python 实例手册

```
#encoding:utf8
#  设定编码-支持中文
```

1 0 说明

手册制作: 雪松 littlepy reboot
更新日期: 2014-10-29
欢迎系统运维加入 Q 群: 198173206   # 加群请回答问题
欢迎运维开发加入 Q 群: 365534424   # 不定期技术分享

请使用"notepad++"打开此文档,"alt+0"将函数折叠后方便查阅
请勿删除信息，转载请说明出处，抵制不道德行为。
错误在所难免，还望指正！

python 实例手册下载地址:
http://hi.baidu.com/quanzhou722/item/cf4471f8e23d3149932af2a7

shell 实例手册最新下载地址:
http://hi.baidu.com/quanzhou722/item/f4a4f3c9eb37f02d46d5c0d9

LazyManage 运维批量管理软件下载[shell]:
http://hi.baidu.com/quanzhou722/item/4ccf7e88a877eaccef083d1a

LazyManage 运维批量管理软件下载[python]:
http://hi.baidu.com/quanzhou722/item/4213db3626a949fe96f88d3c

2 1 基础

2.1 查看帮助

```
import os
for i in dir(os):
    print i             #  模块的方法
help(os.path)           #  方法的帮助
```

2.2 调试

python -m trace -t aaaaaa.py

pip 模块安装

yum install python-pip # centos 安装 pip
sudo apt-get install python-pip # ubuntu 安装 pip
pip 官方安装脚本
wget https://raw.github.com/pypa/pip/master/contrib/get-pip.py
python get-pip.py
加载环境变量
vim /etc/profile
export PATH=/usr/local/python27/bin:$PATH
. /etc/profile

pip install Package # 安装包 pip install requests
pip show --files Package # 查看安装包时安装了哪些文件
pip show --files Package # 查看哪些包有更新
pip install --upgrade Package # 更新一个软件包
pip uninstall Package # 卸载软件包

2.3 变量

r=r'\n' # 输出时原型打印
u=u'中文' # 定义为 unicode 编码
global x # 全局变量
a = 0 or 2 or 1 # 布尔运算赋值,a 值为 True 既不处理后面,a 值为 2. None、字符串''、空元组()、空列表[],空字典{}、0、空字符串都是 false
name = raw_input("input:").strip() # 输入字符串变量
num = int(raw_input("input:").strip()) # 输入字符串 str 转为 int 型
locals() # 所有局部变量组成的字典
locals().values() # 所有局部变量值的列表
os.popen("date -d @{0} +'%Y-%m-%d %H:%M:%S'".format(12)).read() # 特殊情况引用变量 {0} 代表第一个参数

2.4 打印

```
# 字符串 %s  整数 %d  浮点 %f  原样打印 %r
print '字符串: %s 整数: %d 浮点: %f 原样打印: %r' % ('aa',2,1.0,'r')
print 'abc',          # 有逗号,代表不换行打印,在次打印会接着本行打印
```

2.5 列表

```
# 列表元素的个数最多 536870912
shoplist = ['apple', 'mango', 'carrot', 'banana']
shoplist[2] = 'aa'
del shoplist[0]
shoplist.insert('4','www')
shoplist.append('aaa')
shoplist[::-1]      # 倒着打印 对字符翻转串有效
shoplist[2::3]      # 从第二个开始每隔三个打印
shoplist[:-1]       # 排除最后一个
'\t'.join(li)       # 将列表转换成字符串
sys.path[1:1]=[5] # 在位置 1 前面插入列表中一个值
list(set(['qwe', 'as', '123', '123']))    # 将列表通过集合去重复
eval("['1','a']")                                  # 将字符串当表达式求值,得到列表
```

2.6 元组

```
# 不可变
zoo = ('wolf', 'elephant', 'penguin')
```

2.7 字典

```
ab = {          'Swaroop'   : 'swaroopch@byteofpython.info',
                'Larry'     : 'larry@wall.org',
        }
ab['c'] = 80          # 添加字典元素
del ab['Larry']     # 删除字典元素
```

```
ab.keys()              # 查看所有键值
ab.values()           # 打印所有值
ab.has_key('a')     # 查看键值是否存在
ab.items()            # 返回整个字典列表
```

复制字典

```
a = {1: {1: 2, 3: 4}}
b = a
b[1][1] = 8888                    # a 和 b 都为 {1: {1: 8888, 3: 4}}
import copy
c = copy.deepcopy(a)              # 再次赋值 b[1][1] = 9999 拷贝字典为新的字典,互不干扰

a[2] = copy.deepcopy(a[1])      # 复制出第二个 key，互不影响  {1: {1: 2, 3: 4},2: {1: 2, 3: 4}}
```

2.8 流程结构

2.8.1 if 判断

```
# 布尔值操作符 and or not 实现多重判断
if a == b:
    print '=='
elif a < b:
    print b
else:
    print a
fi
```

2.8.2 while 循环

```
while True:
    if a == b:
        print "=="
        break
    print "!="
```

```
else:
    print 'over'

count=0
while(count<9):
    print count
    count += 1
```

2.8.3 for 循环

```
sorted()              # 返回一个序列(列表)
zip()                 # 返回一个序列(列表)
enumerate()           # 返回循环列表序列  for i,v in enumerate(['a','b']):
reversed()            # 反序迭代器对象
dict.iterkeys()       # 通过键迭代
dict.itervalues()     # 通过值迭代
dict.iteritems()      # 通过键-值对迭代
randline()            # 文件迭代
iter(obj)             # 得到 obj 迭代器  检查 obj 是不是一个序列
iter(a,b)             # 重复调用 a,直到迭代器的下一个值等于 b
for i in range(1, 5):
    print i
else:
    print 'over'

list = ['a','b','c','b']
for i in range(len(list)):
    print list[i]
for x, Lee in enumerate(list):
    print "%d %s Lee" % (x+1,Lee)

# enumerate  使用函数得到索引值和对应值
for i, v in enumerate(['tic', 'tac', 'toe']):
    print(i, v)
```

2.8.4 流程结构简写

```
[ i * 2 for i in [8,-2,5]]
[16,-4,10]
```

```
[ i for i in range(8) if i %2 == 0 ]
[0,2,4,6]
```

2.9 tab 补全

```
# vim /usr/lib/python2.7/dist-packages/tab.py
# python startup file
import sys
import readline
import rlcompleter
import atexit
import os
# tab completion
readline.parse_and_bind('tab: complete')
# history file
histfile = os.path.join(os.environ['HOME'], '.pythonhistory')
```

2.10 函数

```
def printMax(a, b = 1):
    if a > b:
        print a
        return a
    else:
        print b
        return b
x = 5
y = 7
printMax(x, y)

def update(*args,**kwargs):
    p=''
    for i,t in kwargs.items():
            p = p+ '%s=%s,' %(i,str(t))
    sql = "update    'user' set (%s) where (%s)" %(args[0],p)
    print sql

update('aaa',uu='uu',id=3)
```

2.11 模块

```
# Filename: mymodule.py
def sayhi():
    print 'mymodule'
version = '0.1'

# 使用模块中方法
import mymodule
from mymodule import sayhi, version
mymodule.sayhi()    # 使用模块中函数方法
```

2.12 类对象的方法

```
class Person:
    # 实例化初始化的方法
    def __init__(self, name ,age):
        self.name = name
        self.age = age
        print self.name
    # 有 self 此函数为方法
    def sayHi(self):
        print 'Hello, my name is', self.name
    # 对象消逝的时候被调用
    def __del__(self):
        print 'over'
# 实例化对象
p = Person('Swaroop')
# 使用对象方法
p.sayHi()
# 继承
class Teacher(Person):
    def __init__(self, name, age, salary):
        Person.__init__(self, name, age)
        self.salary = salary
        print '(Initialized Teacher: %s)' % self.name
    def tell(self):
        Person.tell(self)
        print 'Salary: "%d"' % self.salary
t = Teacher('Mrs. Shrividya', 40, 30000)
```

2.13 执行模块类中的所有方法

```
# moniItems.py
import sys, time
import inspect

class mon:
    def __init__(self, n):
        self.name = n
        self.data = dict()
    def run(self):
        print 'hello', self.name
        return self.runAllGet()
    def getDisk(self):
        return 222
    def getCpu(self):
        return 111
    def runAllGet(self):
        for fun in inspect.getmembers(self, predicate=inspect.ismethod):
            print fun[0], fun[1]
            if fun[0][:3] == 'get':
                self.data[fun[0][3:]] = fun[1]()
        print self.data
        return self.data

# 模块导入使用
from moniItems import mon
m = mon()
m.runAllGet()
```

2.14 文件处理

```
# 模式: 读'r'  写[清空整个文件]'w' 追加[文件需要存在]'a' 读写'r+' 二进制文件'b'  'rb','wb','rb+'
```

2.14.1 写文件

```
i={'ddd':'ccc'}
f = file('poem.txt', 'a')
f.write("string")
f.write(str(i))
f.flush()
f.close()
```

2.14.2 读文件

```
f = file('/etc/passwd','r')
c = f.read().strip()            # 读取为一个大字符串，并去掉最后一个换行符
for i in c.spilt('\n'):        # 用换行符切割字符串得到列表循环每行
    print i
f.close()
```

读文件 1

```
f = file('/etc/passwd','r')
while True:
    line = f.readline()       # 返回一行
    if len(line) == 0:
        break
    x = line.split(":")                          # 冒号分割定义序列
    #x = [ x for x in line.split(":") ]   # 冒号分割定义序列
    #x = [ x.split("/") for x in line.split(":") ]   # 先冒号分割,在/分割 打印 x[6][1]
    print x[6],"\n",
f.close()
```

读文件 2

```
f = file('/etc/passwd')
c = f.readlines()           # 读入所有文件内容,可反复读取,大文件时占用内存较大
for line in c:
    print line.rstrip(),
f.close()
```

读文件 3

```
for i in open('b.txt'):    # 直接读取也可迭代,并有利于大文件读取,但不可反复读取
    print i,
```

