中，是否只有路径代表一个目录才进行监控，是否要追踪符号链接，是否进行一次性监控，当首次事件出现后就停止监控。

inotify\_rm\_watch 从监控列表中移出监控项目。

read 读取包含一个或者多个事件信息的缓存。

close 关闭文件描述符，并且移除所有在该描述符上的所有监控。当关于某实例的所有文件描述符都关闭时，资源和下层对象都将释放，以供内核再次使用。

因此，典型的监控程序需要进行如下操作：

使用 inotify\_init 打开一个文件描述符

添加一个或者多个监控

等待事件

处理事件，然后返回并等待更多事件

当监控不再活动时，或者接到某个信号之后，关闭文件描述符，清空，然后退出。

Inotify 可以监视的文件系统事件包括：

IN\_ACCESS，即文件被访问

IN\_MODIFY，文件被write

IN\_ATTRIB，文件属性被修改，如chmod、chown、touch等

IN\_CLOSE\_WRITE，可写文件被close

IN\_CLOSE\_NOWRITE，不可写文件被close

IN\_OPEN，文件被open

IN\_MOVED\_FROM，文件被移走,如mv

IN\_MOVED\_TO，文件被移来，如mv、cp

IN\_CREATE，创建新文件

IN\_DELETE，文件被删除，如rm

IN\_DELETE\_SELF，自删除，即一个可执行文件在执行时删除自己

IN\_MOVE\_SELF，自移动，即一个可执行文件在执行时移动自己

IN\_UNMOUNT，宿主文件系统被umount

IN\_CLOSE，文件被关闭，等同于(IN\_CLOSE\_WRITE | IN\_CLOSE\_NOWRITE)

IN\_MOVE，文件被移动，等同于(IN\_MOVED\_FROM | IN\_MOVED\_TO)

