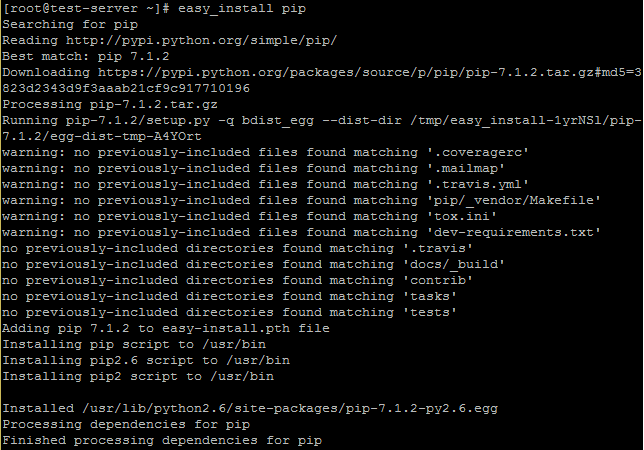
pip使用

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



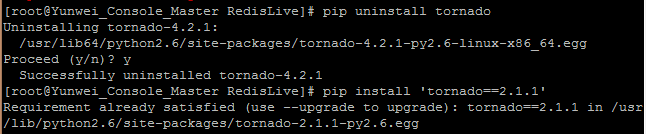
安装特定版本的package，通过使用==, >=, <=, >, <来指定一个版本号。

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

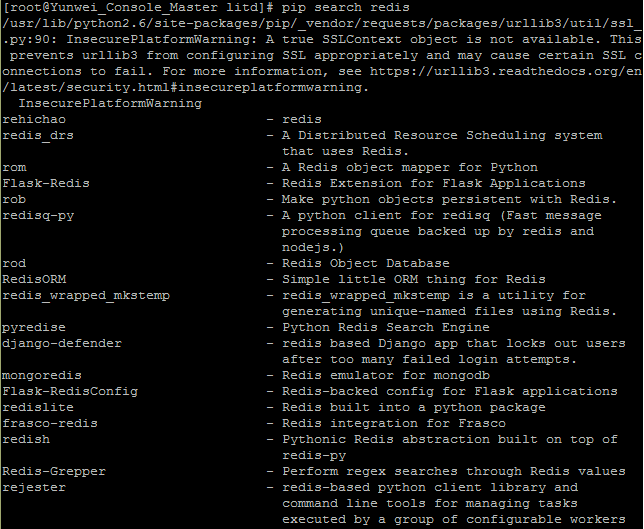
pip install -U python-dateutil

卸载包pip uninstall 包名称

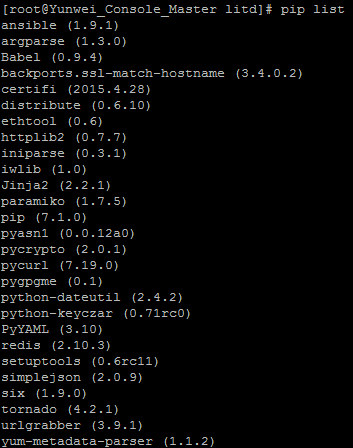


查询包pip search "包名称"