2.14.3 追加日志

```
log = open('/home/peterli/xuesong','a')
print >> log,'faaa'
log.close()
```

2.14.4 with 读文件

```
with open('a.txt') as f:
    for i in f:
        print i
    print f.read()          # 打印所有内容为字符串
    print f.readlines()     # 打印所有内容按行分割的列表
```

2.14.5 csv 读配置文件

```
192.168.1.5,web # 配置文件按逗号分割
list = csv.reader(file('a.txt'))
for line in list:
    print line                  #   ['192.168.1.5', 'web']
```

2.15 内建函数

```
dir(sys)              # 显示对象的属性
help(sys)             # 交互式帮助
int(obj)              # 转型为整形
str(obj)              # 转为字符串
len(obj)              # 返回对象或序列长度
open(file,mode)       # 打开文件  #mode (r  读,w  写, a 追加)
range(0,3)            # 返回一个整形列表
raw_input("str:")     # 等待用户输入
type(obj)             # 返回对象类型
abs(-22)              # 绝对值
random                # 随机数
choice()              # 随机返回给定序列的一个元素
divmod(x,y)           # 函数完成除法运算，返回商和余数。
round(x[,n])          # 函数返回浮点数 x 的四舍五入值，如给出 n 值，则代表舍入
```

到小数点后的位数

strip()　　# 是去掉字符串两端多于空格,该句是去除序列中的所有字串两端多余的空格

del　　# 删除列表里面的数据

cmp(x,y)　　# 比较两个对象　　#根据比较结果返回一个整数，如果 x<y，则返回-1；如果 x>y，则返回 1,如果 x==y 则返回 0

max()　　# 字符串中最大的字符

min()　　# 字符串中最小的字符

sorted()　　# 对序列排序

reversed()　　# 对序列倒序

enumerate()　　# 返回索引位置和对应的值

sum()　　# 总和

list()　　# 变成列表可用于迭代

eval('3+4')　　# 将字符串当表达式求值 得到 7

exec 'a=100'　　# 将字符串按 python 语句执行

exec(a+'=new')　　# 将变量 a 的值作为新的变量

tuple()　　# 变成元组可用于迭代　　#一旦初始化便不能更改的数据结构,速度比 list 快

zip(s,t)　　# 返回一个合并后的列表　s = ['11','22']　t = ['aa','bb']　[('11', 'aa'), ('22', 'bb')]

isinstance(object,int)　　# 测试对象类型 int

xrange([lower,]stop[,step])　　# 函数与 range()类似，但 xrnage()并不创建列表，而是返回一个 xrange 对象

2.16　　字符串相关模块

string　　# 字符串操作相关函数和工具

re　　# 正则表达式

struct　　# 字符串和二进制之间的转换

c/StringIO　　# 字符串缓冲对象,操作方法类似于 file 对象

base64　　# Base16\32\64 数据编解码

codecs　　# 解码器注册和基类

crypt　　# 进行单方面加密

difflib　　# 找出序列间的不同

hashlib　　# 多种不同安全哈希算法和信息摘要算法的 API

hma　　# HMAC 信息鉴权算法的 python 实现

md5　　# RSA 的 MD5 信息摘要鉴权

rotor　　# 提供多平台的加解密服务

sha　　# NIAT 的安全哈希算法 SHA

stringprep　　# 提供用于 IP 协议的 Unicode 字符串

textwrap　　# 文本包装和填充

unicodedate　　# unicode 数据库

2.17 列表类型内建函数

```
list.append(obj)                        # 向列表中添加一个对象 obj
list.count(obj)                         # 返回一个对象 obj 在列表中出现的次数
list.extend(seq)                        # 把序列 seq 的内容添加到列表中
list.index(obj,i=0,j=len(list))   # 返回 list[k] == obj 的 k 值,并且 k 的范围在 i<=k<j;否则异常
list.insert(index.obj)                  # 在索引量为 index 的位置插入对象 obj
list.pop(index=-1)                      # 删除并返回指定位置的对象,默认是最后一个对象
list.remove(obj)                        # 从列表中删除对象 obj
list.reverse()                          # 原地翻转列表
list.sort(func=None,key=None,reverse=False)   # 以指定的方式排序列表中成员,如果 func 和 key 参数指定,则按照指定的方式比较各个元素,如果 reverse 标志被置为 True,则列表以反序排列
```

2.18 序列类型操作符

```
seq[ind]                # 获取下标为 ind 的元素
seq[ind1:ind2]          # 获得下标从 ind1 到 ind2 的元素集合
seq * expr              # 序列重复 expr 次
seq1 + seq2             # 连接 seq1 和 seq2
obj in seq              # 判断 obj 元素是否包含在 seq 中
obj not in seq          # 判断 obj 元素是否不包含在 seq 中
```

2.19 字符串类型内建方法

```
string.expandtabs(tabsize=8)                    # tab 符号转为空格 #默认 8 个空格
string.endswith(obj,beg=0,end=len(staring))     # 检测字符串是否已 obj 结束,如果是返回 True #如果 beg 或 end 指定检测范围是否已 obj 结束
string.count(str,beg=0,end=len(string))         # 检测 str 在 string 里出现次数 f.count('\n',0,len(f)) 判断文件行数
string.find(str,beg=0,end=len(string))          # 检测 str 是否包含在 string 中
string.index(str,beg=0,end=len(string))         # 检测 str 不在 string 中,会报异常
```

```
string.isalnum()                                    # 如果 string 至少有一个字符并且所有字符都是字母或数字则返回 True
string.isalpha()                                    # 如果 string 至少有一个字符并且所有字符都是字母则返回 True
string.isnumeric()                                  # 如果 string 只包含数字字符,则返回 True
string.isspace()                                    # 如果 string 包含空格则返回 True
string.isupper()                                    # 字符串都是大写返回 True
string.islower()                                    # 字符串都是小写返回 True
string.lower()                                      # 转换字符串中所有大写为小写
string.upper()                                      # 转换字符串中所有小写为大写
string.lstrip()                                     # 去掉 string 左边的空格
string.rstrip()                                     # 去掉 string 字符末尾的空格
string.replace(str1,str2,num=string.count(str1))   # 把 string 中的 str1 替换成 str2,如果 num 指定,则替换不超过 num 次
string.startswith(obj,beg=0,end=len(string))        # 检测字符串是否以 obj 开头
string.zfill(width)                                 # 返回字符长度为 width 的字符,原字符串右对齐,前面填充 0
string.isdigit()                                    # 只包含数字返回 True
string.split("分隔符")                               # 把 string 切片成一个列表
":".join(string.split())                            # 以:作为分隔符,将所有元素合并为一个新的字符串
```

2.20 序列类型相关的模块

```
array          # 一种受限制的可变序列类型,元素必须相同类型
copy           # 提供浅拷贝和深拷贝的能力
operator       # 包含函数调用形式的序列操作符 operator.concat(m,n)
re             # perl 风格的正则表达式查找
StringIO       # 把长字符串作为文件来操作 如: read() \ seek()
cStringIO      # 把长字符串作为文件来操,作速度更快,但不能被继承
textwrap       # 用作包装/填充文本的函数,也有一个类
types          # 包含 python 支持的所有类型
collections    # 高性能容器数据类型
```

2.21 字典内建方法

```
dict.clear()                                        # 删除字典中所有元素
```

```
dict copy()                                  # 返回字典(浅复制)的一个副本
dict.fromkeys(seq,val=None)                  # 创建并返回一个新字典,以 seq 中的元素做该字典的键,val 做该字典中所有键对的初始值
dict.get(key,default=None)                   # 对字典 dict 中的键 key,返回它对应的值 value,如果字典中不存在此键,则返回 default 值
dict.has_key(key)                            # 如果键在字典中存在,则返回 True 用 in 和 not in 代替
dicr.items()                                 # 返回一个包含字典中键、值对元组的列表
dict.keys()                                  # 返回一个包含字典中键的列表
dict.iter()                                  # 方法 iteritems()、iterkeys()、itervalues() 与它们对应的非迭代方法一样,不同的是它们返回一个迭代子,而不是一个列表
dict.pop(key[,default])                      # 和方法 get()相似.如果字典中 key 键存在,删除并返回 dict[key]
dict.setdefault(key,default=None)            # 和 set()相似,但如果字典中不存在 key 键,由 dict[key]=default 为它赋值
dict.update(dict2)                           # 将字典 dict2 的键值对添加到字典 dict
dict.values()                                # 返回一个包含字典中所有值得列表

dict([container])        # 创建字典的工厂函数。提供容器类(container),就用其中的条目填充字典
len(mapping)             # 返回映射的长度(键-值对的个数)
hash(obj)                # 返回 obj 哈希值,判断某个对象是否可做一个字典的键值
```

2.22 集合方法

```
s.update(t)                                  # 用 t 中的元素修改 s,s 现在包含 s 或 t 的成员     s |= t
s.intersection_update(t)                     # s 中的成员是共用属于 s 和 t 的元素 s &= t
s.difference_update(t)                       # s 中的成员是属于 s 但不包含在 t 中的元素 s -= t
s.symmetric_difference_update(t)             # s 中的成员更新为那些包含在 s 或 t 中,但不是 s 和 t 共有的元素    s ^= t
s.add(obj)                                   # 在集合 s 中添加对象 obj
s.remove(obj)                                # 从集合 s 中删除对象 obj;如果 obj 不是集合 s 中的元素(obj not in s),将引发 KeyError 错误
s.discard(obj)                               # 如果 obj 是集合 s 中的元素,从集合 s 中删除对象 obj
s.pop()                                      # 删除集合 s 中的任意一个对象,并返回它
s.clear()                                    # 删除集合 s 中的所有元素
```

```
s.issubset(t)                    # 如果 s 是 t 的子集,则返回 True      s <= t
s.issuperset(t)                  # 如果 t 是 s 的超集,则返回 True      s >= t
s.union(t)                       # 合并操作;返回一个新集合,该集合是 s 和 t 的并集     s | t
s.intersection(t)                # 交集操作;返回一个新集合,该集合是 s 和 t 的交集     s & t
s.difference(t)                  # 返回一个新集合,改集合是 s 的成员,但不是 t 的成员    s - t
s.symmetric_difference(t)        # 返回一个新集合,该集合是 s 或 t 的成员,但不是 s 和 t 共有的成员      s ^ t
s.copy()                         # 返回一个新集合,它是集合 s 的浅复制
obj in s                         # 成员测试;obj 是 s 中的元素  返回 True
obj not in s                     # 非成员测试:obj 不是 s 中元素  返回 True
s == t                           # 等价测试  是否具有相同元素
s != t                           # 不等价测试
s < t                            # 子集测试;s!=t 且 s 中所有元素都是 t 的成员
s > t                            # 超集测试;s!=t 且 t 中所有元素都是 s 的成员
```