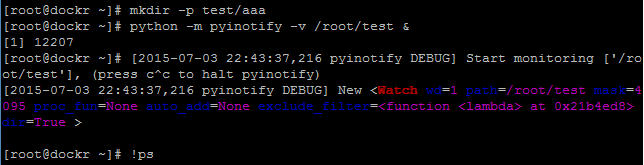
安装pyinotify

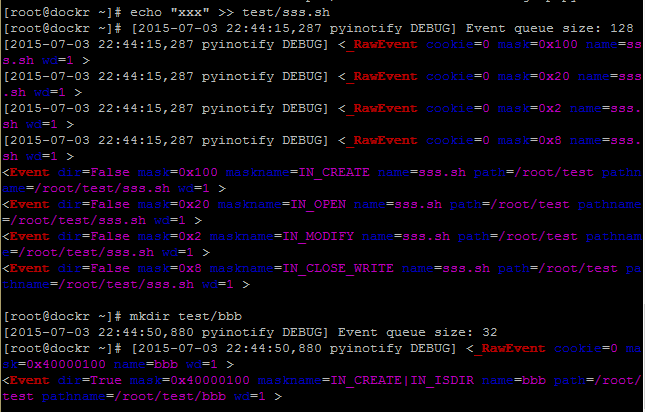
yum install python-pip

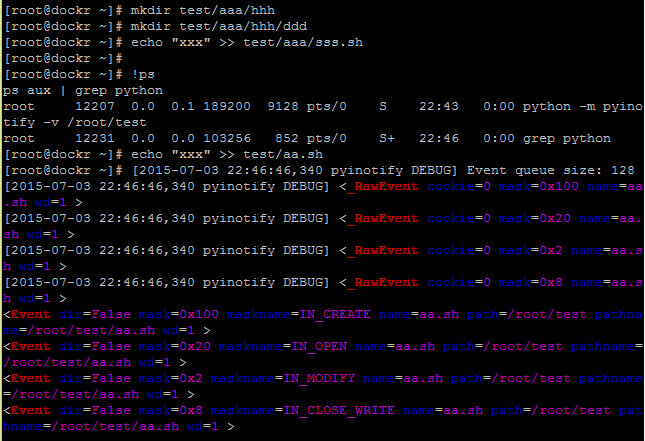
pip install pyinotify

pyinotify初试

python -m pyinotify -v /root/test监测/root/test目录的变化





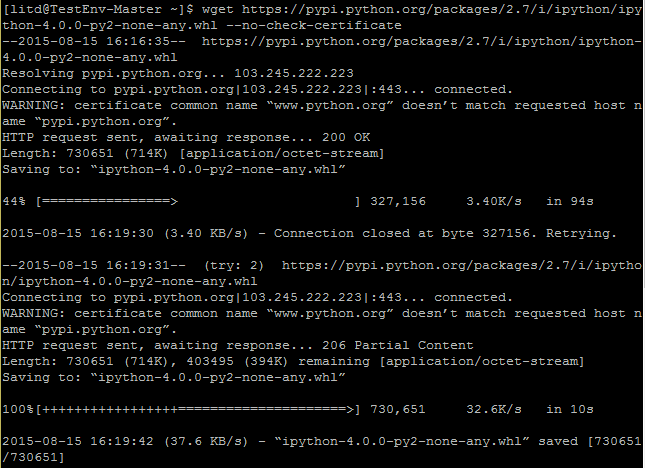


只记录了对/root/test目录的操作，没有记录/root/test下级目录的变化

ipython安装与使用

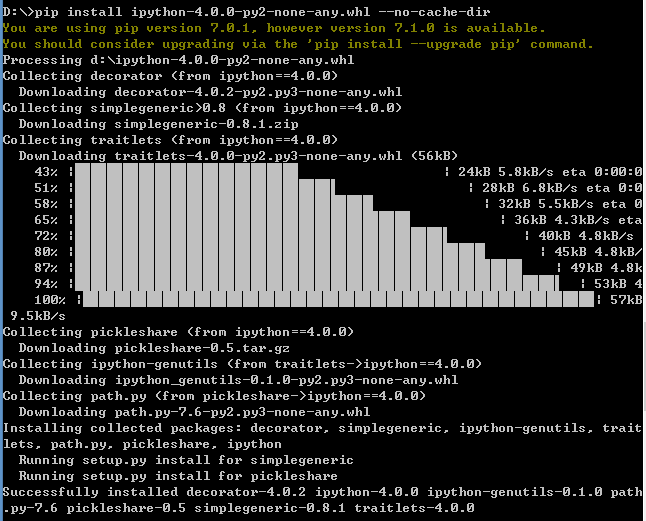
在cmd直接使用pip install ipython安装是简单的做法，但常常网速导致time out ，因此可以在Linux下载ipython-4.0.0-py2-none-any.whl后

wget https://pypi.python.org/packages/2.7/i/ipython/ipython-4.0.0-py2-none-any.whl --no-check-certificate

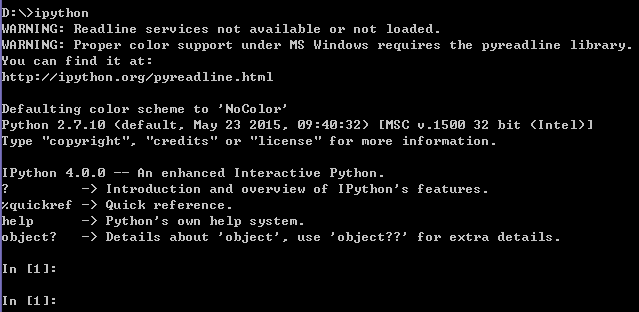


在D盘的根目录，执行

pip install ipython-4.0.0-py2-none-any.whl --no-cache-dir



安装完成

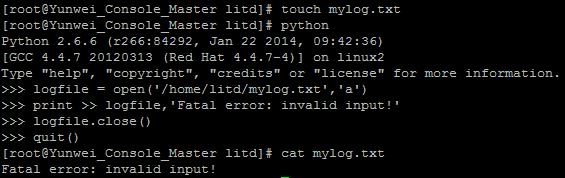


Print保存到文件

logfile = open('/home/litd/mylog.txt','a')

print >> logfile,'Fatal error: invalid input!'

logfile.close()



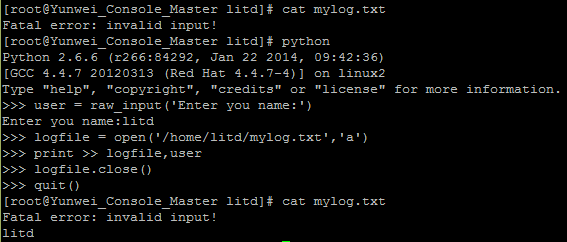
使用raw\_input()捕获输入，然后追加到文件

user = raw\_input('Enter you name:')

logfile = open('/home/litd/mylog.txt','a')

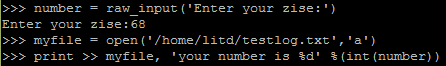
print >> logfile,user

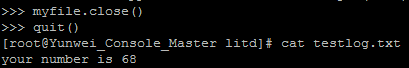
logfile.close()



raw\_input()只能捕获文本输入，对于数字还需要用int()转换为整数类型

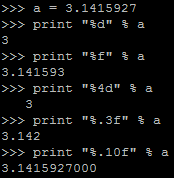
print >> myfile, 'your number is %d' %(int(number))





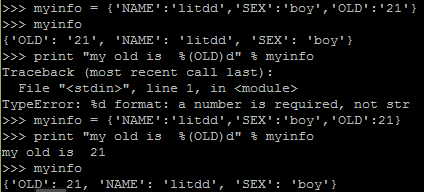
%d %s %f %r

其中，%s和%d就是占位符，%d只能输出整数,int类型；%f输出浮点数而且可以按照要求输出小数位数，如果要求的小数位数过多，后面就用0补全。

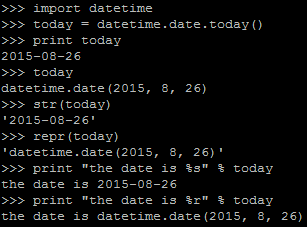


%d可以格式化列表

print "my old is %(OLD)d" % myinfo



不管是%r还是%s(%d)都是把做为整数的对象转化为字符串输出了，而不是输出整数。但是%r和%s是有点区别的， %s调用的是str()函数把对象转化为str类型，而%r是调用了repr()将对象转化为字符串。



承接变量的还有{ }.format( )

print "my name is {0} and i am {2}".format("litdd","boy",21)



print "my name is {NAME}".format(NAME="litdd")



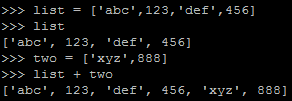
Python 数据类型

Python有五个标准的数据类型：

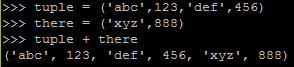
• 数字

• 字符串

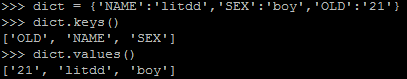
• 列表



• 元组

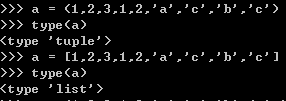


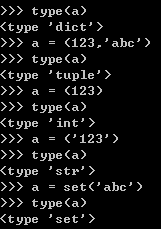
• 字典



列表和元组之间的主要区别是：列表括在括号（[]）和它们的元素和大小是可以改变的，而元组在圆括号()，不能被更新。元组可以被认为是只读列表。

Python字典是一种哈希表型，由键 - 值对组成。字典键几乎可以是任何Python类型，但通常是数字或字符串。值可以是任意Python的对象。字典是由花括号括号（{}），可分配值，并用方括号（[]）访问。



