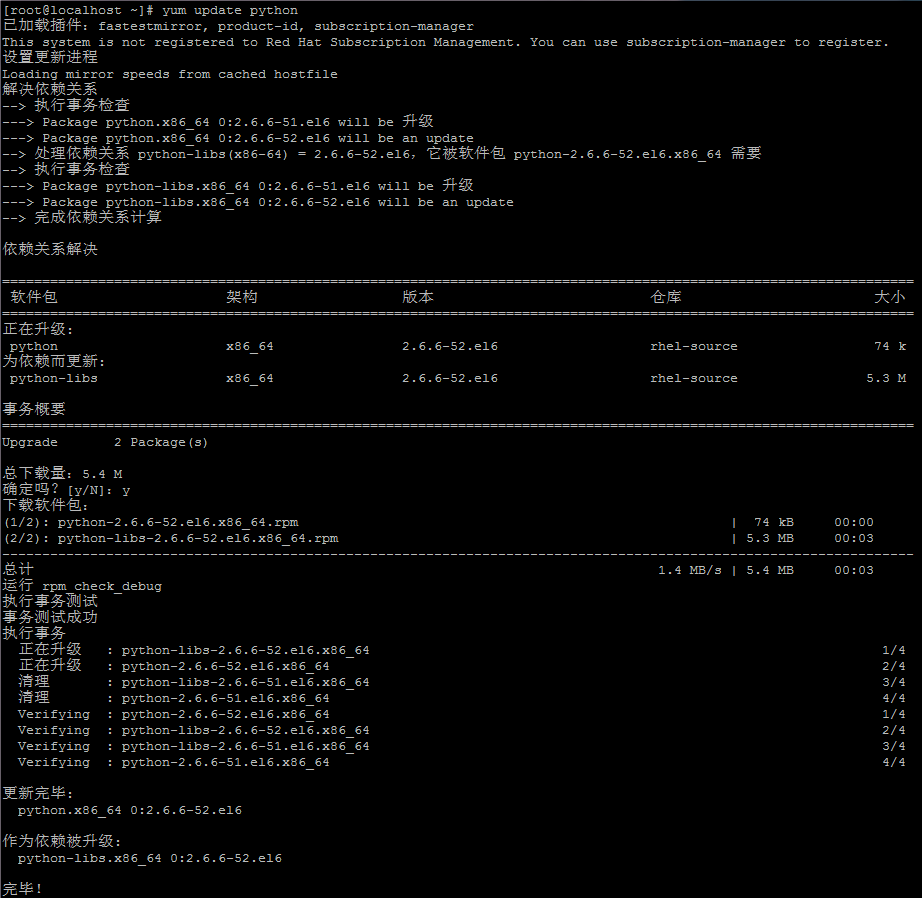
Python和第三方库安装

（1）yum升级python



Python的shell路径



Python脚本





（2）源代码安装python 2.7.9

wget <https://www.python.org/ftp/python/2.7.9/Python-2.7.9.tar.xz>

rpm -e python –nodeps

tar xf Python-2.7.9.tar.xz

cd Python-2.7.9

./configure --prefix=/usr/local

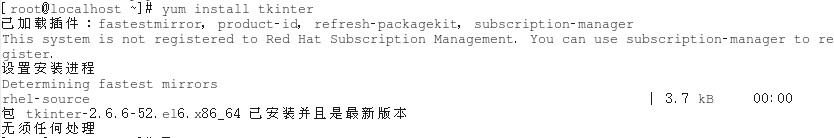
make && make install

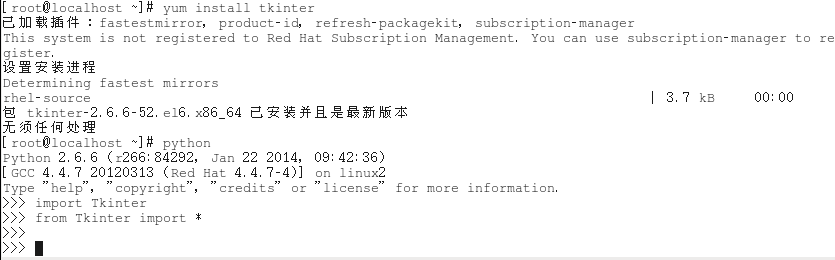
ln -s /usr/local/bin/python2.7 /usr/bin/python