2.23 序列化

```
#!/usr/bin/python
import cPickle
obj = {'1':['4124','1241','124'],'2':['12412','142','1241']}

pkl_file = open('account.pkl','wb')
cPickle.down(obj,pkl_file)
pkl_file.close()

pkl_file = open('account.pkl','rb')
account_list = cPickle.load(pkl_file)
pkl_file.close()
```

2.24 文件对象方法

```
file.close()                     # 关闭文件
file.fileno()                    # 返回文件的描述符
file.flush()                     # 刷新文件的内部缓冲区
```

```
file.isatty()                    # 判断 file 是否是一个类 tty 设备
file.next()                      # 返回文件的下一行,或在没有其他行时引发 StopIteration 异常
file.read(size=-1)               # 从文件读取 size 个字节,当未给定 size 或给定负值的时候,读取剩余的所有字节,然后作为字符串返回
file.readline(size=-1)           # 从文件中读取并返回一行(包括行结束符),或返回最大 size 个字符
file.readlines(sizhint=0)        # 读取文件的所有行作为一个列表返回
file.xreadlines()                # 用于迭代,可替换 readlines()的一个更高效的方法
file.seek(off, whence=0)         # 在文件中移动文件指针,从 whence(0 代表文件起始,1 代表当前位置,2 代表文件末尾)偏移 off 字节
file.tell()                      # 返回当前在文件中的位置
file.truncate(size=file.tell())  # 截取文件到最大 size 字节,默认为当前文件位置
file.write(str)                  # 向文件写入字符串
file.writelines(seq)             # 向文件写入字符串序列 seq;seq 应该是一个返回字符串的可迭代对象
```

2.25 文件对象的属性

```
file.closed          # 表示文件已被关闭,否则为 False
file.encoding        # 文件所使用的编码  当 unicode 字符串被写入数据时,它将自动使用 file.encoding 转换为字节字符串;若 file.encoding 为 None 时使用系统默认编码
file.mode            # Access 文件打开时使用的访问模式
file.name            # 文件名
file.newlines        # 未读取到行分隔符时为 None,只有一种行分隔符时为一个字符串,当文件有多种类型的行结束符时,则为一个包含所有当前所遇到的行结束符的列表
file.softspace       # 为 0 表示在输出一数据后,要加上一个空格符,1 表示不加
```

2.26 异常处理

```
# try 中使用 sys.exit(2) 会被捕获,无法退出脚本,可使用 os._exit(2) 退出脚本

class ShortInputException(Exception):    # 继承 Exception 异常的类,定义自己的异常
    def __init__(self, length, atleast):
        Exception.__init__(self)
        self.length = length
        self.atleast = atleast
try:
```

```
        s = raw_input('Enter something --> ')
        if len(s) < 3:
            raise ShortInputException(len(s), 3)       # 触发异常
    except EOFError:
        print '\nWhy did you do an EOF on me?'
    except ShortInputException, x:          # 捕捉指定错误信息
        print 'ShortInputException:    %d | %d' % (x.length, x.atleast)
    except Exception as err:                 # 捕捉所有其它错误信息内容
        print str(err)
    #except urllib2.HTTPError as err:     # 捕捉外部导入模块的错误
    #except:                                    # 捕捉所有其它错误 不会看到错误内容
    #        print 'except'
    finally:                                   # 无论什么情况都会执行 关闭文件或断开连接等
        print 'finally'
    else:                                      # 无任何异常 无法和 finally 同用
        print 'No exception was raised.'
```

2.26.1 不可捕获的异常

```
NameError:                  # 尝试访问一个未申明的变量
ZeroDivisionError:         # 除数为零
SyntaxErrot:               # 解释器语法错误
IndexError:                # 请求的索引元素超出序列范围
KeyError:                  # 请求一个不存在的字典关键字
IOError:                   # 输入/输出错误
AttributeError:            # 尝试访问未知的对象属性
ImportError                # 没有模块
IndentationError           # 语法缩进错误
KeyboardInterrupt          # ctrl+C
SyntaxError                # 代码语法错误
ValueError                 # 值错误
TypeError                  # 传入对象类型与要求不符合
```

2.26.2 内建异常

```
BaseException               # 所有异常的基类
SystemExit                  # python 解释器请求退出
KeyboardInterrupt           # 用户中断执行
```

```
Exception                   # 常规错误的基类
StopIteration               # 迭代器没有更多的值
GeneratorExit               # 生成器发生异常来通知退出
StandardError               # 所有的内建标准异常的基类
ArithmeticError             # 所有数值计算错误的基类
FloatingPointError          # 浮点计算错误
OverflowError               # 数值运算超出最大限制
AssertionError              # 断言语句失败
AttributeError              # 对象没有这个属性
EOFError                    # 没有内建输入,到达 EOF 标记
EnvironmentError            # 操作系统错误的基类
IOError                     # 输入/输出操作失败
OSError                     # 操作系统错误
WindowsError                # windows 系统调用失败
ImportError                 # 导入模块/对象失败
KeyboardInterrupt           # 用户中断执行(通常是 ctrl+c)
LookupError                 # 无效数据查询的基类
IndexError                  # 序列中没有此索引(index)
KeyError                    # 映射中没有这个键
MemoryError                 # 内存溢出错误(对于 python 解释器不是致命的)
NameError                   # 未声明/初始化对象(没有属性)
UnboundLocalError           # 访问未初始化的本地变量
ReferenceError              # 若引用试图访问已经垃圾回收了的对象
RuntimeError                # 一般的运行时错误
NotImplementedError         # 尚未实现的方法
SyntaxError                 # python 语法错误
IndentationError            # 缩进错误
TabError                    # tab 和空格混用
SystemError                 # 一般的解释器系统错误
TypeError                   # 对类型无效的操作
ValueError                  # 传入无效的参数
UnicodeError                # Unicode 相关的错误
UnicodeDecodeError          # Unicode 解码时的错误
UnicodeEncodeError          # Unicode 编码时的错误
UnicodeTranslateError       # Unicode 转换时错误
Warning                     # 警告的基类
DeprecationWarning          # 关于被弃用的特征的警告
FutureWarning               # 关于构造将来语义会有改变的警告
OverflowWarning             # 旧的关于自动提升为长整形的警告
PendingDeprecationWarning   # 关于特性将会被废弃的警告
RuntimeWarning              # 可疑的运行时行为的警告
SyntaxWarning               # 可疑的语法的警告
UserWarning                 # 用户代码生成的警告
```

2.26.3 触发异常

```
raise  exclass                  # 触发异常,从 exclass 生成一个实例(不含任何异常参数)
raise  exclass()                # 触发异常,但现在不是类;通过函数调用操作符(function calloperator:"()")作用于类名生成一个新的 exclass 实例,同样也没有异常参数
raise exclass, args         # 触发异常,但同时提供的异常参数 args,可以是一个参数也可以是元组
raise exclass(args)         # 触发异常,同上
raise exclass, args, tb   # 触发异常,但提供一个跟踪记录(traceback)对象 tb 供使用
raise exclass,instance     # 通过实例触发异常(通常是 exclass 的实例)
raise  instance                 # 通过实例触发异常;异常类型是实例的类型:等价于 raise instance.__class__, instance
raise string                  # 触发字符串异常
raise string, srgs          # 触发字符串异常,但触发伴随着 args
raise  string,args,tb       # 触发字符串异常,但提供一个跟踪记录(traceback)对象 tb 供使用
raise                           # 重新触发前一个异常,如果之前没有异常,触发 TypeError
```

2.26.4 跟踪异常栈

```
# traceback  获取异常相关数据都是通过 sys.exc_info()函数得到的
import traceback
import sys
try:
    s = raw_input()
    print int(s)
except ValueError:
    # sys.exc_info() 返回值是元组，第一个 exc_type 是异常的对象类型，exc_value 是异常的值，exc_tb 是一个 traceback 对象，对象中包含出错的行数、位置等数据
    exc_type, exc_value, exc_tb = sys.exc_info()
    print "\n%s \n %s \n %s\n" %(exc_type, exc_value, exc_tb )
    traceback.print_exc()           # 打印栈跟踪信息
```

2.27 抓取全部错误信息存如字典

```
import sys, traceback

try:
    s = raw_input()
    int(s)
except:
    exc_type, exc_value, exc_traceback = sys.exc_info()
    traceback_details = {
                         'filename':
exc_traceback.tb_frame.f_code.co_filename,
                         'lineno'    : exc_traceback.tb_lineno,
                         'name'                                    :
exc_traceback.tb_frame.f_code.co_name,
                         'type'       : exc_type.__name__,
                         'message' : exc_value.message,
                        }

    del(exc_type, exc_value, exc_traceback)
    print traceback_details
    f = file('test1.txt', 'a')

f.write("%s %s %s %s %s\n" %(traceback_details['filename'],traceback_details['lineno'],trace
back_details['name'],traceback_details['type'],traceback_details['message'], ))
    f.flush()
    f.close()
```