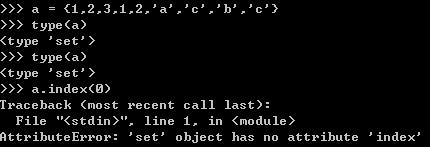


{ }也是set

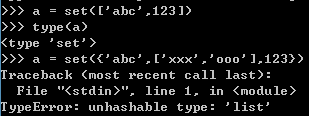


set

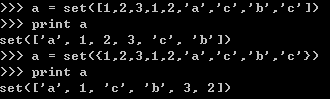
set不是python的一种数据类型，是一个无序不重复元素集，不记录元素位置或者插入点



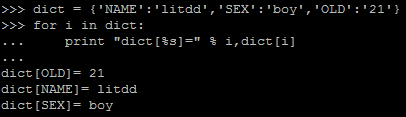
不能包含list



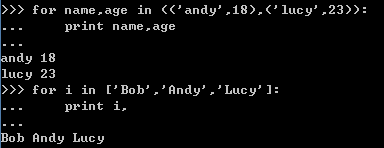
剔除list、set中的重复值



dict遍历

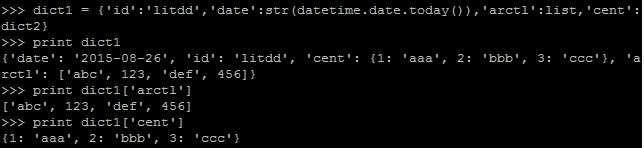


元祖遍历



dict包含list和dict

字典的value可以是list，可以是dict



Python支持四种不同的数值类型：

• int (有符号整数)

• long (长整数[也可以以八进制和十六进制表示])

• float (浮点实数值)

• complex (复数)

数据类型转换

|  |  |
| --- | --- |
| **函数** | **描述** |
| int(x [,base]) | 将x转换为一个整数。基数指定为base，如果x是一个字符串。 |
| long(x [,base] ) | 将x转换为一个长整数。基数指定为base，如果x是一个字符串。 |
| float(x) | 将x转换到一个浮点数。 |
| complex(real [,imag]) | 创建一个复数。 |
| str(x) | 转换对象x为字符串表示形式。 |
| repr(x) | 对象x转换为一个表达式字符串。 |
| eval(str) | 计算一个字符串，并返回一个对象。 |
| tuple(s) | 把s转换为一个元组。 |
| list(s) | 把s转换为一个列表。 |
| set(s) | 把s转换为一个集合。 |
| dict(d) | 创建一个字典。 d必须的（键，值）元组序列。 |
| frozenset(s) | 把s转换为冻结集。 |
| chr(x) | 整数转换为一个字符。 |
| unichr(x) | 整数转换为一个Unicode字符。 |
| ord(x) | 转换单个字符为整数值。 |
| hex(x) | 将整数转换为十六进制字符串。 |
| oct(x) | 将整数转换为以八进制的字符串。 |

list取偶

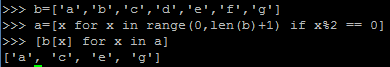
我有一个list，怎么取出其中的偶数列呢？有两个方法：

1. list[x] for x in 偶数list

b=['a','b','c','d','e','f','g']

a=[x for x in range(0,len(b)+1) if x%2 == 0]

[b[x] for x in a]



2）[x for x in list if list.index(x)%2 == 0]



stirng和list的相互转换

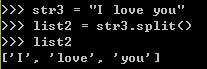
string数据类型的split方法可以将string转换为list

str3 = "I love you"

list2 = str3.split()

list2

['I', 'love', 'you']



特殊情况也可以

str1 = '''I Love

... "python"

... '''

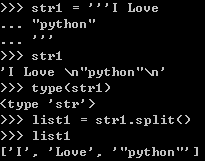
str1

'I Love \n"python"\n'

list1 = str1.split()

list1

['I', 'Love', '"python"']



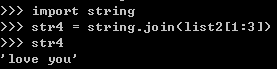
string模块的join方法可以将list拼接为string

>>> import string

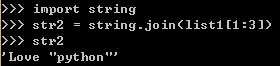
>>> str4 = string.join(list2[1:3])

>>> str4

'love you'