（3）使用tkinter模块

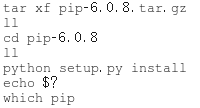
yum install tkinter





（4）使用PIL模块

Pip下载地址：<https://pypi.python.org/pypi/pip>





安装python的处理图像的工具库模块Python Imaging Library

版本更新http://www.pythonware.com/products/pil/#pil117

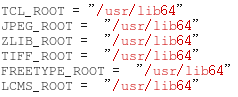
下载地址<http://effbot.org/media/downloads/Imaging-1.1.7.tar.gz>

wget http://effbot.org/media/downloads/Imaging-1.1.7.tar.gz

tar xf Imaging-1.1.7.tar.gz

cd Imaging-1.1.7

修改Imaging-1.1.7/setup.py 这个文件，修改下面几行代码（默认TCL\_ROOT的设置为NONE，这里要传到系统库的路径才行）：



安装需要的所有开发包

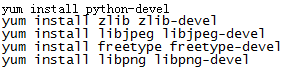
yum install python-devel

yum install zlib zlib-devel

yum install libjpeg libjpeg-devel

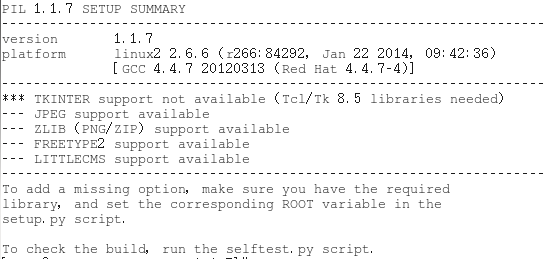
yum install freetype freetype-devel

yum install libpng libpng-devel

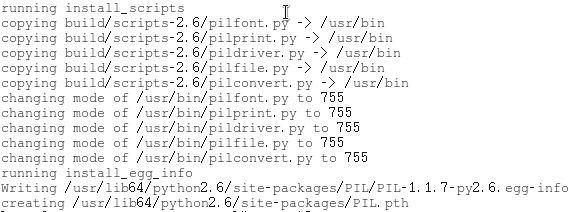


还要不要yum install lcms lcms-devel？

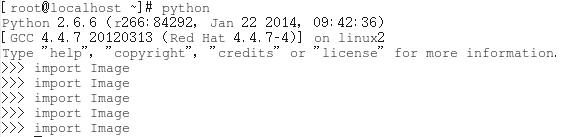
进行安装前的检查：python setup.py build\_ext -i



安装python setup.py install



安装成功，



1. raw\_input和input

在Python中最常用的从键盘获取输入的函数是raw\_input()和input()。但是这两个函数有很大的区别：

raw\_input()以字符串的形式返回用户输入的一切内容；

input()会根据输入内容的形式确定返回的形式。

#!/usr/bin/python

# -\*- coding: utf-8 -\*- #设定字符串编码环境为utf-8

name = raw\_input('please enter your name:')

print 'hello,%s' % name

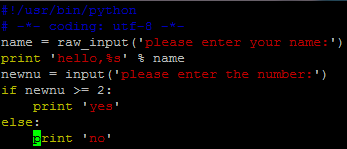
newnu = input('please enter the number:')

if newnu >= 2:

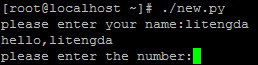
print 'yes'

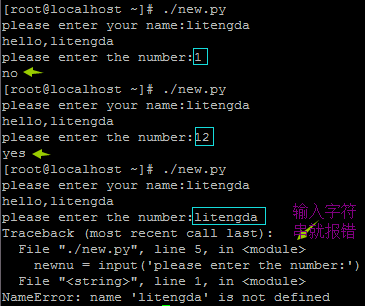
else:

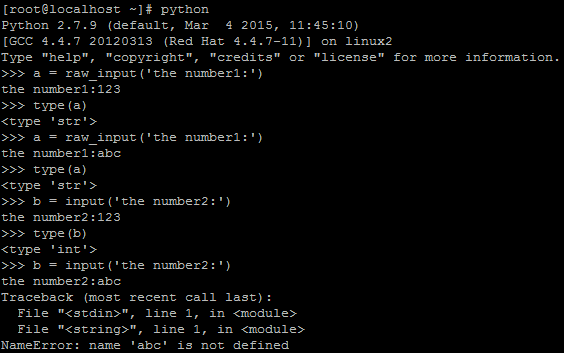
print 'no'



执行脚本

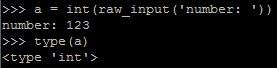






从raw\_input()读取的内容永远以字符串的形式返回，用int()可以把字符串转换为整型：

int(raw\_input('birth: '))



1. list和tuple

Python内置的一种数据类型是列表：list。list是一种有序的集合

>>> users = ['aaa','bbb','ccc']

>>> users

['aaa', 'bbb', 'ccc']

>>> len(users)

3

>>> users[0]

'aaa'

>>> users[1]

'bbb'

>>> users[2]

'ccc'

>>> users[-1]

'ccc'

>>> users[-2]

'bbb'

>>> users.append('ddd')

>>> users

['aaa', 'bbb', 'ccc', 'ddd']

>>> users.insert(2,'eee')

>>> users

['aaa', 'bbb', 'eee', 'ccc', 'ddd']

>>> users.pop(2)

'eee'