调试 log

```
# cgitb 覆盖了默认 sys.excepthook 全局异常拦截器
def func(a, b):
    return a / b
if __name__ == '__main__':
    import cgitb
    cgitb.enable(format='text')
    func(1, 0)
```

2.28 函数式编程的内建函数

apply(func[,nkw][,kw])　　# 用可选的参数来调用 func,nkw 为非关键字参数,kw 为关键字参数;返回值是函数调用的返回值

filter(func,seq)　　# 调用一个布尔函数 func 来迭代遍历每个 seq 中的元素;返回一个使 func 返回值为 true 的元素的序列

map(func,seq1[,seq2])　　# 将函数 func 作用于给定序列(s)的每个元素,并用一个列表来提供返回值;如果 func 为 None,func 表现为一个身份函数,返回一个含有每个序列中元素集合的 n 个元组的列表

reduce(func,seq[,init])　　# 将二元函数作用于 seq 序列的元素,每次携带一堆(先前的结果以及下一个序列元素),连续地将现有的结果和下一个值作用在获得的随后的结果上,最后减少我们的序列为一个单一的返回值;如果初始值 init 给定,第一个比较会是 init 和第一个序列元素而不是序列的头两个元素

```
# filter 即通过函数方法只保留结果为真的值组成列表
def f(x): return x % 2 != 0 and x % 3 != 0
f(3)　　# 函数结果是 False　3 被 filter 抛弃
f(5)　　# 函数结果是 True　 5 被加入 filter 最后的列表结果
filter(f, range(2, 25))
[5, 7, 11, 13, 17, 19, 23]

# map 通过函数对列表进行处理得到新的列表
def cube(x): return x*x*x
map(cube, range(1, 11))
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

reduce 通过函数会先接收初始值和序列的第一个元素，然后是返回值和下一个元素，依此类推

```
def add(x,y): return x+y
```

reduce(add, range(1, 11))　　# 结果 55　是 1 到 10 的和　x 的值是上一次函数返回的结果，y 是列表中循环的值

2.29　re 正则

compile(pattern,flags=0)　　# 对正则表达式模式 pattern 进行编译,flags 是可选标识符,并返回一个 regex 对象

match(pattern,string,flags=0)　　# 尝试用正则表达式模式 pattern 匹配字符串 string,flags 是可选标识符,如果匹配成功,则返回一个匹配对象;否则返回 None

search(pattern,string,flags=0)　　# 在字符串 string 中搜索正则表达式模式 pattern 的第一次出现,flags 是可选标识符,如果匹配成功,则返回一个匹配对象;否则返回 None

findall(pattern,string[,flags])　　# 在字符串 string 中搜索正则表达式模式 pattern 的所有(非重复)出现:返回一个匹配对象的列表　# pattern=u'\u4e2d\u6587'　代表 UNICODE

finditer(pattern,string[,flags])　　# 和 findall()相同,但返回的不是列表而是迭代器;对于每个匹配,该迭代器返回一个匹配对象

split(pattern,string,max=0)　　# 根据正则表达式 pattern 中的分隔符把字符 string 分割为一个列表,返回成功匹配的列表,最多分割 max 次(默认所有)

sub(pattern,repl,string,max=0)　　# 把字符串 string 中所有匹配正则表达式 pattern 的地方替换成字符串 repl,如果 max 的值没有给出,则对所有匹配的地方进行替换(subn()会返回一个表示替换次数的数值)

group(num=0)　　# 返回全部匹配对象(或指定编号是 num 的子组)

groups()　　# 返回一个包含全部匹配的子组的元组(如果没匹配成功,返回一个空元组)

例子

```
re.findall(r'a[be]c','123abc456eaec789')　　# 返回匹配对象列表 ['abc', 'aec']
re.findall("(.)12[34](..)",a)　　# 取出匹配括号中内容
a='qedqwe123dsf'
re.search("(.)123",a ).group(1)　　# 搜索匹配的取第 1 个标签
re.match("^(1|2) *(.*) *abc$", str).group(2)　　# 取第二个标签
re.match("^(1|2) *(.*) *abc$", str).groups()　　# 取所有标签
re.sub('[abc]','A','alex')　　# 替换
for i in re.finditer(r'\d+',s):　　# 迭代
    print i.group(),i.span()　　#
```

搜索网页中 UNICODE 格式的中文

```
QueryAdd='http://www.anti-spam.org.cn/Rbl/Query/Result'
Ip='222.129.184.52'
s = requests.post(url=QueryAdd, data={'IP':Ip})
re.findall(u'\u4e2d\u56fd', s.text, re.S)
```

2.30　编码转换

```
a='中文'　　# 编码未定义按输入终端 utf8 或 gbk
u=u'中文'　　# 定义为 unicode 编码　u 值为 u'\u4e2d\u6587'
u.encode('utf8')　　# 转为 utf8 格式 u 值为 '\xe4\xb8\xad\xe6\x96\x87'
print u　　# 结果显示 中文
print u.encode('utf8')　　# 转为 utf8 格式,当显示终端编码为 utf8　结果显示 中文　编码不一致则乱码
print u.encode('gbk')　　# 当前终端为 utf8 故乱码
ord('4')　　# 字符转 ASCII 码
chr(52)　　# ASCII 码转字符
```

遍历递归

```
[os.path.join(x[0],y) for x in os.walk('/root/python/5') for y in x[2]]

for i in os.walk('/root/python/5/work/server'):
    print i
```

3 2 常用模块

3.1 sys

```
sys.argv                 # 取参数列表
sys.exit(2)              # 退出脚本返回状态 会被 try 截取
sys.exc_info()           # 获取当前正在处理的异常类
sys.version              # 获取 Python 解释程序的版本信息
sys.maxint               # 最大的 Int 值   9223372036854775807
sys.maxunicode           # 最大的 Unicode 值
sys.modules              # 返回系统导入的模块字段，key 是模块名，value 是模块
sys.path                 # 返回模块的搜索路径，初始化时使用 PYTHONPATH 环境
变量的值
sys.platform             # 返回操作系统平台名称
sys.stdout               # 标准输出
sys.stdin                # 标准输入
sys.stderr               # 错误输出
sys.exec_prefix          # 返回平台独立的 python 文件安装的位置
sys.stdin.readline()   # 从标准输入读一行
sys.stdout.write("a") # 屏幕输出 a
```

3.2 os

```
# 相对 sys 模块 os 模块更为底层 os._exit() try 无法抓取
os.popen('id').read()        # 执行系统命令得到返回结果
os.system()                  # 得到返回状态 返回无法截取
os.name                      # 返回系统平台 Linux/Unix 用户是'posix'
os.getenv()                  # 读取环境变量
os.putenv()                  # 设置环境变量
os.getcwd()                  # 当前工作路径
```

```
os.chdir()                  # 改变当前工作目录
os.walk('/root/')           # 递归路径
```

3.2.1 文件处理

```
mkfifo()/mknod()        # 创建命名管道/创建文件系统节点
remove()/unlink()       # 删除文件
rename()/renames()      # 重命名文件
*stat()                 # 返回文件信息
symlink()               # 创建符号链接
utime()                 # 更新时间戳
tmpfile()               # 创建并打开('w+b')一个新的临时文件
walk()                  # 遍历目录树下的所有文件名
```

3.2.2 目录/文件夹

```
chdir()/fchdir()        # 改变当前工作目录/通过一个文件描述符改变当前工作目录
chroot()                # 改变当前进程的根目录
listdir()               # 列出指定目录的文件
getcwd()/getcwdu()      # 返回当前工作目录/功能相同,但返回一个 unicode 对象
mkdir()/makedirs()      # 创建目录/创建多层目录
rmdir()/removedirs()    # 删除目录/删除多层目录
```

3.2.3 访问/权限

```
saccess()               # 检验权限模式
chmod()                 # 改变权限模式
chown()/lchown()        # 改变 owner 和 groupID 功能相同,但不会跟踪链接
umask()                 # 设置默认权限模式
```

3.2.4 文件描述符操作

```
open()                  # 底层的操作系统 open(对于稳健,使用标准的内建 open()函数)
read()/write()          # 根据文件描述符读取/写入数据 按大小读取文件部分内容
```

dup()/dup2() # 复制文件描述符号/功能相同,但是复制到另一个文件描述符

3.2.5 设备号

```
makedev()  # 从 major 和 minor 设备号创建一个原始设备号
major()/minor()  # 从原始设备号获得 major/minor 设备号
```

os.path 模块

```
os.path.expanduser('~/.ssh/key')  # 家目录下文件的全路径
```

3.2.6 分隔

```
os.path.basename()  # 去掉目录路径,返回文件名
os.path.dirname()  # 去掉文件名,返回目录路径
os.path.join()  # 将分离的各部分组合成一个路径名
os.path.spllt()  # 返回(dirname(),basename())元组
os.path.splitdrive()  # 返回(drivename,pathname)元组
os.path.splitext()  # 返回(filename,extension)元组
```

3.2.7 信息

```
os.path.getatime()  # 返回最近访问时间
os.path.getctime()  # 返回文件创建时间
os.path.getmtime()  # 返回最近文件修改时间
os.path.getsize()  # 返回文件大小(字节)
```

3.2.8 查询

```
os.path.exists()  # 指定路径(文件或目录)是否存在
os.path.isabs()  # 指定路径是否为绝对路径
os.path.isdir()  # 指定路径是否存在且为一个目录
os.path.isfile()  # 指定路径是否存在且为一个文件
os.path.islink()  # 指定路径是否存在且为一个符号链接
os.path.ismount()  # 指定路径是否存在且为一个挂载点
os.path.samefile()  # 两个路径名是否指向同一个文件
```

3.2.9 相关模块