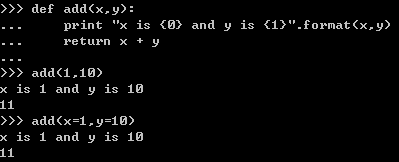


复杂的也没关系



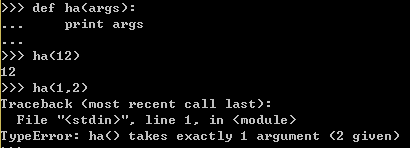
函数参数

两个参数

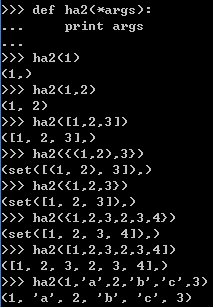


args、\*args、\*\*args

args不能接受除string以外的字符串

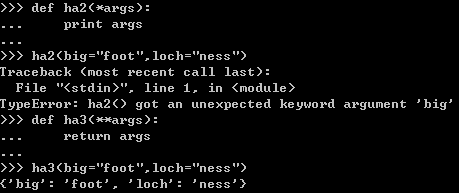


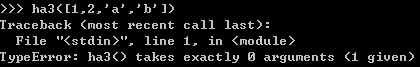
\*args接受list、set、tuple但是不能接受dict





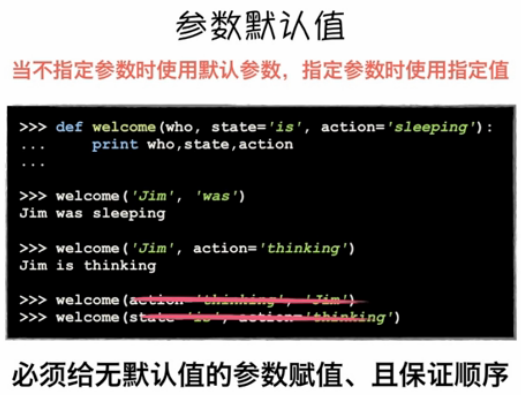
\*\*args可以生成dict，但是不接受list和dict





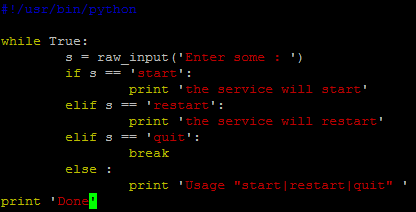


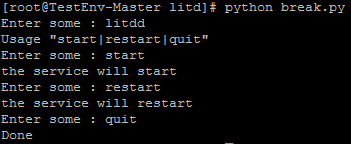




while和for循环

while循环接受start、restart、quit这三个参数，捕捉到quit参数立即退出





同时while循环也支持break、continue

i = 0

while (i < 10):

if i == 4:

i = i + 1

break;

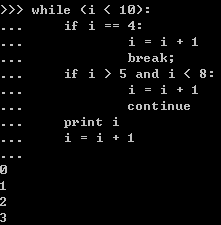
if i > 5 and i < 8:

i = i + 1

continue

print i

i = i + 1



i = 0

while (i < 10):

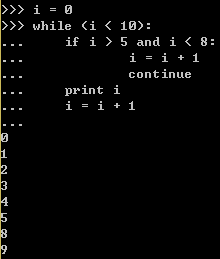
if i > 5 and i < 8:

i = i + 1

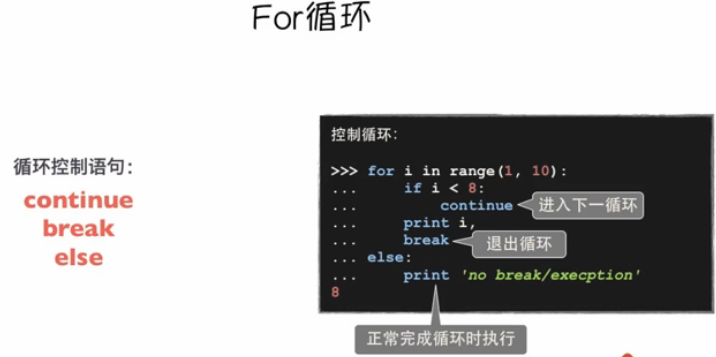
continue

print i

i = i + 1



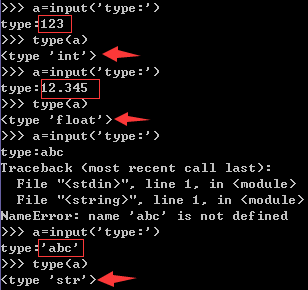
for循环的扩展语句

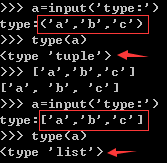


http://edu.51cto.com/lesson/id-26506.html

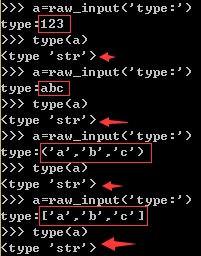
input、rwa\_input

input不改变所接受的标准数据类型





rwa\_input将接受的所有数据类型转换为sring，并且接受标准非标准数据



装饰器

装饰器的作用就是在函数调用方法不变的情况下，把此函数包装成另一个函数来使用

实例代码：

#-\*- coding: utf-8 -\*-

def auth(func):

def wrapper(\*args,\*\*kwargs):

user = raw\_input("input\_passwd: ")

if user == 'admin':

print "welcome login."

return func(\*args,\*\*kwargs)

else:

print "--wrong passwd,access denied---"

return wrapper

#使用语法糖@来装饰函数,运行task前会先调auth

@auth

def task(name,age=33):

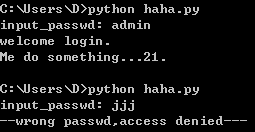
print "%s do something...%s." % (name,age)

task("Me",21)



运行结果：





迭代器

iter()方法生成迭代器

obj = range(6)

itor = iter(obj)

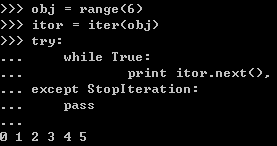
try:

while True:

print itor.next(),

except StopIteration:

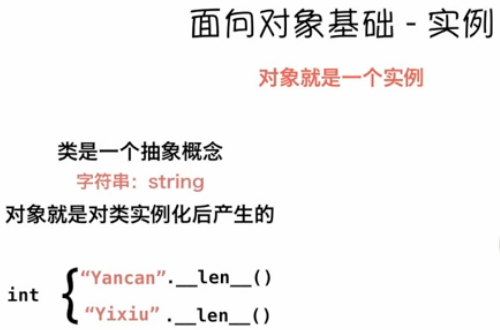
pass

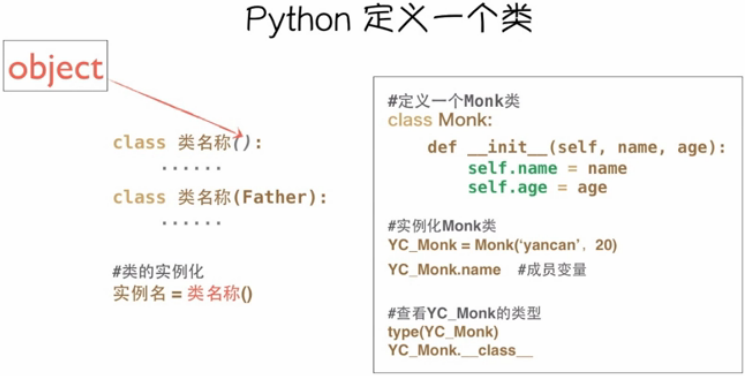


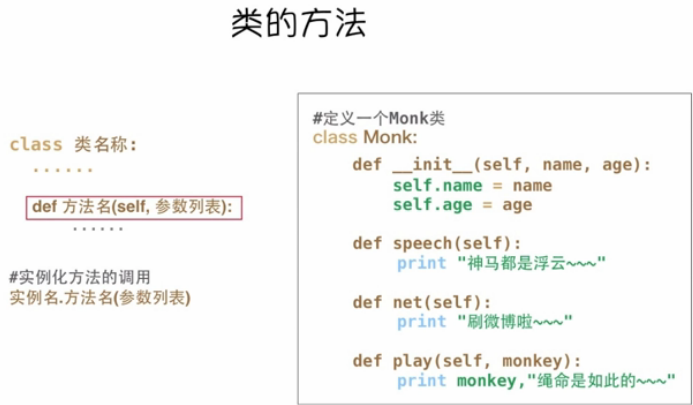


类

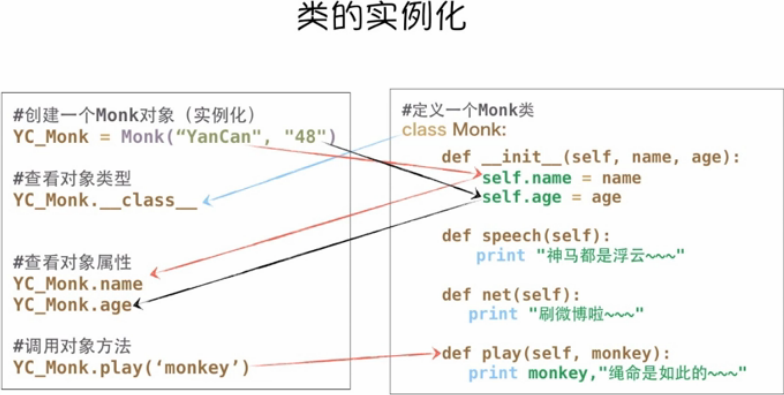








类的实例化



class test:

def \_\_init\_\_(self,name,age):

self.name = name

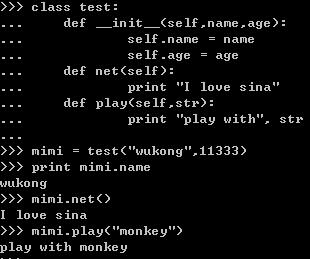
self.age = age

def net(self):

print "I love sina"

def play(self,str):

print "play with", str



IO基础

其实是file类



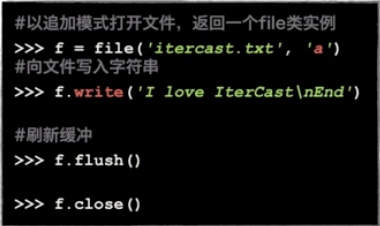
Open函数-读文件



read() readline() readlines()



file类-写文件



实战

交互式输入保存到文件，并用换行符代替enter键；输入exit时就退出

f = file('testfile.txt','a')

while True:

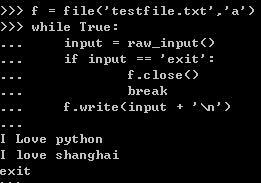
input = raw\_input()

if input == 'exit':

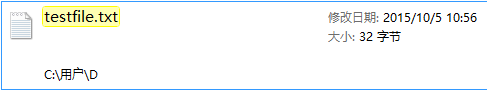
f.close()

break

f.write(input + '\n')



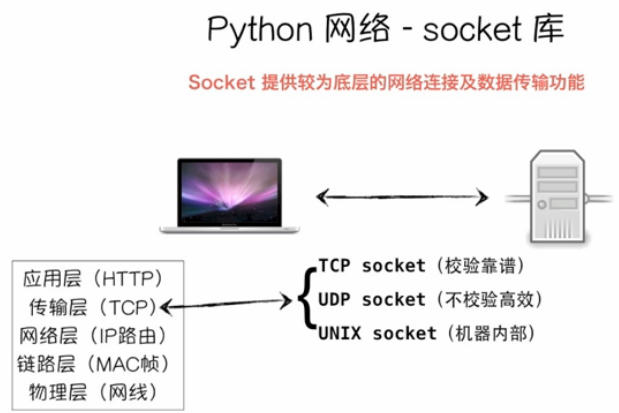
文件在



文件的内容



socket







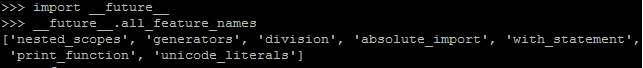


HTTP

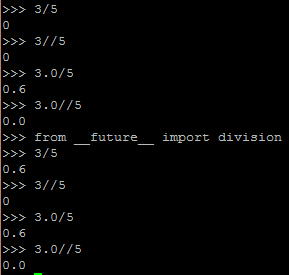




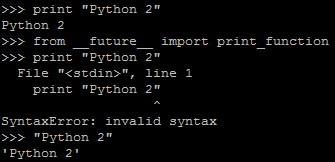
**\_\_future\_\_模块**



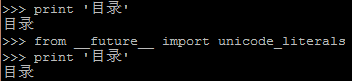
division 新的除法特性，本来的除号`/`对于分子分母是整数的情况会取整，但新特性中在此情况下的除法不会取整，取整的使用`//`。如下可见，只有分子分母都是整数时结果不同。