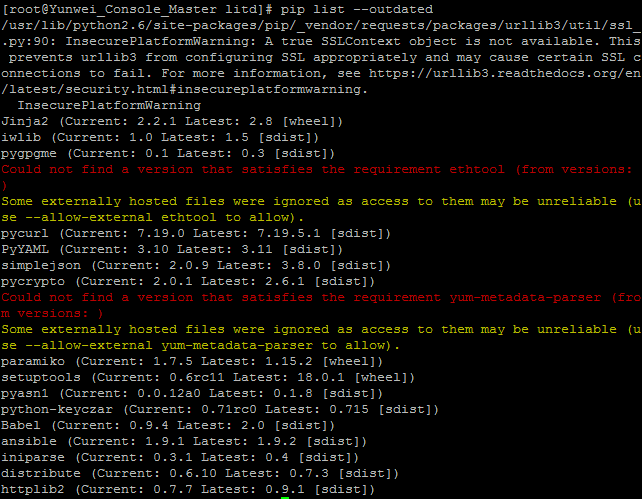
pip search redis



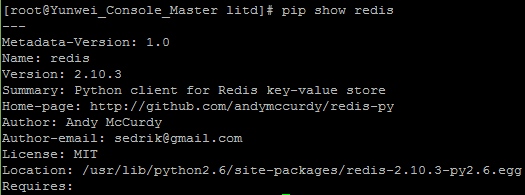
列出安装的packages pip freeze或者pip list



获取过期的库：pip list –outdated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

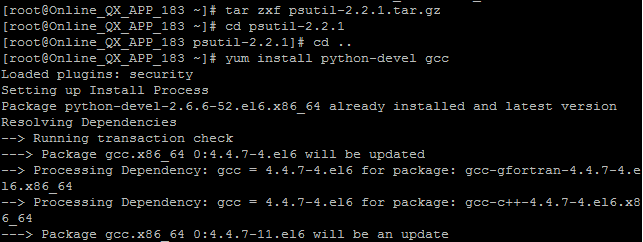
下载链接：https://pypi.python.org/packages/source/p/psutil/psutil-2.2.1.tar.gz

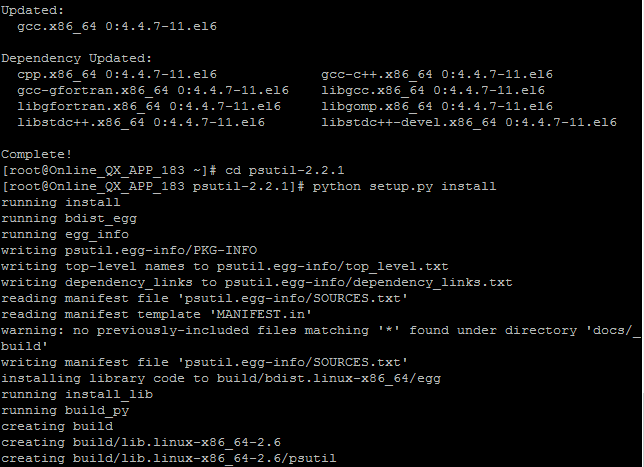
tar zxf psutil-2.2.1.tar.gz

yum install python-devel gcc -y

cd psutil-2.2.1

python setup.py install

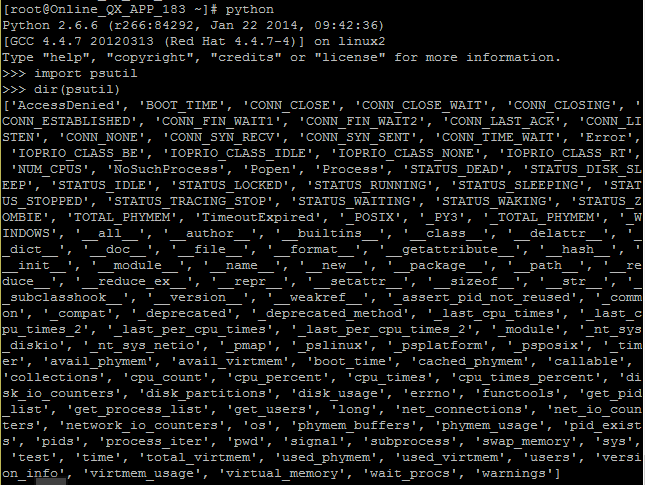




查看psutil所有的函数

import psutil

dir(psutil)

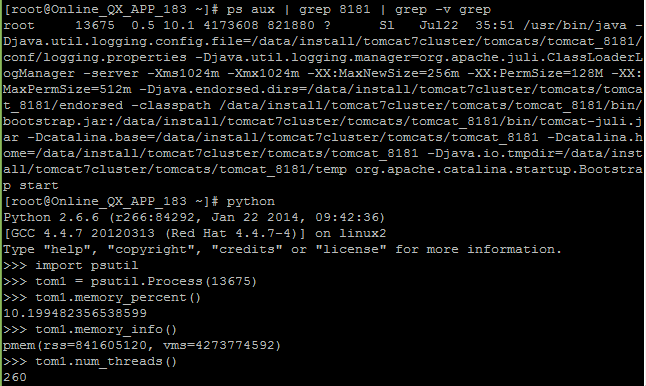


查看JVM的内存使用、threads数量

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)