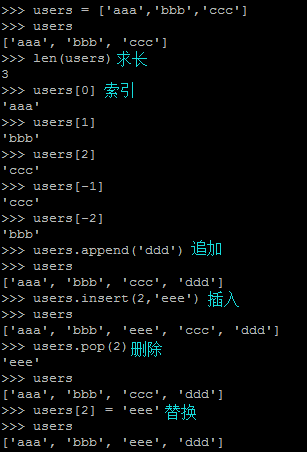
>>> users

['aaa', 'bbb', 'ccc', 'ddd']

>>> users[2] = 'eee'

>>> users

['aaa', 'bbb', 'eee', 'ddd']



>>> new = [123,True,['python','java'],'love']

>>> new

[123, True, ['python', 'java'], 'love']

>>> len(new)

4

>>> now = ['python','java']

>>> last = [123,True,now,'love']

>>> last

[123, True, ['python', 'java'], 'love']

>>> len(last)

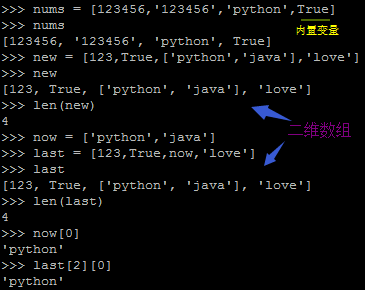
4

>>> now[0]

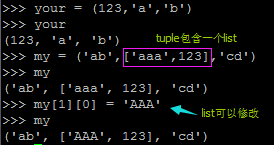
'python'

>>> last[2][0]

'python'



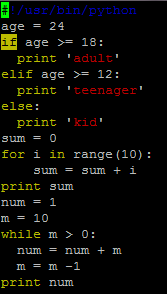
另一种有序列表叫元组：tuple。tuple和list非常类似，但是tuple一旦初始化就不能修改



tuple所谓的“不变”是说，tuple的每个元素，指向永远不变。不可变的tuple有什么意义？因为tuple不可变，所以代码更安全。如果可能，能用tuple代替list就尽量用tuple。

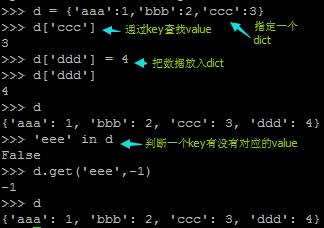
1. if和while、for

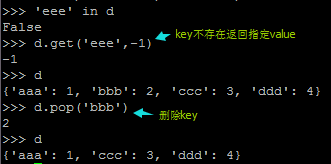
if语句执行有个特点，它是从上往下判断，如果在某个判断上是True，把该判断对应的语句执行后，就忽略掉剩下的elif和else



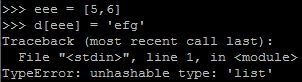
1. set和dict

Python内置了字典：dict的支持，dict全称dictionary，在其他语言中也称为map，使用键-值（key-value）存储，具有极快的查找速度。key-value存储方式，在放进去的时候，必须根据key算出value的存放位置，这样，取的时候才能根据key直接拿到value。一个key只能对应一个value。

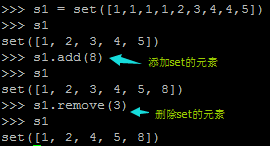




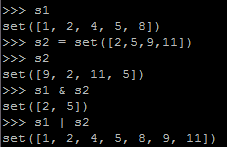
在Python中，字符串、整数等都是不可变的，因此，可以放心地作为key。而list是可变的，就不能作为key：



set和dict类似，也是一组key的集合，但不存储value。由于key不能重复，所以，在set中，没有重复的key。



set可以看成数学意义上的无序和无重复元素的集合，因此，两个set可以做数学意义上的交集、并集等操作：



1. 函数

计算xn

#!/usr/bin/python

def power(x, n):

s = 1

while n > 0:

n = n - 1

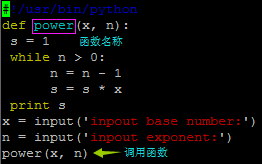
s = s \* x

print s

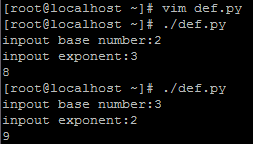
x = input('inpout base number:')

n = input('inpout exponent:')

power(x, n)

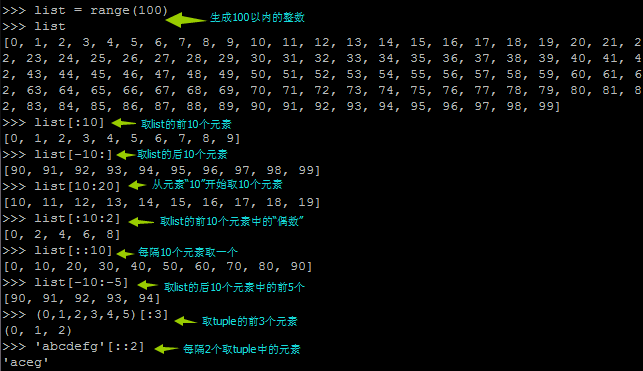


执行脚本



1. 切片

Python提供了切片（Slice）操作符, 取一个list或tuple的部分元素



1. 迭代

如果给定一个list或tuple，我们可以通过for循环来遍历这个list或tuple，这种遍历我们成为迭代（Iteration）。

#!/usr/bin/python

d = {'a':1,'b':2,'c':3,'d':4}

for key in d:

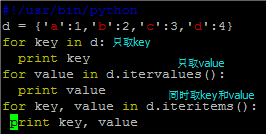
print key

for value in d.itervalues():

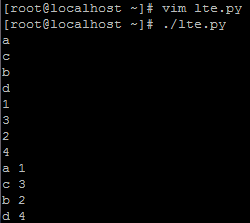
print value

for key, value in d.iteritems():

print key, value



执行脚本



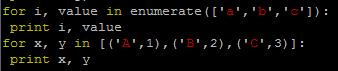
Python内置的enumerate函数可以把一个list变成索引-元素对，这样就可以在for循环中同时迭代索引和元素本身

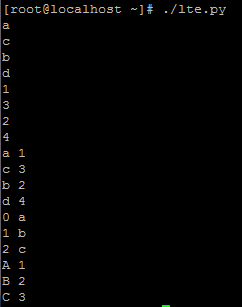
for i, value in enumerate(['a','b','c']):

print i, value

for x, y in [('A',1),('B',2),('C',3)]:

print x, y



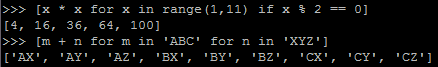


1. 列表生成式

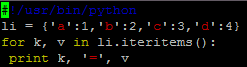
列表生成式即List Comprehensions，把要生成的元素x \* x放到前面，后面跟for循环，就可以把list创建出来

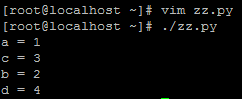
[x \* x for x in range(1,11) if x % 2 == 0] 取1-11的偶数并作平方运算；

[m + n for m in 'ABC' for n in 'XYZ'] 分别从ABC和XYZ中取出1个值合并。



同时迭代dict的iteritems()key和value





列表生成式也可以使用两个变量来生成list

[k + '=' + v for k, v in d.iteritems()]



1. 生成器

列表元素可以按照某种算法推算出来，可以在循环的过程中不断推算出后续的元素，这样就不必创建完整的list，从而节省大量的空间。在Python中，这种一边循环一边计算的机制，称为生成器（Generator）

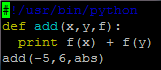
1. 高阶函数

既然变量可以指向函数，函数的参数能接收变量，那么一个函数就可以接收另一个函数作为参数，这种函数就称之为高阶函数。

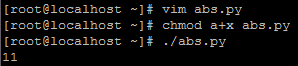
def add(x,y,f):

print f(x) + f(y)

add(-5,6,abs)



abs是python内置绝对值运算的函数，把abs函数赋值给变量f  
执行脚本



1. 多线程

运行多个线程类似并发运行多个不同的程序，但是有以下好处：

1.同一进程下的多个线程与主线程共享数据空间，这样在互相共享信息、通信上比独立进程之间更容易。

2.线程通常也叫做轻量级进程(light-weight processes)，与进程相比不需要太多的内存；线程比进程消耗系统资源少。

一个线程有开始，有运行队列，有结束。它有一个指针(句柄)用以跟踪线程在当前上下文中的运行情况。

1.线程可以被中断

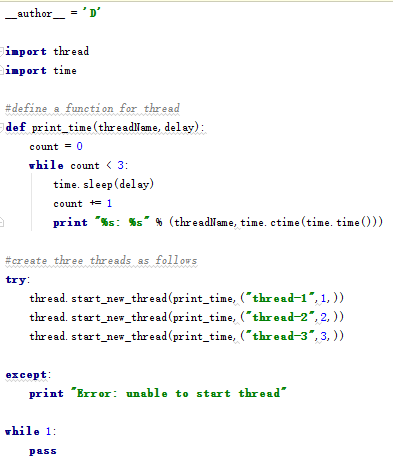
2.线程可以在其他线程运行时挂起，称为让步。

可以通过thread模块中的start\_new\_thread方法创建一个新的线程：thread.start\_new\_thread(function,args[,kwargs]，function是一个python函数，args是一个参数列表。

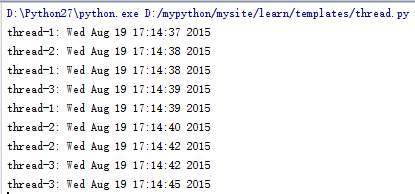
调用这个方法会立刻返回，新创建的线程启动并使用参数列表调用传入的函数(将参数列表传递给该函数)，当该函数返回时，该线程终结。