print\_function 新的print是一个函数，如果导入此特性，之前的print语句就不能用了。



unicode\_literals 这个是对字符串使用unicode字符。如果系统默认字符编码集是UTF8，该功能就不需要



absolute\_import 这个没有搞懂，如果我当前目录有一个sys.py，我用的时候总是会调用系统的sys。如果当前目录和sys.path的路径中都有一个foo，则都会调用当前目录下的foo。另外，我运行的方式都是 python filename.py的方式，而如果使用python -c "import filename"则又是另一种答案。这个问题还没有解决，搜到的最详细的讨论见http://bytes.com/topic/python/answers/596703-\_\_future\_\_-import-absolute\_import

nested\_scopes 这个是修改嵌套函数或lambda函数中变量的搜索顺序，从`当前函数命名空间->模块命名空间`的顺序更改为了`当前函数命名空间->父函数命名空间->模块命名空间`,python2.7.5中默认使用

generators 生成器，对应yield的语法，python2.7.5中默认使用

with\_statement 是使用with关键字，python2.7.5是默认使用

首先是可以做个性化的用法，比如你喜欢用print（）而不是print

\_\_future\_\_的模块可以让python2和python3有良好的兼容性，常用的语句有以下：

from \_\_future\_\_ import print\_function

from \_\_future\_\_ import unicode\_literals

from \_\_future\_\_ import division

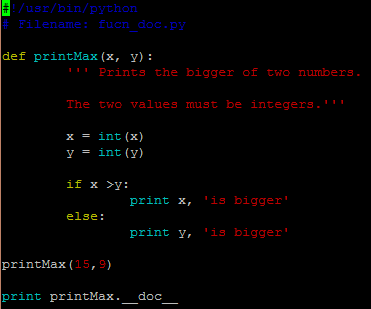
from \_\_future\_\_ import absolute\_import

**\_\_doc\_\_**

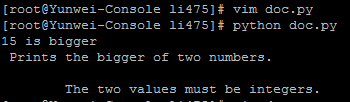
Python的一个非常奇妙的特性，文档字符串，简称docstrings。DocStrings 是一个重要的工具，它可以使程序文档更加简单易懂。

文档字符串的惯用法：一个多行的字符串，它的首行以大写字母开始，句号结尾；第二行是空行；第三行开始是详细的描述内容

\_\_doc\_\_是函数的注释部分，用’’’引起来的部分



执行效果



**os模块**

os.listdir(dirname)：列出dirname下的目录和文件

os.getcwd()：获得当前工作目录

os.curdir:返回当前目录('.')

os.chdir(dirname):改变工作目录到dirname

os.path.isdir(name):判断name是不是一个目录，name不是目录就返回false

os.path.isfile(name):判断name是不是一个文件，不存在name也返回false

os.path.exists(name):判断是否存在文件或目录name

os.path.getsize(name):获得文件大小，如果name是目录返回0L

os.path.abspath(name):获得绝对路径

os.path.normpath(path):规范path字符串形式

os.path.split(name):分割文件名与目录 （事实上，如果你完全使用目录，它也会将最后>一个目录作为文件名而分离，同时它不会判断文件或目录是否存在）

os.path.splitext():分离文件名与扩展名

os.path.join(path,name):连接目录与文件名或目录

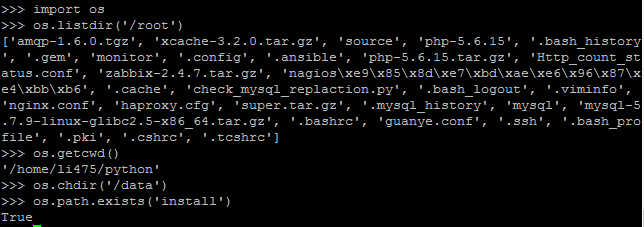
os.path.basename(path):返回文件名

os.path.dirname(path):返回文件路径

os.remove(dir) #dir为要删除的文件夹或者文件路径

os.rmdir(path) #path要删除的目录的路径。需要说明的是，使用os.rmdir删除的目录必须

为空目录，否则函数出错。

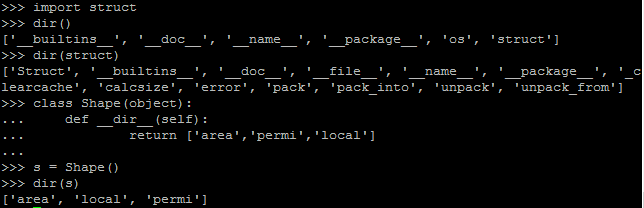


**dir()**

中文说明：不带参数时，返回当前范围内的变量、方法和定义的类型列表；带参数时，返回参数的属性、方法列表。如果参数包含方法\_\_dir\_\_()，该方法将被调用。如果参数不包含\_\_dir\_\_()，该方法将最大限度地收集参数信息。

参数object： 对象、变量、类型。

版本：该函数在python各个版本中都有，但是每个版本中显示的属性细节有所不同。使用时注意区别。



**PyUnit模块**

为了让单元测试代码能够被测试和维护人员更容易地理解，最好的解决办法是让开发人员遵循一定的规范来编写用于测试的代码，具体到Python程序员来讲，则是要采用 PyUnit这一自动测试框架来构造单元测试用例。目前PyUnit已经得到了大多数Python开发人员的认可，成了事实上的Python单元测试标准。

#!/usr/bin/python

#--coding=utf-8--

from widget import Widget

import unittest

#执行测试的类

class WidgetTestCase(unittest.TestCase):

def setUP(self):

self.widget = widget()

def tearDown(self):

self.widget = None

def testSize(self):

self.asserEqual(self.widget.getSize(), (40,40))