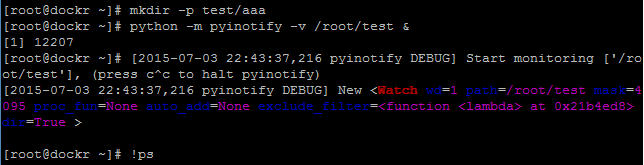
安装pyinotify

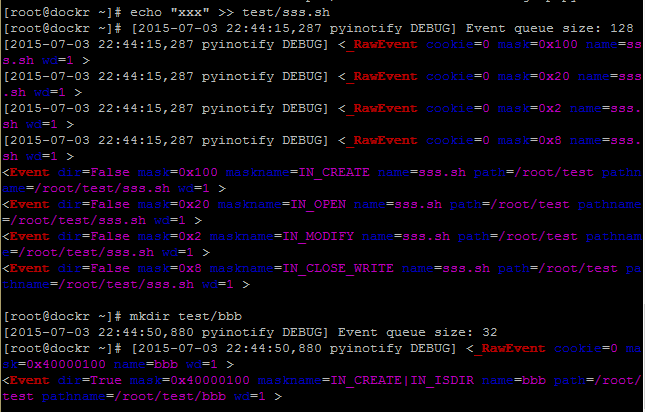
yum install python-pip

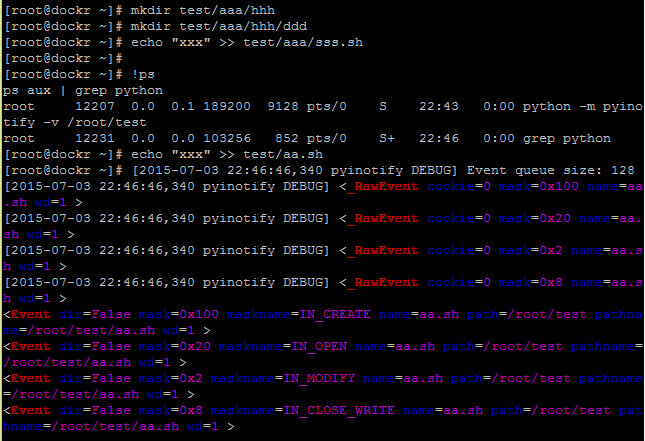
pip install pyinotify

pyinotify初试

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





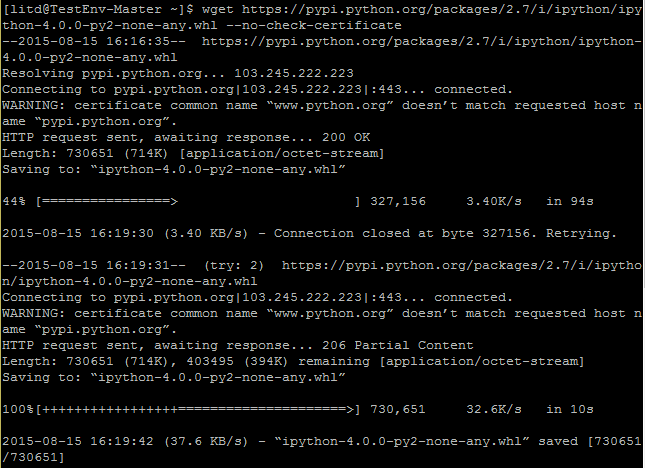


只记录了对/root/test目录的操作，没有记录/root/test下级目录的变化

ipython安装与使用

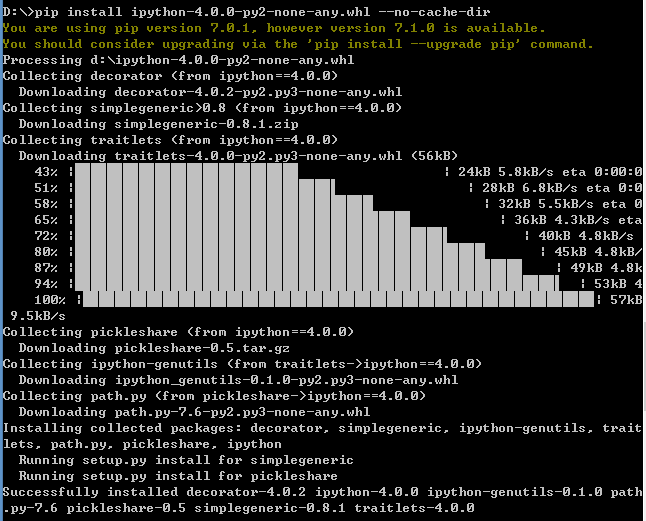
在cmd直接使用pip install ipython安装是简单的做法，但常常网速导致time out ，因此可以在Linux下载ipython-4.0.0-py2-none-any.whl后