实例代码如下：

**import** thread  
**import** time  
  
*#define a function for thread***def** print\_time(threadName,delay):  
 count = 0  
 **while** count < 3:  
 time.sleep(delay)  
 count += 1  
 **print "%s: %s"** % (threadName,time.ctime(time.time()))  
  
*#create three threads as follows***try**:  
 thread.start\_new\_thread(print\_time,(**"thread-1"**,1,))  
 thread.start\_new\_thread(print\_time,(**"thread-2"**,2,))  
 thread.start\_new\_thread(print\_time,(**"thread-3"**,3,))  
  
**except**:  
 **print "Error: unable to start thread"  
  
while** 1:  
 **pass**



运行结果如下：



一共执行了3次，同一秒进行了两个线程

使用threading模块创建线程

thread模块在可控性易用性上不如新的threading模块。

threading模块在实现了thread模块的所有方法的基础上添加了另外一些方法(这些方法可以看作静态方法)：

1.threading.activeCount():返回当前存活线程的数量。

2.threading.currentThread():在调用线程中返回被调用线程的数量

3.threading.enumerate():返回当前存活线程的列表

threading模块的Thread类中也提供了一些方法(实例方法):

1.run():一个线程的运行主体(可以类比Java中的Thread\Runnable的run方法)

2.start():调用线程实例的该方法启动该线程。

3.join([time]):调用join()，主线程会等待该线程结束，可以设置超时时间，其原理是:join方法将本线程添加至父线程的线程池，父线程轮询该线程池。

4.isAlive():检查当前线程是否还在执行。

5.getName():获取线程的name

6.setName():设置线程的name

用threading模块创建一个线程有以下步骤：

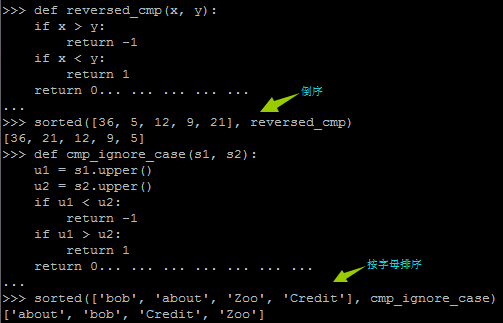
1.继承threading.Tread

2.覆盖\_\_init\_\_方法

3.覆盖run方法来实现线程主体

Sorted-排序算法

sorted()函数也是一个高阶函数，它还可以接收一个比较函数来实现自定义的排序。默认情况下，对字符串排序，是按照ASCII的大小比较的，由于'Z' < 'a'，结果，大写字母Z会排在小写字母a的前面。



\_\_name\_\_的使用

模块是对象，并且所有的模块都有一个内置属性 \_\_name\_\_。一个模块的 \_\_name\_\_ 的值取决于您如何应用模块。如果 import 一个模块，那么模块\_\_name\_\_ 的值通常为模块文件名，不带路径或者文件扩展名。但是您也可以像一个标准的程序样直接运行模块，在这 种情况下, \_\_name\_\_ 的值将是一个特别缺省"\_\_main\_\_"。

test1.py

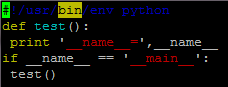
#!/usr/bin/env python

def test():

print '\_\_name\_\_=',\_\_name\_\_

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

test()



test2.py

#!/usr/bin/env python

import test1

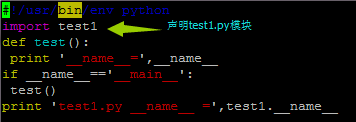
def test():

print '\_\_name\_\_=',\_\_name\_\_

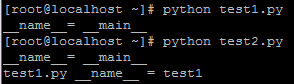
if \_\_name\_\_=='\_\_main\_\_':

test()

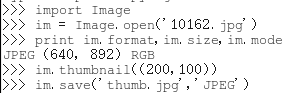
print 'test1.py \_\_name\_\_ =',test1.\_\_name\_\_

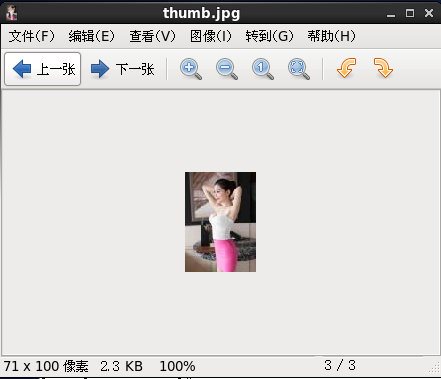


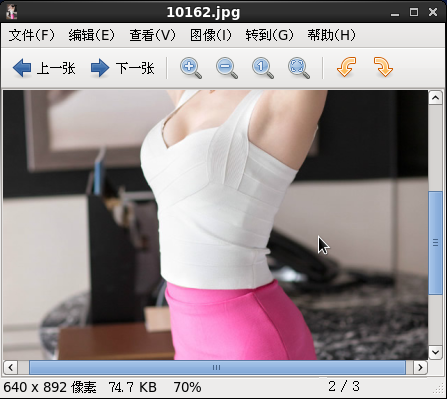
执行脚本



thumbnail生成缩略图







第一个GUI程序

#!/usr/bin/env python

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

from Tkinter import \*

class Application(Frame):

def \_\_init\_\_(self, master=None):