#构造测试集

def suite():

suite = unittest.TestSuite()

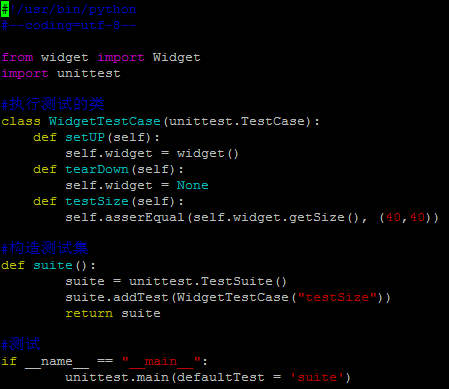
suite.addTest(WidgetTestCase("testSize"))

return suite

#测试

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main(defaultTest = 'suite')



在采用Python单元测试框架后，用于测试的代码做了相应的改动：

用import语句引入unittest模块。

让所有执行测试的类都继承于TestCase类，可以将TestCase看成是对特定类进行测试的方法的集合。

在setUp()方法中进行测试前的初始化工作，并在tearDown()方法中执行测试后的清除工作，setUp()和tearDown()都是TestCase类中定义的方法。

在testSize()中调用assertEqual()方法，对Widget类中getSize()方法的返回值和预期值进行比较，确保两者是相等的，assertEqual()也是TestCase类中定义的方法。

提供名为suite()的全局方法，PyUnit在执行测试的过程调用suit()方法来确定有多少个测试用例需要被执行，可以将TestSuite看成是包含所有测试用例的一个容器。

虽然看起来有点复杂，但PyUnit使得所有的Python程序员都可以使用同样的Python单元测试方法，测试过程不再是杂乱无章的了，而是在同一规范指导下进行的有序行为，这就是使用PyUnit这一自动单元测试框架所带来的最大好处。

**代码示例**

**BaseHTTPServer**

日常开发过程中，我们经常需要修改一些放在 CDN 上的静态文件(如 JavaScript、CSS、HTML 文件等)，这个过程中，我们希望能有一种方式将线上 CDN 的目录映射为本地硬盘上的某个目录，这样，当我们在本地修改了某个文件时，不需要发布，刷新后马上能看到效果。

实现这个功能很简单，关键点如下：

1、在本地开启一个 HTTP 服务，监听 80 端口;

2、修改系统 hosts 文件，添加“127.0.0.1 a.mycdn.com”，将 CDN 域名绑定为本地服务>器地址;

3、配置本地 HTTP 服务，接收到一个 GET 请求后，先检查本地硬盘上是否存在对应的文>件，如存在，则返回这个文件的内容，如不存在，则返回线上对应的内容。

下面的两段代码的功能还相对比较简单，比如没有输出内容的 MIME-Type、Content-Length 等头信息，对可能的阻塞操作(如读取文件超时等)也没有做特别的处理。

代码

#!/usr/bin/python

#--coding=utf-8--

import os

import BaseHTTPServer

LOCAL\_FOLDERS = [

"/home/li475/cdn"

]

BASE\_URL = "http://120.55.195.89:35680"

class WebRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):

def do\_GET(self):

print "Request for '%s' recevied." % self.path

for floder in LOCAL\_FOLDERS:

fn = os.path.join(floder, self.path.replace("/", os.sep)[1:])

if os.path.itself(fn):

self.send\_response(200)

self.wfile.write(open(fn, "rb").read())

break

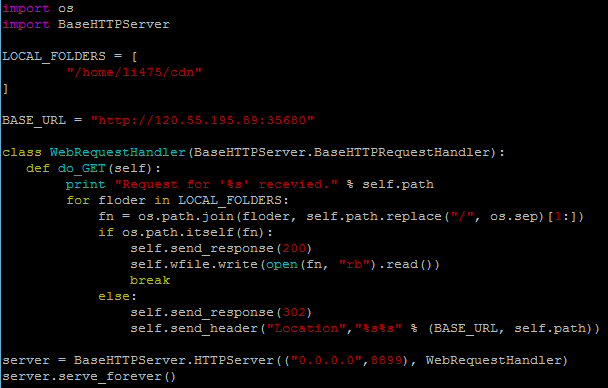
else:

self.send\_response(302)

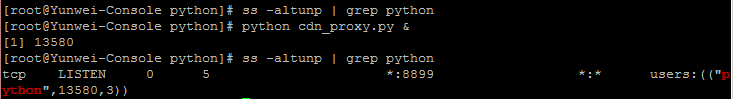
self.send\_header("Location","%s%s" % (BASE\_URL, self.path))

server = BaseHTTPServer.HTTPServer(("0.0.0.0",8899), WebRequestHandler)