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

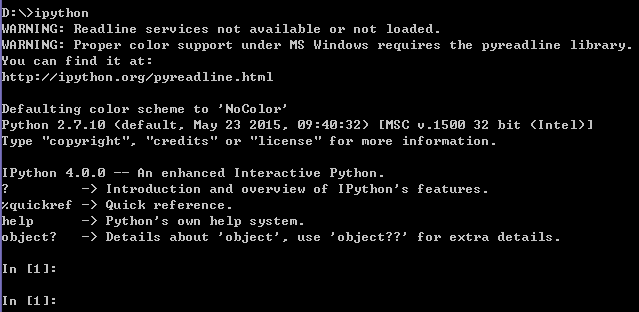


在D盘的根目录，执行

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



安装完成

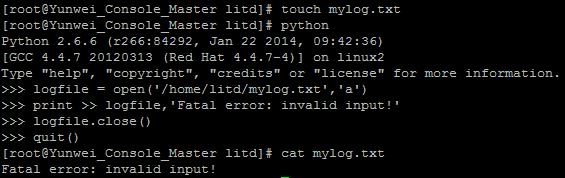


Print保存到文件

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

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

logfile.close()



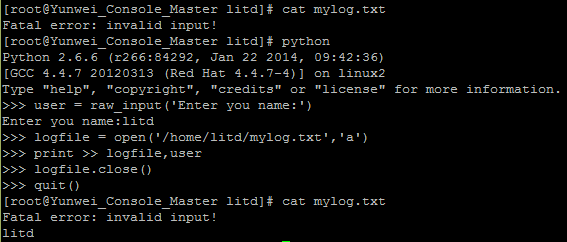
使用raw\_input()捕获输入，然后追加到文件

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

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

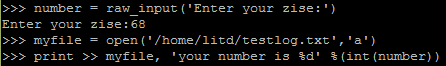
print >> logfile,user

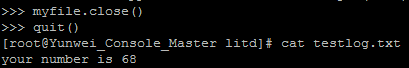
logfile.close()



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

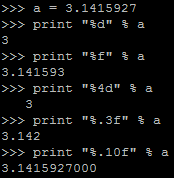
print >> myfile, 'your number is %d' %(int(number))





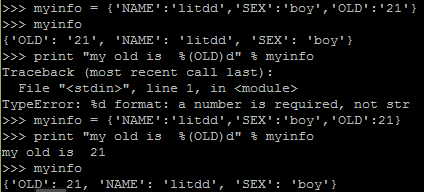
%d %s %f %r

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。

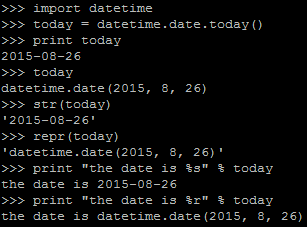


%d可以格式化列表

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")



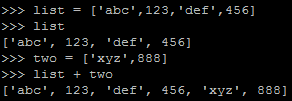
Python 数据类型

Python有五个标准的数据类型：

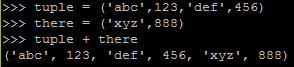
• 数字

• 字符串

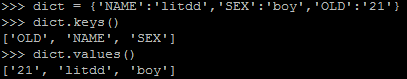
• 列表



• 元组

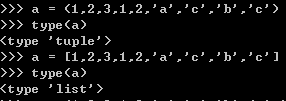


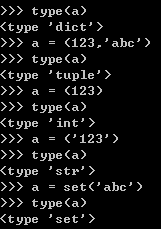
• 字典



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。



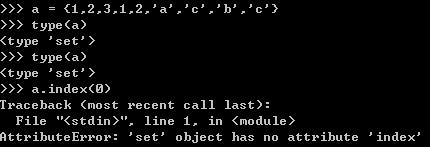


{ }也是set

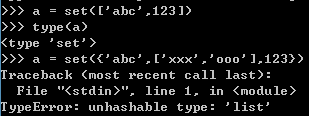


set

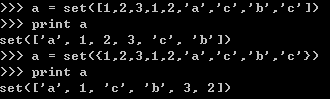
set不是python的一种数据类型，是一个无序不重复元素集，不记录元素位置或者插入点