Frame.\_\_init\_\_(self, master)

self.pack()

self.createWidgets()

def createWidgets(self):

self.helloLabel = Label(self, text='Hello, world!')

self.helloLabel.pack()

self.quitButton = Button(self, text='Quit', command=self.quit)

self.quitButton.pack()

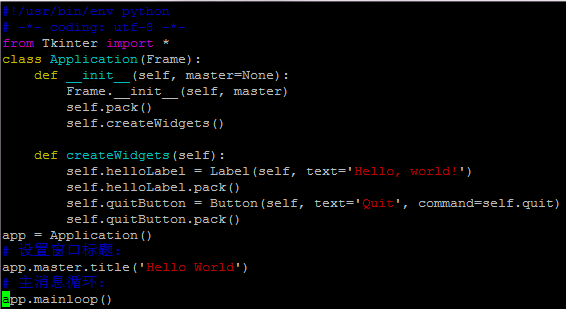
app = Application()

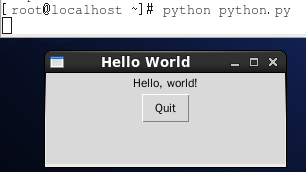
# 设置窗口标题:

app.master.title('Hello World')

# 主消息循环:

app.mainloop()





GUI编程(Tkinter)

#!/usr/bin/env python

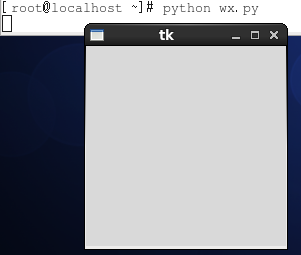
# -\*- coding: utf-8 -\*-

import Tkinter

top = Tkinter.Tk()

# 进入消息循环

top.mainloop()



<http://www.yiibai.com/python/python_gui_programming.html>

发送邮件(Email SMTP)

#!/usr/bin/python

import smtplib

sender = 'python@localdomain.com'

receivers = ['litengdait@163.com']

message = """From: python

To: To litengdait@163.com

Subject: SMTP e-mail test

This is a test e-mail message.

"""

try:

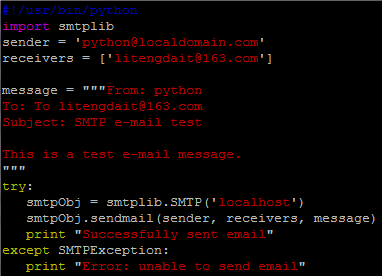
smtpObj = smtplib.SMTP('localhost')

smtpObj.sendmail(sender, receivers, message)

print "Successfully sent email"

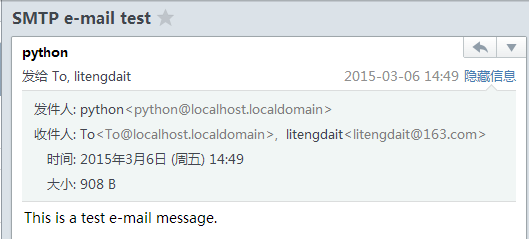
except SMTPException:

print "Error: unable to send email"