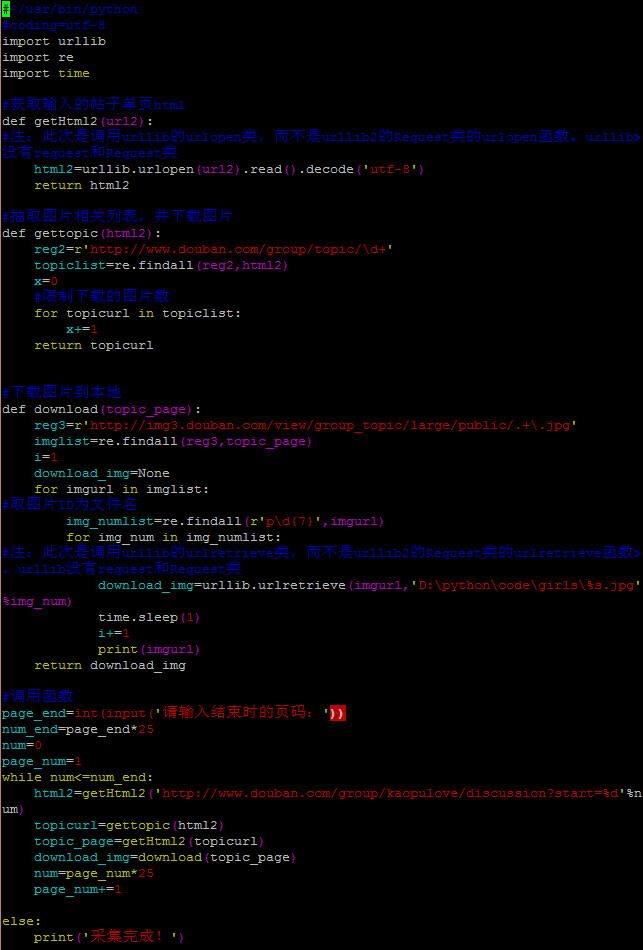
server.serve\_forever()



在后台运行该程序，会在本地打开一个socket监听端口



**爬虫-豆瓣图片**



#!/usr/bin/python

#coding=utf-8

import urllib

import re

import time

#获取输入的帖子单页html

def getHtml2(url2):

html2=urllib.urlopen(url2).read().decode('utf-8')

return html2

#抽取图片相关列表，并下载图片

def gettopic(html2):

reg2=r'http://www.douban.com/group/topic/\d+'

topiclist=re.findall(reg2,html2)

x=0

#限制下载的图片数

for topicurl in topiclist:

x+=1

return topicurl

#下载图片到本地

def download(topic\_page):

reg3=r'http://img3.douban.com/view/group\_topic/large/public/.+\.jpg'

imglist=re.findall(reg3,topic\_page)

i=1

download\_img=None

for imgurl in imglist:

#取图片ID为文件名

img\_numlist=re.findall(r'p\d{7}',imgurl)

for img\_num in img\_numlist:

download\_img=urllib.urlretrieve(imgurl,'D:\python\code\girls\%s.jpg'%img\_num)

time.sleep(1)

i+=1

print(imgurl)

return download\_img

#调用函数

page\_end=int(input('请输入结束时的页码：'))

num\_end=page\_end\*25

num=0

page\_num=1

while num<=num\_end:

html2=getHtml2('http://www.douban.com/group/kaopulove/discussion?start=%d'%num)

topicurl=gettopic(html2)

topic\_page=getHtml2(topicurl)

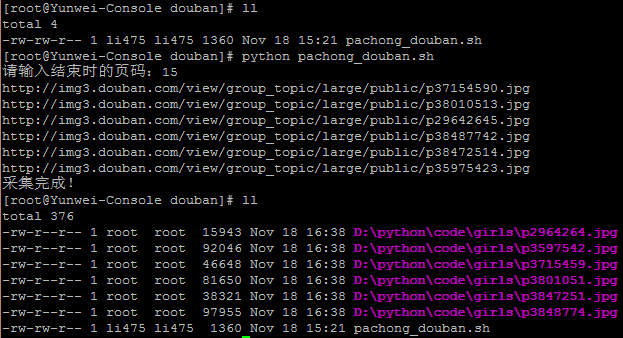
download\_img=download(topic\_page)

num=page\_num\*25

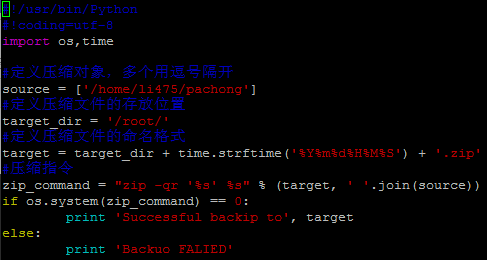
page\_num+=1

else:

print('采集完成！')



**zip压缩**



#!/usr/bin/Python

#!coding=utf-8

import os,time

source = ['/home/li475/pachong']

target\_dir = '/root/'

target = target\_dir + time.strftime('%Y%m%d%H%M%S') + '.zip'

zip\_command = "zip -qr '%s' %s" % (target, ' '.join(source))

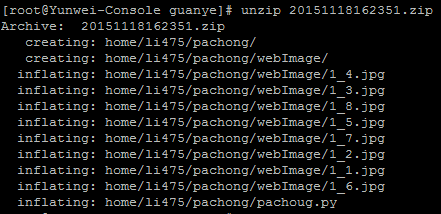
if os.system(zip\_command) == 0:

print 'Successful backip to', target

else:

print 'Backuo FALIED'





**下载图片-代码示例**



# !/usr/bin python

#--\*-- coding:utf-8 --\*--

import os

import urllib

import re

#url = "http://www.111cn.net/anzhuo/photolist-4.htm"

url = ""

def getHtml(url):

file = urllib.urlopen(url)

html = file.read()

return html

def getImageList(html):

reg = 'http[^"]\*?\.jpg'

imgre = re.compile(reg)

imgList = re.findall(imgre,html)

return imgList

def printImageList(imgList):

for i in imgList:

print i

def download(imgList, page):

x = 1

for imgurl in imgList:

print 'download file '+str(x)+' start'

urllib.urlretrieve(imgurl,'./webImage/%s\_%s.jpg'%(page,x))

print 'download file '+ str(x)+ ' end'

x+=1

def downImageNum(pagenum):

page = 1

pageNumber = pagenum

while(page <= pageNumber):

html = getHtml(url)#获得url指向的html内容

imageList = getImageList(html)

printImageList(imageList)

download(imageList,page)

page = page+1

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

os.system('mkdir webImage')

url = raw\_input("enter the web page:")

downImageNum(1)