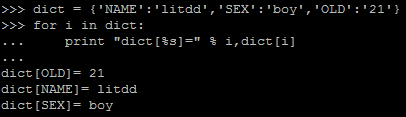
不能包含list



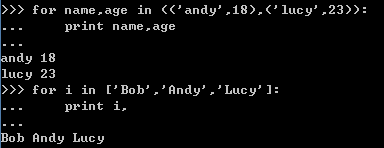
剔除list、set中的重复值



dict遍历

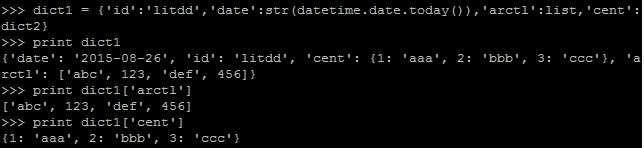


元祖遍历



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

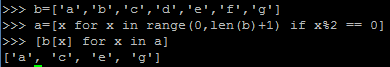
我有一个list，怎么取出其中的偶数列呢？有两个方法：

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

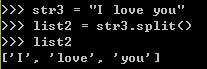
string数据类型的split方法可以将string转换为list

str3 = "I love you"

list2 = str3.split()

list2

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



特殊情况也可以

str1 = '''I Love

... "python"

... '''

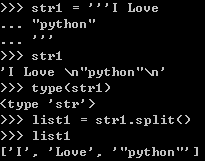
str1

'I Love \n"python"\n'

list1 = str1.split()

list1

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



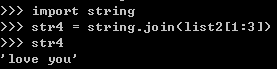
string模块的join方法可以将list拼接为string

>>> import string

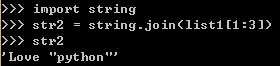
>>> str4 = string.join(list2[1:3])

>>> str4

'love you'

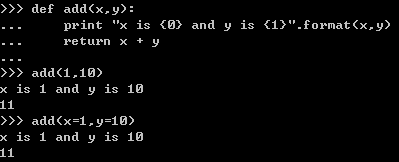


复杂的也没关系



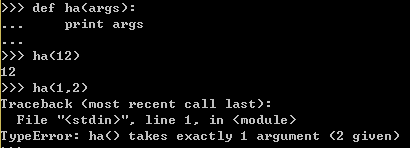
函数参数

两个参数

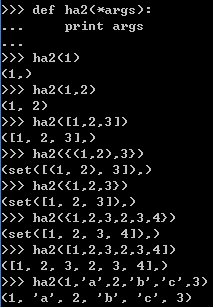


args、\*args、\*\*args

args不能接受除string以外的字符串

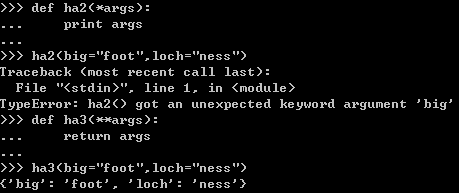


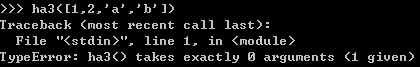
\*args接受list、set、tuple但是不能接受dict





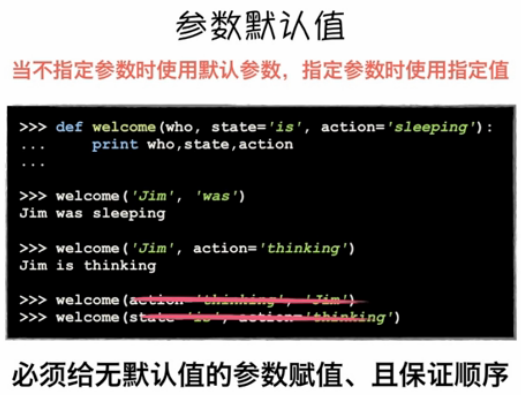
\*\*args可以生成dict，但是不接受list和dict





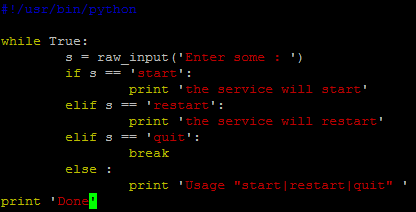


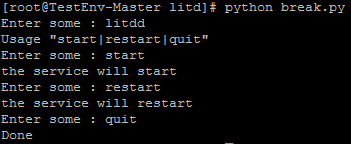




while和for循环

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;