```
base64                  # 提供二进制字符串和文本字符串间的编码/解码操作
binascii                # 提供二进制和 ASCII 编码的二进制字符串间的编码/解码操作
bz2                     # 访问 BZ2 格式的压缩文件
csv                     # 访问 csv 文件(逗号分隔文件)
csv.reader(open(file))
filecmp                 # 用于比较目录和文件
fileinput               # 提供多个文本文件的行迭代器
getopt/optparse         # 提供了命令行参数的解析/处理
glob/fnmatch            # 提供 unix 样式的通配符匹配的功能
gzip/zlib               # 读写 GNU zip(gzip)文件(压缩需要 zlib 模块)
shutil                  # 提供高级文件访问功能
c/StringIO              # 对字符串对象提供类文件接口
tarfile                 # 读写 TAR 归档文件,支持压缩文件
tempfile                # 创建一个临时文件
uu                      # uu 格式的编码和解码
zipfile                 # 用于读取 zip 归档文件的工具
environ['HOME']         # 查看系统环境变量
```

3.2.10 子进程

```
os.fork()       # 创建子进程,并复制父进程所有操作  通过判断 pid = os.fork() 的 pid 值,分别执行父进程与子进程操作，0 为子进程
os.wait()       # 等待子进程结束
```

跨平台 os 模块属性

```
linesep         # 用于在文件中分隔行的字符串
sep             # 用来分隔文件路径名字的字符串
pathsep         # 用于分割文件路径的字符串
curdir          # 当前工作目录的字符串名称
pardir          # 父目录字符串名称
```

commands

```
commands.getstatusoutput('id')      # 返回元组(状态,标准输出)
commands.getoutput('id')            # 只返回执行的结果，忽略返回值
commands.getstatus('file')          # 返回 ls -ld file 执行的结果
```

3.3 文件和目录管理

```
import shutil
shutil.copyfile('data.db', 'archive.db')                    # 拷贝文件
shutil.move('/build/executables', 'installdir')          # 移动文件或目录
```

3.4 文件通配符

```
import glob
glob.glob('*.py')       # 查找当前目录下 py 结尾的文件
```

3.5 随机模块

```
import random
random.choice(['apple', 'pear', 'banana'])     # 随机取列表一个参数
random.sample(xrange(100), 10)   # 不重复抽取 10 个
random.random()                              # 随机浮点数
random.randrange(6)                        # 随机整数范围
```

3.6 发送邮件

3.6.1 发送邮件内容

```
#!/usr/bin/python
#encoding:utf8
# 导入 smtplib 和 MIMEText
import smtplib
from email.mime.text import MIMEText

# 定义发送列表
mailto_list=["272121935@qq.com","272121935@163.com"]
```

```
# 设置服务器名称、用户名、密码以及邮件后缀
mail_host = "smtp.163.com"
mail_user = "mailuser"
mail_pass = "password"
mail_postfix="163.com"

# 发送邮件函数
def send_mail(to_list, sub):
    me = mail_user + "<"+mail_user+"@"+mail_postfix+">"
    fp = open('context.txt')
    msg = MIMEText(fp.read(),_charset="utf-8")
    fp.close()
    msg['Subject'] = sub
    msg['From'] = me
    msg['To'] = ";".join(to_list)
    try:
        send_smtp = smtplib.SMTP()
        send_smtp.connect(mail_host)
        send_smtp.login(mail_user, mail_pass)
        send_smtp.sendmail(me, to_list, msg.as_string())
        send_smtp.close()
        return True
    except Exception, e:
        print str(e)
        return False

if send_mail(mailto_list,"标题"):
    print "测试成功"
else:
    print "测试失败"
```

3.6.2 发送附件

```
#!/usr/bin/python
#encoding:utf8
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders

def send_mail(to_list, sub, filename):
    me = mail_user + "<"+mail_user+"@"+mail_postfix+">"
```

```
        msg = MIMEMultipart()
        msg['Subject'] = sub
        msg['From'] = me
        msg['To'] = ";".join(to_list)
        submsg = MIMEBase('application', 'x-xz')
        submsg.set_payload(open(filename,'rb').read())
        encoders.encode_base64(submsg)
        submsg.add_header('Content-Disposition', 'attachment', filename=filename)
        msg.attach(submsg)
        try:
                send_smtp = smtplib.SMTP()
                send_smtp.connect(mail_host)
                send_smtp.login(mail_user, mail_pass)
                send_smtp.sendmail(me, to_list, msg.as_string())
                send_smtp.close()
                return True
        except Exception, e:
                print str(e)[1]
                return False

# 设置服务器名称、用户名、密码以及邮件后缀
mail_host = "smtp.163.com"
mail_user = "xuesong"
mail_pass = "mailpasswd"
mail_postfix = "163.com"
mailto_list = ["272121935@qq.com","quanzhou722@163.com"]
title = 'check'
filename = 'file_check.html'
if send_mail(mailto_list,title,filename):
        print "发送成功"
else:
        print "发送失败"
```

3.7 解压缩

3.7.1 gzip 压缩

```
import gzip
f_in = open('file.log', 'rb')
```

```
f_out = gzip.open('file.log.gz', 'wb')
f_out.writelines(f_in)
f_out.close()
f_in.close()
```

gzip 压缩 1

```
File = 'xuesong_18.log'
g = gzip.GzipFile(filename="", mode='wb', compresslevel=9, fileobj=open((r'%s.gz' %File),'wb'))
g.write(open(r'%s' %File).read())
g.close()
```

gzip 解压

```
g = gzip.GzipFile(mode='rb', fileobj=open((r'xuesong_18.log.gz'),'rb'))
open((r'xuesong_18.log'),'wb').write(g.read())
```

压缩 tar.gz

```
import os
import tarfile
tar = tarfile.open("/tmp/tartest.tar.gz","w:gz")    # 创建压缩包名
for path,dir,files in os.walk("/tmp/tartest"):      # 递归文件目录
    for file in files:
        fullpath = os.path.join(path,file)
        tar.add(fullpath)                           # 创建压缩包
tar.close()
```

3.7.2 解压 tar.gz

```
import tarfile
tar = tarfile.open("/tmp/tartest.tar.gz")
#tar.extract("/tmp")                                # 全部解压到指定路径
names = tar.getnames()                              # 包内文件名
for name in names:
    tar.extract(name,path="./")                     # 解压指定文件
tar.close()
```

3.7.3 zip 压缩

```
import zipfile,os
f = zipfile.ZipFile('filename.zip', 'w' ,zipfile.ZIP_DEFLATED)      # ZIP_STORE 为默认表不压缩. ZIP_DEFLATED 表压缩
#f.write('file1.txt')                                               # 将文件写入压缩包
for path,dir,files in os.walk("tartest"):                          # 递归压缩目录
    for file in files:
        f.write(os.path.join(path,file))                           # 将文件逐个写入压缩包
f.close()
```

zip 解压

```
if zipfile.is_zipfile('filename.zip'):                             # 判断一个文件是不是 zip 文件
    f = zipfile.ZipFile('filename.zip')
    for file in f.namelist():                                      # 返回文件列表
        f.extract(file, r'/tmp/')                                  # 解压指定文件
    #f.extractall()                                                # 解压全部
    f.close()
```

3.8 时间

```
import time
time.time()                                                        # 时间戳[浮点]
time.localtime()[1] - 1                                            # 上个月
int(time.time())                                                   # 时间戳[整 s]
tomorrow.strftime('%Y%m%d_%H%M')                                   # 格式化时间
time.strftime('%Y-%m-%d_%X',time.localtime( time.time() ) )        # 时间戳转日期
time.mktime(time.strptime('2012-03-28 06:53:40', '%Y-%m-%d %H:%M:%S'))    # 日期转时间戳
```

判断输入时间格式是否正确

```
#encoding:utf8
import time
while 1:
    atime=raw_input('输入格式如[14.05.13 13:00]:')
    try:
        btime=time.mktime(time.strptime('%s:00' %atime, '%y.%m.%d %H:%M:%S'))
        break
```

```
                except:
                    print '时间输入错误,请重新输入，格式如[14.05.13 13:00]'
```

上一个月最后一天

```
    import datetime

lastMonth=datetime.date(datetime.date.today().year,datetime.date.today().month,1)-datetime.timedelta(1)
    lastMonth.strftime("%Y/%m")
```

前一天

```
    (datetime.datetime.now() + datetime.timedelta(days=-1) ).strftime('%Y%m%d')
```

两日期相差天数

```
    import datetime
    d1 = datetime.datetime(2005, 2, 16)
    d2 = datetime.datetime(2004, 12, 31)
    (d1 - d2).days
```

向后加 10 个小时

```
    import datetime
    d1 = datetime.datetime.now()
    d3 = d1 + datetime.timedelta(hours=10)
    d3.ctime()
```

参数[optparse]

```
    import os, sys
    import time
    import optparse
    # python aaa.py -t file -p /etc/opt -o aaaaa

    def do_fiotest( type, path, output,):
        print type, path, output,

    def main():
        parser = optparse.OptionParser()
        parser.add_option('-t', '--type', dest = 'type', default = None, help = 'test type[file, device]')
        parser.add_option('-p', '--path', dest = 'path', default = None, help = 'test file path or device path')
        parser.add_option('-o', '--output', dest = 'output', default = None, help = 'result dir path')
```

```
        (o, a) = parser.parse_args()

        if None == o.type or None == o.path or None == o.output:
            print "No device or file or output dir"
            return -1

        if 'file' != o.type and 'device' != o.type:
            print "You need specify test type ['file' or 'device']"
            return -1

        do_fiotest(o.type, o.path, o.output)
        print "Test done!"


    if __name__ == '__main__':
        main()
```

hash

```
    import md5
    m = md5.new('123456').hexdigest()

    import hashlib
    m = hashlib.md5()
    m.update("Nobody inspects")       # 使用 update 方法对字符串 md5 加密
    m.digest()                        # 加密后二进制结果
    m.hexdigest()                     # 加密后十进制结果
    hashlib.new("md5", "string").hexdigest()                    # 对字符串加密
    hashlib.new("md5", open("file").read()).hexdigest()       # 查看文件 MD5 值
```

3.9 隐藏输入密码