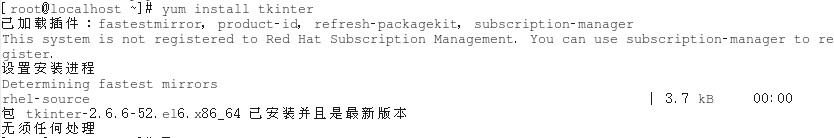
执行脚本

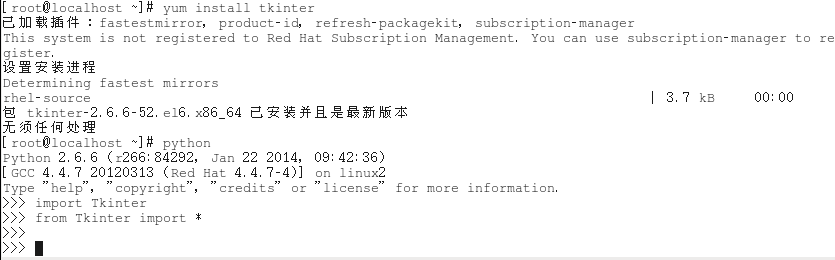




使用Tkinter模块

yum install tkinter



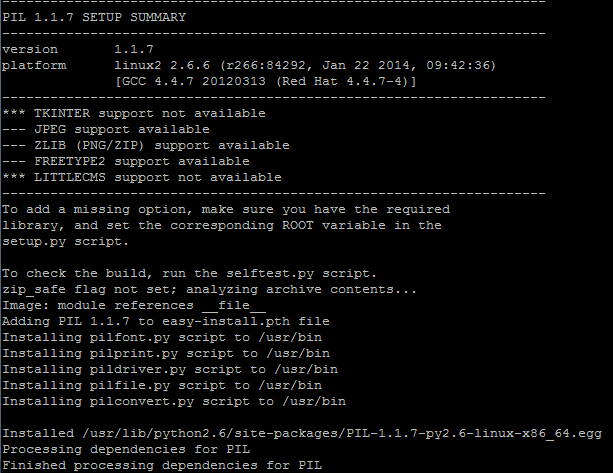


使用PIL模块

Rhel6.5系统自动的python2.6.6带有easy\_install命令，可以使用这个自动安装PIL 1.7.6

yum install python-devel zlib zlib-devel libjpeg libjpeg-devel freetype freetype-devel libpng libpng-devel

easy\_install PIL



下面以源代码编译安装PIL1.1.7

版本更新http://www.pythonware.com/products/pil/#pil117

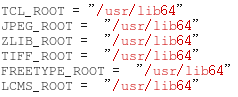
下载地址<http://effbot.org/media/downloads/Imaging-1.1.7.tar.gz>

wget http://effbot.org/media/downloads/Imaging-1.1.7.tar.gz

tar xf Imaging-1.1.7.tar.gz

cd Imaging-1.1.7

修改Imaging-1.1.7/setup.py 这个文件，修改下面几行代码（默认TCL\_ROOT的设置为NONE，这里要传到系统库的路径才行）：



安装需要的所有开发包

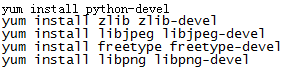
yum install python-devel

yum install zlib zlib-devel

yum install libjpeg libjpeg-devel

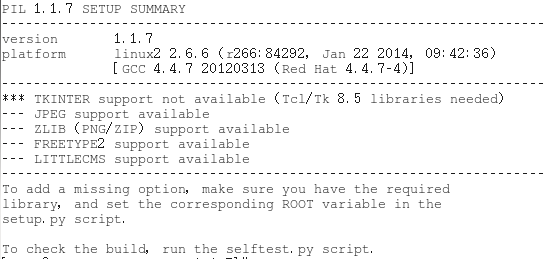
yum install freetype freetype-devel

yum install libpng libpng-devel

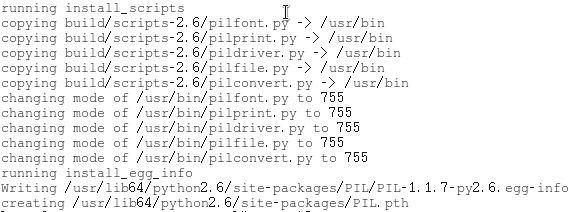


还要不要yum install lcms lcms-devel？

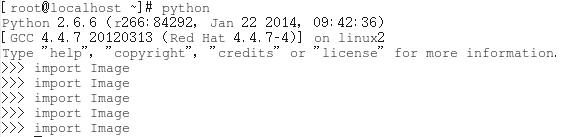
进行安装前的检查：python setup.py build\_ext -i



安装python setup.py install



安装成功，



1. tkinter小窗口

#!/usr/bin/env python

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

from Tkinter import \*

class Application(Frame):

def \_\_init\_\_(self, master=None):

Frame.\_\_init\_\_(self, master)

self.pack()

self.createWidgets()

def createWidgets(self):

self.helloLabel = Label(self, text='Hello, world!')

self.helloLabel.pack()

self.quitButton = Button(self, text='Quit', command=self.quit)

self.quitButton.pack()

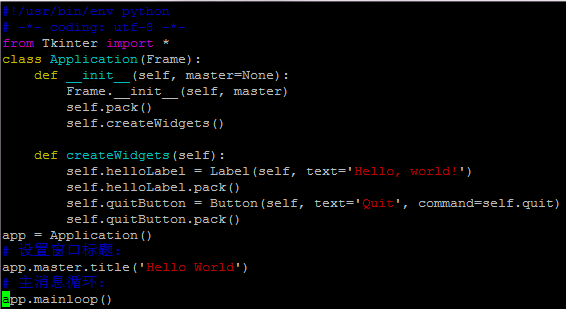
app = Application()

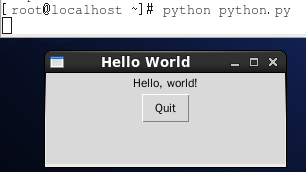
# 设置窗口标题:

app.master.title('Hello World')

# 主消息循环:

app.mainloop()





#!/usr/bin/env python

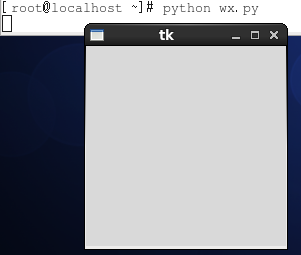
# -\*- coding: utf-8 -\*-

import Tkinter

top = Tkinter.Tk()