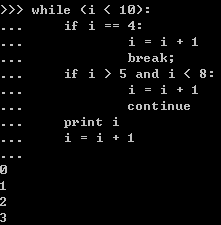
if i > 5 and i < 8:

i = i + 1

continue

print i

i = i + 1



i = 0

while (i < 10):

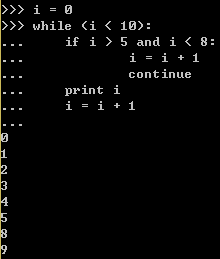
if i > 5 and i < 8:

i = i + 1

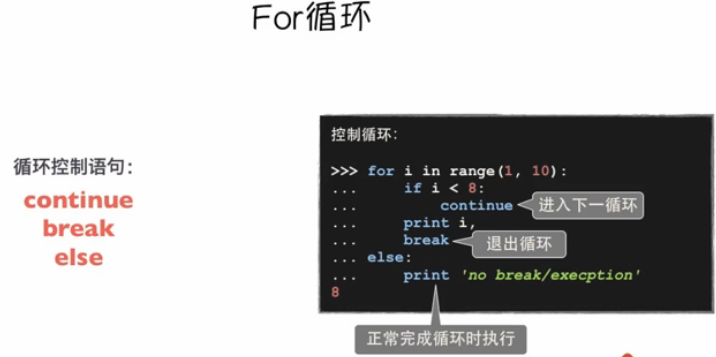
continue

print i

i = i + 1



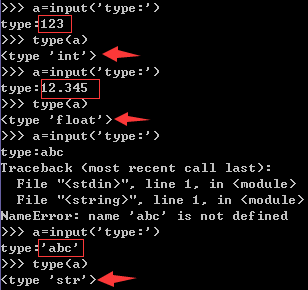
for循环的扩展语句

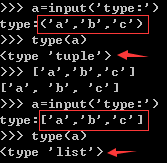


http://edu.51cto.com/lesson/id-26506.html

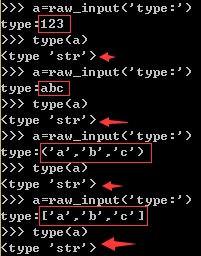
input、rwa\_input

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



迭代器

iter()方法生成迭代器

obj = range(6)

itor = iter(obj)

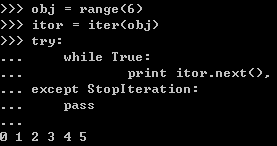
try:

while True:

print itor.next(),

except StopIteration:

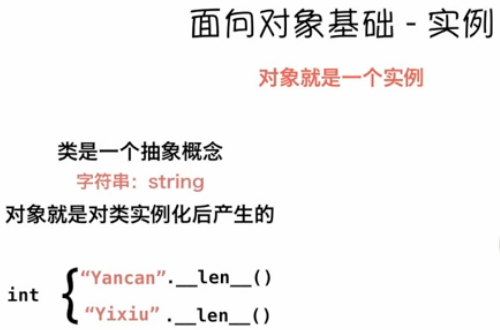
pass

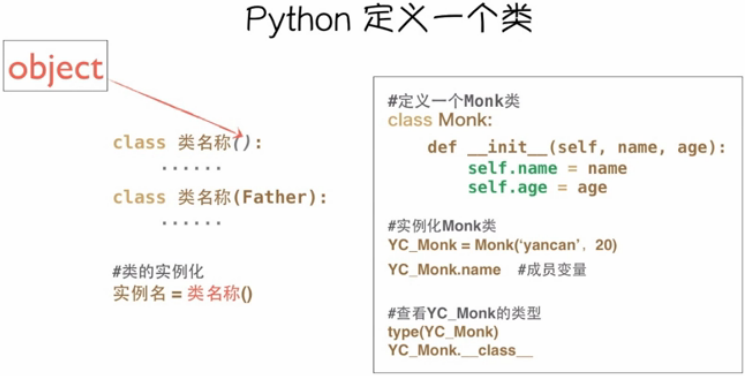


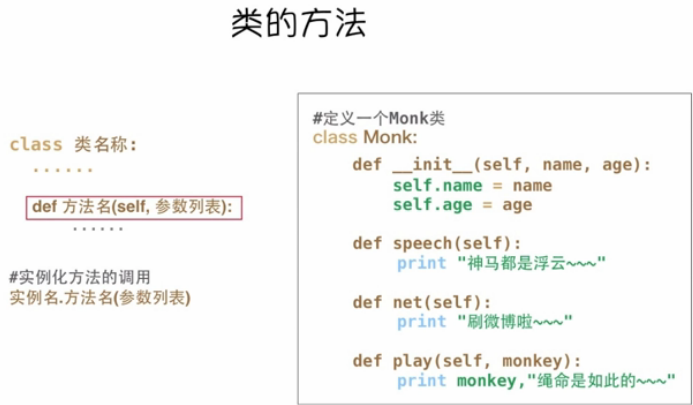


类

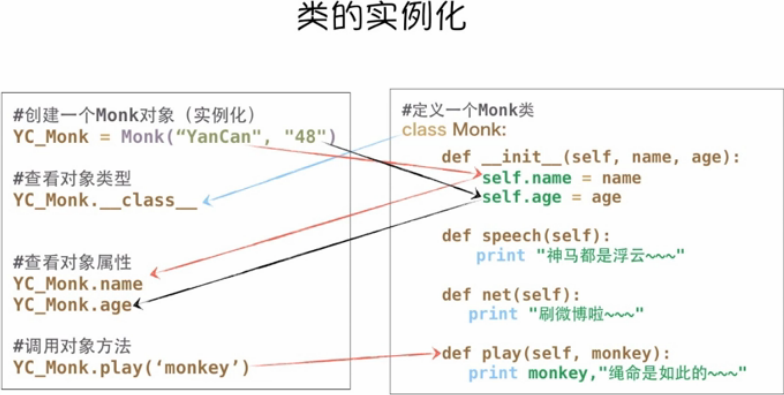








类的实例化



class test:

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

self.name = name

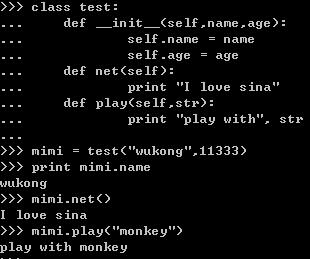
self.age = age

def net(self):

print "I love sina"

def play(self,str):

print "play with", str



IO基础

其实是file类



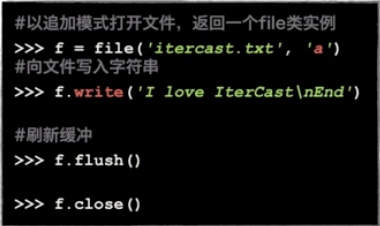
Open函数-读文件



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:

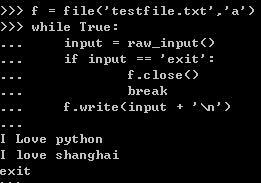
input = raw\_input()

if input == 'exit':

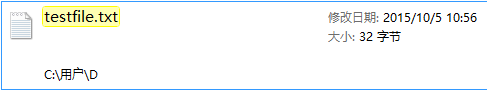
f.close()

break

f.write(input + '\n')



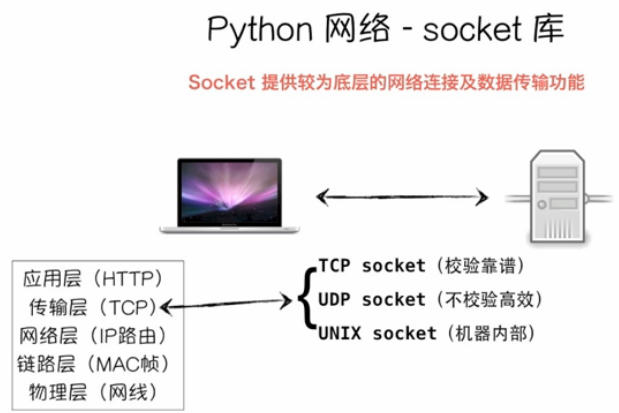
文件在



文件的内容



socket









HTTP