```
    import getpass
    passwd=getpass.getpass()
```

3.10 string 打印 a-z

```
    import string
    string.lowercase            # a-z 小写
```

string.uppercase # A-Z 大小

3.11 paramiko [ssh 客户端]

安装

```
sudo apt-get install python-setuptools
easy_install
sudo apt-get install python-all-dev
sudo apt-get install build-essential
```

paramiko 实例(账号密码登录执行命令)

```
#!/usr/bin/python
#ssh
import paramiko
import sys,os

host = '10.152.15.200'
user = 'peterli'
password = '123456'

s = paramiko.SSHClient()                                    # 绑定实例
s.load_system_host_keys()                                   # 加载本地HOST 主机文件
s.set_missing_host_key_policy(paramiko.AutoAddPolicy())     # 允许连接不在know_hosts 文件中的主机
s.connect(host,22,user,password,timeout=5)                  # 连接远程主机
while True:
        cmd=raw_input('cmd:')
        stdin,stdout,stderr = s.exec_command(cmd)           # 执行命令
        cmd_result = stdout.read(),stderr.read()            # 读取命令结果
        for line in cmd_result:
                print line,
s.close()
```

paramiko 实例(传送文件)

```
#!/usr/bin/evn python
import os
import paramiko
host='127.0.0.1'
port=22
```

```
username = 'peterli'
password = '123456'
ssh=paramiko.Transport((host,port))
privatekeyfile = os.path.expanduser('~/.ssh/id_rsa')
mykey = paramiko.RSAKey.from_private_key_file(  os.path.expanduser('~/.ssh/id_rsa'))    # 加载 key 不使用 key 可不加
ssh.connect(username=username,password=password)                # 连接远程主机
# 使用 key 把 password=password 换成 pkey=mykey
sftp=paramiko.SFTPClient.from_transport(ssh)                    # SFTP 使用 Transport 通道
sftp.get('/etc/passwd','pwd1')                                  # 下载 两端都要指定文件名
sftp.put('pwd','/tmp/pwd')                                      # 上传
sftp.close()
ssh.close()
```

paramiko 实例(密钥执行命令)

```
#!/usr/bin/python
#ssh
import paramiko
import sys,os
host = '10.152.15.123'
user = 'peterli'
s = paramiko.SSHClient()
s.load_system_host_keys()
s.set_missing_host_key_policy(paramiko.AutoAddPolicy())
privatekeyfile  =  os.path.expanduser('~/.ssh/id_rsa')          # 定义 key 路径
mykey = paramiko.RSAKey.from_private_key_file(privatekeyfile)
#
mykey=paramiko.DSSKey.from_private_key_file(privatekeyfile,password='061128')      # DSSKey 方式 password 是 key 的密码
s.connect(host,22,user,pkey=mykey,timeout=5)
cmd=raw_input('cmd:')
stdin,stdout,stderr = s.exec_command(cmd)
cmd_result = stdout.read(),stderr.read()
for line in cmd_result:
        print line,
s.close()
```

ssh 并发(Pool 控制最大并发)

```
#!/usr/bin/env python
#encoding:utf8
#ssh_concurrent.py

import multiprocessing
import sys,os,time
import paramiko

def ssh_cmd(host,port,user,passwd,cmd):
    msg = "----------Result:%s----------" % host

    s = paramiko.SSHClient()
    s.load_system_host_keys()
    s.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        s.connect(host,22,user,passwd,timeout=5)
        stdin,stdout,stderr = s.exec_command(cmd)

        cmd_result = stdout.read(),stderr.read()
        print msg
        for line in cmd_result:
                print line,

        s.close()
    except paramiko.AuthenticationException:
        print msg
        print 'AuthenticationException Failed'
    except paramiko.BadHostKeyException:
        print msg
        print "Bad host key"

result = []
p = multiprocessing.Pool(processes=20)
cmd=raw_input('CMD:')
f=open('serverlist.conf')
list = f.readlines()
f.close()
for IP in list:
    print IP
    host=IP.split()[0]
    port=int(IP.split()[1])
    user=IP.split()[2]
    passwd=IP.split()[3]
```

```
            result.append(p.apply_async(ssh_cmd,(host,port,user,passwd,cmd)))

        p.close()

        for res in result:
            res.get(timeout=35)
```

ssh 并发(取文件状态并发送邮件)

```
        #!/usr/bin/python
        #encoding:utf8
        #config file: ip.list

        import paramiko
        import multiprocessing
        import smtplib
        import sys,os,time,datetime,socket,re
        from email.mime.text import MIMEText

        #  配置文件(IP 列表)
        Conf = 'ip.list'
        user_name = 'peterli'
        user_pwd = 'passwd'
        port = 22
        PATH = '/home/peterli/'

        #  设置服务器名称、用户名、密码以及邮件后缀
        mail_host = "smtp.163.com"
        mail_user = "xuesong"
        mail_pass = "mailpasswd"
        mail_postfix = "163.com"
        mailto_list = ["272121935@qq.com","quanzhou722@163.com"]
        title = 'file check'

        DATE1=(datetime.datetime.now()                                          +
datetime.timedelta(days=-1) ).strftime('%Y%m%d')
        file_path = '%s%s' %(PATH,DATE1)

        def Ssh_Cmd(file_path,host_ip,user_name,user_pwd,port=22):

            s = paramiko.SSHClient()
            s.load_system_host_keys()
            s.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
                try:

        s.connect(hostname=host_ip,port=port,username=user_name,password=user_pwd)
                    stdin,stdout,stderr = s.exec_command('stat %s' %file_path)
                    stat_result = '%s%s' %(stdout.read(),stderr.read())
                    if stat_result.find('No such file or directory') == -1:
                        file_status = 'OK\t'
                        stdin,stdout,stderr = s.exec_command('du -sh %s' %file_path)
                        cmd1_result                                                    =
'%s_%s' %(stat_result.split()[32],stat_result.split()[33].split('.')[0])
                        cmd2_result = ('%s%s' %(stdout.read(),stderr.read())).split()[0]
                    else:
                        file_status = '未生成\t'
                        cmd1_result = 'null'
                        cmd2_result = 'null'
                    q.put(['Login successful'])
                    s.close()
                except socket.error:
                    file_status = '主机或端口错误'
                    cmd1_result = '-'
                    cmd2_result = '-'
                except paramiko.AuthenticationException:
                    file_status = '用户或密码错误'
                    cmd1_result = '-'
                    cmd2_result = '-'
                except paramiko.BadHostKeyException:
                    file_status = 'Bad host key'
                    cmd1_result = '-'
                    cmd2_result = '-'
                except:
                    file_status = 'ssh 异常'
                    cmd1_result = '-'
                    cmd2_result = '-'

        r.put('%s\t-\t%s\t%s\t%s\t%s\n'  %(time.strftime('%Y-%m-%d_%H:%M'),host_ip,file_status,c
md2_result,cmd1_result))

            def Concurrent(Conf,file_path,user_name,user_pwd,port):
                # 执行总计
                total = 0
                # 读取配置文件
                f=open(Conf)
                list = f.readlines()
                f.close()
```

```
        # 并发执行
        process_list = []
        log_file = file('file_check.log', 'w')
        log_file.write('检查时间\t\t 业务\tIP\t\t 文件状态\t 大小\t 生成时间\n')
        for host_info in list:
            # 判断配置文件中注释行跳过
            if host_info.startswith('#'):
                continue
            # 取变量,其中任意变量未取到就跳过执行
            try:
                host_ip=host_info.split()[0].strip()
                #user_name=host_info.split()[1]
                #user_pwd=host_info.split()[2]
            except:
                log_file.write('Profile error: %s\n' %(host_info))
                continue
            #try:
            #    port=int(host_info.split()[3])
            #except:
            #    port=22
            total +=1
            p = multiprocessing.Process(target=Ssh_Cmd,args=(file_path,host_ip,user_name,user_pwd,port))
            p.start()
            process_list.append(p)
        for j in process_list:
            j.join()
        for j in process_list:
            log_file.write(r.get())

        successful = q.qsize()
        log_file.write('执行完毕。 总执行:%s 登录成功:%s 登录失败:%s\n' %(total,successful,total - successful))
        log_file.flush()
        log_file.close()

    def send_mail(to_list, sub):
        me = mail_user + "<"+mail_user+"@"+mail_postfix+">"
        fp = open('file_check.log')
        msg = MIMEText(fp.read(),_charset="utf-8")
        fp.close()
        msg['Subject'] = sub
        msg['From'] = me
        msg['To'] = ";".join(to_list)
```

```
        try:
            send_smtp = smtplib.SMTP()
            send_smtp.connect(mail_host)
            send_smtp.login(mail_user, mail_pass)
            send_smtp.sendmail(me, to_list, msg.as_string())
            send_smtp.close()
            return True
        except Exception, e:
            print str(e)[1]
            return False

    if __name__ == '__main__':
        q = multiprocessing.Queue()
        r = multiprocessing.Queue()
        Concurrent(Conf,file_path,user_name,user_pwd,port)
        if send_mail(mailto_list,title):
            print "发送成功"
        else:
            print "发送失败"
```

LazyManage 并发批量操作(判断非 root 交互到 root 操作)