# 进入消息循环

top.mainloop()



<http://www.yiibai.com/python/python_gui_programming.html>

#!/usr/bin/ env python

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

from Tkinter import \*

import tkMessageBox

class Application(Frame):

def \_\_init\_\_(self, master=None):

Frame.\_\_init\_\_(self, master)

self.pack()

self.createWidgets()

def createWidgets(self):

self.nameInput = Entry(self)

self.nameInput.pack()

self.alertButton = Button(self, text='Hello', command=self.hello)

self.alertButton.pack()

def hello(self):

name = self.nameInput.get() or 'world'

tkMessageBox.showinfo('Message', 'Hello, %s' % name)

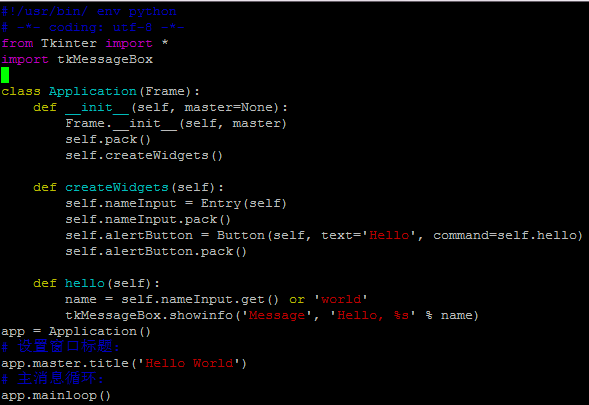
app = Application()

# 设置窗口标题:

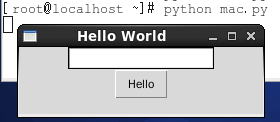
app.master.title('Hello World')

# 主消息循环:

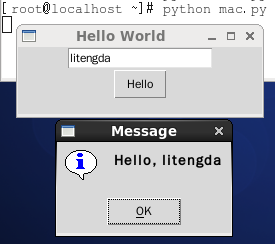
app.mainloop()



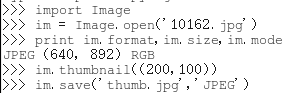
执行脚本

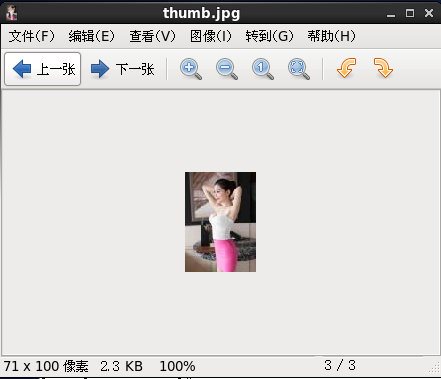


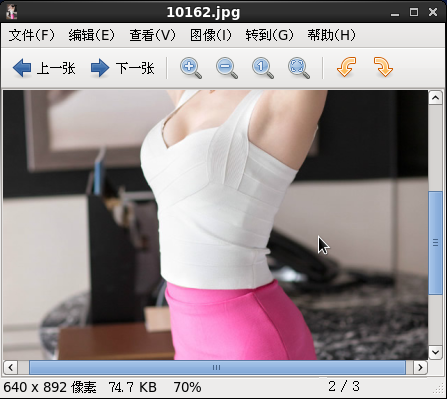
填入信息



2. thumbnail生成缩略图







**实例**

1. 函数 – 可变参数

参数的名字前面有个 \* 号，就是可变参数，我们可以传入0个、1个或多个参数，该参数被看成一个 tuple 。下面是计算n个值的平均数：

#!/usr/bin/env python

from \_\_future\_\_ import division

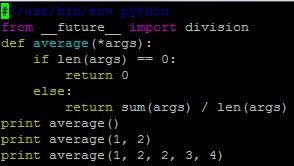
def average(\*args):

if len(args) == 0:

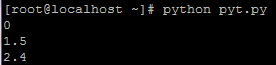
return 0

else:

return sum(args) / len(args)



执行脚本



1. .append( )

扩展元素，比如每次从listA取一个元素加入到listB：

#!/usr/bin/env python

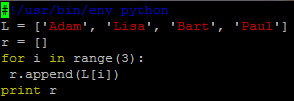
L = ['Adam', 'Lisa', 'Bart', 'Paul']

r = []

for i in range(3):

r.append(L[i])

print r



执行脚本



1. :

“:”在Python中大多数指切片，截取list、字符串

设计一个函数，它接受一个字符串，然后返回一个仅首字母变成大写的字符串，可以这样写：

def firstCharUpper(s):

return s[:1].upper()+s[1-len(s):]

print firstCharUpper('hello')

print firstCharUpper('sunday')

print firstCharUpper('september')