```
    #!/usr/bin/python
    #encoding:utf8
    # LzayManage.py
    # config file: serverlist.conf

    import paramiko
    import multiprocessing
    import sys,os,time,socket,re

    def Ssh_Cmd(host_ip,Cmd,user_name,user_pwd,port=22):
        s = paramiko.SSHClient()
        s.load_system_host_keys()
        s.set_missing_host_key_policy(paramiko.AutoAddPolicy())

s.connect(hostname=host_ip,port=port,username=user_name,password=user_pwd)
        stdin,stdout,stderr = s.exec_command(Cmd)
        Result = '%s%s' %(stdout.read(),stderr.read())
        q.put('successful')
        s.close()
        return Result.strip()

    def
```

```
Ssh_Su_Cmd(host_ip,Cmd,user_name,user_pwd,root_name,root_pwd,port=22):
                s = paramiko.SSHClient()
                s.load_system_host_keys()
                s.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    s.connect(hostname=host_ip,port=port,username=user_name,password=user_pwd)
                ssh = s.invoke_shell()
                time.sleep(0.1)
                ssh.send('su - %s\n' %(root_name))
                buff = ''
                while not buff.endswith('Password: '):
                        resp = ssh.recv(9999)
                        buff +=resp
                ssh.send('%s\n' %(root_pwd))
                buff = ''
                while True:
                        resp = ssh.recv(9999)
                        buff +=resp
                        if ': incorrect password' in buff:
                                su_correct='passwd_error'
                                break
                        elif buff.endswith('# '):
                                su_correct='passwd_correct'
                                break
                if su_correct == 'passwd_correct':
                        ssh.send('%s\n' %(Cmd))
                        buff = ''
                        while True:
                                resp = ssh.recv(9999)
                                if resp.endswith('# '):
                                        buff +=re.sub('\[.*@.*\]# $','',resp)
                                        break
                                buff +=resp
                        Result = buff.lstrip('%s' %(Cmd))
                        q.put('successful')
                elif su_correct == 'passwd_error':
                        Result = "\033[31mroot 密码错误\033[m"
                s.close()
                return Result.strip()

        def Send_File(host_ip,PathList,user_name,user_pwd,Remote='/tmp',port=22):
                s=paramiko.Transport((host_ip,port))
                s.connect(username=user_name,password=user_pwd)
                sftp=paramiko.SFTPClient.from_transport(s)
```

```
                    for InputPath in PathList:
                        LocalPath = re.sub('^\./','',InputPath.rstrip('/'))
                        RemotePath = '%s/%s' %( Remote , os.path.basename( LocalPath ))
                        try:
                            sftp.rmdir(RemotePath)
                        except:
                            pass
                        try:
                            sftp.remove(RemotePath)
                        except:
                            pass
                        if os.path.isdir(LocalPath):
                            sftp.mkdir(RemotePath)
                            for path,dirs,files in os.walk(LocalPath):
                                for dir in dirs:
                                    dir_path = os.path.join(path,dir)

    sftp.mkdir('%s/%s' %(RemotePath,re.sub('^%s/' %LocalPath,'',dir_path)))
                                for file in files:
                                    file_path = os.path.join(path,file)

    sftp.put( file_path,'%s/%s' %(RemotePath,re.sub('^%s/' %LocalPath,'',file_path)))
                        else:
                            sftp.put(LocalPath,RemotePath)
                    q.put('successful')
                    sftp.close()
                    s.close()
                    Result = '%s    \033[32m 传送完成\033[m' % PathList
                    return Result

            def
Ssh(host_ip,Operation,user_name,user_pwd,root_name,root_pwd,Cmd=None,PathList=None,po
rt=22):
                    msg = "\033[32m----------Result:%s----------\033[m" % host_ip
                    try:
                        if Operation == 'Ssh_Cmd':
                            Result                                                    =
Ssh_Cmd(host_ip=host_ip,Cmd=Cmd,user_name=user_name,user_pwd=user_pwd,port=port)
                        elif Operation == 'Ssh_Su_Cmd':
                            Result                                                    =
Ssh_Su_Cmd(host_ip=host_ip,Cmd=Cmd,user_name=user_name,user_pwd=user_pwd,root_nam
e=root_name,root_pwd=root_pwd,port=port)
                        elif Operation == 'Ssh_Script':
```

```
    Send_File(host_ip=host_ip,PathList=PathList,user_name=user_name,user_pwd=user_pwd,port=port)
                    Script_Head = open(PathList[0]).readline().strip()
                    LocalPath = re.sub('^\./','',PathList[0].rstrip('/'))
                    Cmd = '%s /tmp/%s' %( re.sub('^#!','',Script_Head), os.path.basename( LocalPath ))
                    Result = Ssh_Cmd(host_ip=host_ip,Cmd=Cmd,user_name=user_name,user_pwd=user_pwd,port=port)
                elif Operation == 'Ssh_Su_Script':

    Send_File(host_ip=host_ip,PathList=PathList,user_name=user_name,user_pwd=user_pwd,port=port)
                    Script_Head = open(PathList[0]).readline().strip()
                    LocalPath = re.sub('^\./','',PathList[0].rstrip('/'))
                    Cmd = '%s /tmp/%s' %( re.sub('^#!','',Script_Head), os.path.basename( LocalPath ))
                    Result = Ssh_Su_Cmd(host_ip=host_ip,Cmd=Cmd,user_name=user_name,user_pwd=user_pwd,root_name=root_name,root_pwd=root_pwd,port=port)
                elif Operation == 'Send_File':
                    Result = Send_File(host_ip=host_ip,PathList=PathList,user_name=user_name,user_pwd=user_pwd,port=port)
                else:
                    Result = '操作不存在'

            except socket.error:
                Result = '\033[31m 主机或端口错误\033[m'
            except paramiko.AuthenticationException:
                Result = '\033[31m 用户名或密码错误\033[m'
            except paramiko.BadHostKeyException:
                Result = '\033[31mBad host key\033[m['
            except IOError:
                Result = '\033[31m 远程主机已存在非空目录或没有写权限\033[m'
            except:
                Result = '\033[31m 未知错误\033[m'
            r.put('%s\n%s\n' %(msg,Result))

        def Concurrent(Conf,Operation,user_name,user_pwd,root_name,root_pwd,Cmd=None,PathList=None,port=22):
            # 读取配置文件
            f=open(Conf)
            list = f.readlines()
```

```
                f.close()
                # 执行总计
                total = 0
                # 并发执行
                for host_info in list:
                    # 判断配置文件中注释行跳过
                    if host_info.startswith('#'):
                        continue
                    # 取变量,其中任意变量未取到就跳过执行
                    try:
                        host_ip=host_info.split()[0]
                        #user_name=host_info.split()[1]
                        #user_pwd=host_info.split()[2]
                    except:
                        print('Profile error: %s' %(host_info) )
                        continue
                    try:
                        port=int(host_info.split()[3])
                    except:
                        port=22
                    total +=1
                    p = multiprocessing.Process(target=Ssh,args=(host_ip,Operation,user_name,user_pwd,root_name,root_pwd,Cmd,PathList,port))
                    p.start()
                # 打印执行结果
                for j in range(total):
                    print(r.get() )
                if Operation == 'Ssh_Script' or Operation == 'Ssh_Su_Script':
                    successful = q.qsize() / 2
                else:
                    successful = q.qsize()
                print('\033[32m 执 行 完 毕 [ 总 执 行 :%s 成 功 :%s 失败:%s]\033[m' %(total,successful,total - successful) )
                q.close()
                r.close()

            def Help():
                print('''  1.执行命令
                2.执行脚本          \033[32m[位置 1 脚本(必须带脚本头),后可带执行脚本所需要的包\文件\文件夹路径,空格分隔]\033[m
                3.发送文件          \033[32m[传送的包\文件\文件夹路径,空格分隔]\033[m
                退出: 0\exit\quit
                帮助: help\h\?
```

```
        注意: 发送文件默认为/tmp 下,如已存在同名文件会被强制覆盖,非空目录则中断操作.执行脚本先将本地脚本及包发送远程主机上,发送规则同发送文件
        ''')

    if __name__=='__main__':
        # 定义 root 账号信息
        root_name = 'root'
        root_pwd = 'peterli'
        user_name='peterli'
        user_pwd='<++(3Ie'
        # 配置文件
        Conf='serverlist.conf'
        if not os.path.isfile(Conf):
            print('\033[33m 配置文件 %s 不存在\033[m' %(Conf) )
            sys.exit()
        Help()
        while True:
            i = raw_input("\033[35m[请选择操作]: \033[m").strip()
            q = multiprocessing.Queue()
            r = multiprocessing.Queue()
            if i == '1':
                if user_name == root_name:
                    Operation = 'Ssh_Cmd'
                else:
                    Operation = 'Ssh_Su_Cmd'
                Cmd = raw_input('CMD: ').strip()
                if len(Cmd) == 0:
                    print('\033[33m 命令为空\033[m')
                    continue

    Concurrent(Conf=Conf,Operation=Operation,user_name=user_name,user_pwd=user_pwd,root_name=root_name,root_pwd=root_pwd,Cmd=Cmd)
            elif i == '2':
                if user_name == root_name:
                    Operation = 'Ssh_Script'
                else:
                    Operation = 'Ssh_Su_Script'
                PathList = raw_input('\033[36m 本地脚本路径: \033[m').strip().split()
                if len(PathList) == 0:
                    print('\033[33m 路径为空\033[m')
                    continue
                if not os.path.isfile(PathList[0]):
                    print('\033[33m 本地路径 %s 不存在或不是文件
```

```
\033[m' %(PathList[0]) )
                                continue
                            for LocalPath in PathList[1:]:
                                if not os.path.exists(LocalPath):
                                    print('\033[33m  本 地 路 径  %s  不 存 在
\033[m' %(LocalPath) )
                                    break
                            else:

    Concurrent(Conf=Conf,Operation=Operation,user_name=user_name,user_pwd=user_pwd,r
oot_name=root_name,root_pwd=root_pwd,PathList=PathList)
                        elif i == '3':
                            Operation = 'Send_File'
                            PathList = raw_input('\033[36m 本地路径: \033[m').strip().split()
                            if len(PathList) == 0:
                                print('\033[33m 路径为空\033[m')
                                continue
                            for LocalPath in PathList:
                                if not os.path.exists(LocalPath):
                                    print('\033[33m  本 地 路 径  %s  不 存 在
\033[m' %(LocalPath) )
                                    break
                            else:

    Concurrent(Conf=Conf,Operation=Operation,user_name=user_name,user_pwd=user_pwd,r
oot_name=root_name,root_pwd=root_pwd,PathList=PathList)
                        elif i == '0' or i == 'exit' or i == 'quit':
                            print("\033[34m 退出 LazyManage 脚本\033[m")
                            sys.exit()
                        elif i == 'help' or i == 'h' or i == '?':
                            Help()

    pysnmp

        #!/usr/bin/python
        from pysnmp.entity.rfc3413.oneliner import cmdgen

        cg = cmdgen.CommandGenerator()

        # 注意 IP 端口 组默认 public  oid 值
        varBinds = cg.getCmd( cmdgen.CommunityData('any-agent', 'public',0 ),
cmdgen.UdpTransportTarget(('10.10.76.42', 161)), (1,3,6,1,4,1,2021,10,1,3,1), )

        print varBinds[3][0][1]
```

4 3 socket

```
socket.gethostname()        # 获取主机名
from socket import *        # 避免 socket.socket()
s=socket()
s.bind()                    # 绑定地址到套接字
s.listen()                  # 开始 TCP 监听
s.accept()                  # 被动接受 TCP 客户端连接，等待连接的到来
s.connect()                 # 主动初始化 TCP 服务器连接
s.connect_ex()              # connect()函数的扩展版本，出错时返回出错码，而不是跑出异常
s.recv()                    # 接收 TCP 数据
s.send()                    # 发送 TCP 数据
s.sendall()                 # 完整发送 TCP 数据
s.recvfrom()                # 接收 UDP 数据
s.sendto()                  # 发送 UDP 数据
s.getpeername()             # 连接到当前套接字的远端的地址(TCP 连接)
s.getsockname()             # 当前套接字的地址
s.getsockopt()              # 返回指定套接字的参数
s.setsockopt()              # 设置指定套接字的参数
s.close()                   # 关闭套接字
s.setblocking()             # 设置套接字的阻塞与非阻塞模式
s.settimeout()              # 设置阻塞套接字操作的超时时间
s.gettimeout()              # 得到阻塞套接字操作的超时时间
s.filen0()                  # 套接字的文件描述符
s.makefile()                # 创建一个与该套接字关联的文件对象

socket.AF_UNIX              # 只能够用于单一的 Unix 系统进程间通信
socket.AF_INET              # 服务器之间网络通信
socket.AF_INET6             # IPv6

socket.SOCK_STREAM          # 流式 socket , for TCP
socket.SOCK_DGRAM           # 数据报式 socket , for UDP
socket.SOCK_RAW             # 原始套接字，普通的套接字无法处理 ICMP、IGMP 等网络报文，而 SOCK_RAW 可以；其次，SOCK_RAW 也可以处理特殊的 IPv4 报文；此外，利用原始套接字，可以通过 IP_HDRINCL 套接字选项由用户构造 IP 头。

socket.SOCK_RDM             # 是一种可靠的 UDP 形式，即保证交付数据报但不保证顺序。SOCK_RAM 用来提供对原始协议的低级访问，在需要执行某些特殊操作时使用，如发送 ICMP 报文。SOCK_RAM 通常仅限于高级用户或管理员运行的程序使用。
```

socket.SOCK_SEQPACKET　　# 可靠的连续数据包服务

4.1 SocketServer

```
#!/usr/bin/python
#server.py
import SocketServer
import os
class MyTCP(SocketServer.BaseRequestHandler):
    def handle(self):
        while True:
            self.data=self.request.recv(1024).strip()
            if self.data == 'quit' or not self.data:break

            cmd=os.popen(self.data).read()
            if cmd == '':cmd= self.data + ': Command not found'
            self.request.sendall(cmd)
if __name__ == '__main__':
    HOST,PORT = '10.0.0.119',50007
    server = SocketServer.ThreadingTCPServer((HOST,PORT),MyTCP)
    server.serve_forever()
```

4.2 SocketClient

```
#!/usr/bin/python
#client.py
import socket

HOST='10.0.0.119'
PORT=50007
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((HOST,PORT))

while True:
    while True:
        cmd=raw_input('CMD:').strip()
        if cmd != '':break
    s.sendall(cmd)
    data=s.recv(1024).split('\n')
```

```
        print 'cmd:'
        for line in data:print line
    s.close()
```

4.3 ftp

4.3.1 ftpserver

```
    #!/usr/bin/python
    #ftpserver.py

    import SocketServer
    import os
    import cPickle
    import md5
    from time import sleep

    def filer(file1):
        try:
            f = file(file1,'rb')
            return cPickle.load(f)
        except IOError:
            return {}
        except EOFError:
            return {}
        f.close()

    def filew(file1,content):
        f = file(file1,'wb')
        cPickle.dump(content,f)
        f.close()

    class MyTCP(SocketServer.BaseRequestHandler):
        def handle(self):
            i = 0
            while i<3:
                user=self.request.recv(1024).strip()
                userinfo=filer('user.pkl')
                if userinfo.has_key(user.split()[0]):
```

```
                                    if                  md5.new(user.split()[1]).hexdigest()                  ==
userinfo[user.split()[0]]:
                                        results='login successful'
                                        self.request.sendall(results)
                                        login='successful'
                                        break
                                    else:
                                        i = i + 1
                                        results='Error:password not correct'
                                        self.request.sendall(results)
                                        continue
                                else:
                                    i = i + 1
                                    results='Error:password not correct'
                                    self.request.sendall(results)
                                    continue
                                break
                            else:
                                results = 'Error:Wrong password too many times'
                                self.request.sendall(results)
                                login='failure'
                            home_path = os.popen('pwd').read().strip() + '/' + user.split()[0]
                            current_path = '/'
                            print home_path
                            while True:
                                if login == 'failure':
                                    break
                                print
'home_path:%s=current_path:%s' %(home_path,current_path)
                                cmd=self.request.recv(1024).strip()
                                print cmd
                                if cmd == 'quit':
                                    break
                                elif cmd == 'dir':
                                    list=os.listdir('%s%s' %(home_path,current_path))
                                    if list:
                                        dirlist,filelist = '',''
                                        for i in list:
                                            if
os.path.isdir('%s%s%s' %(home_path,current_path,i)):
                                                dirlist = dirlist + '\033[32m' + i + '\033[m\t'
                                            else:
                                                filelist = filelist + i + '\t'
                                        results = dirlist + filelist
```

```
                else:
                    results = '\033[31mnot find\033[m'
                self.request.sendall(results)
            elif cmd == 'pdir':
                self.request.sendall(current_path)
            elif cmd.split()[0] == 'mdir':
                if cmd.split()[1].isalnum():

    tmppath='%s%s%s' %(home_path,current_path,cmd.split()[1])
                    os.makedirs(tmppath)
                    self.request.sendall('\033[32mcreating successful\033[m')
                else:
                    self.request.sendall('\033[31mcreate failure\033[m')
            elif cmd.split()[0] == 'cdir':
                if cmd.split()[1] == '/':
                    tmppath='%s%s' %(home_path,cmd.split()[1])
                    if os.path.isdir(tmppath):
                        current_path = cmd.split()[1]
                        self.request.sendall(current_path)
                    else:
                        self.request.sendall('\033[31mnot_directory\033[m')
                elif cmd.split()[1].startswith('/'):
                    tmppath='%s%s' %(home_path,cmd.split()[1])
                    if os.path.isdir(tmppath):
                        current_path = cmd.split()[1] + '/'
                        self.request.sendall(current_path)
                    else:
                        self.request.sendall('\033[31mnot_directory\033[m')
                else:

    tmppath='%s%s%s' %(home_path,current_path,cmd.split()[1])
                    if os.path.isdir(tmppath):
                        current_path = current_path + cmd.split()[1] + '/'
                        self.request.sendall(current_path)
                    else:
                        self.request.sendall('\033[31mnot_directory\033[m')
            elif cmd.split()[0] == 'get':
                if
os.path.isfile('%s%s%s' %(home_path,current_path,cmd.split()[1])):
                    f                                                    =
file('%s%s%s' %(home_path,current_path,cmd.split()[1]),'rb')
                    self.request.sendall('ready_file')
                    sleep(0.5)
                    self.request.send(f.read())
```

```
                                        f.close()
                                        sleep(0.5)
                                    elif
os.path.isdir('%s%s%s' %(home_path,current_path,cmd.split()[1])):
                                        self.request.sendall('ready_dir')
                                        sleep(0.5)
                                        for dirpath in
os.walk('%s%s%s' %(home_path,current_path,cmd.split()[1])):

    dir=dirpath[0].replace('%s%s' %(home_path,current_path),'',1)
                                            self.request.sendall(dir)
                                            sleep(0.5)
                                            for filename in dirpath[2]:
                                                self.request.sendall(filename)
                                                sleep(0.5)
                                                f = file('%s/%s' %(dirpath[0],filename),'rb')
                                                self.request.send(f.read())
                                                f.close()
                                                sleep(0.5)
                                                self.request.sendall('file_get_done')
                                                sleep(0.5)
                                            else:
                                                self.request.sendall('dir_get_done')
                                            sleep(0.5)
                                    else:
                                        self.request.sendall('get_failure')
                                        continue
                                    self.request.sendall('get_done')

                                elif cmd.split()[0] == 'send':
                                    if
os.path.exists('%s%s%s' %(home_path,current_path,cmd.split()[1])):
                                        self.request.sendall('existing')
                                        action=self.request.recv(1024)
                                        if action == 'cancel':
                                            continue
                                    self.request.sendall('ready')
                                    msg=self.request.recv(1024)
                                    if msg == 'ready_file':
                                        f =
file('%s%s%s' %(home_path,current_path,cmd.split()[1]),'wb')
                                        while True:
                                            data=self.request.recv(1024)
                                            if data == 'file_send_done':break
```

```
                                        f.write(data)
                                    f.close()

                                elif msg == 'ready_dir':
                                    os.system('mkdir
-p %s%s%s' %(home_path,current_path,cmd.split()[1]))
                                    while True:
                                        dir=self.request.recv(1024)
                                        if dir == 'get_done':break
                                        os.system('mkdir
-p %s%s%s' %(home_path,current_path,dir))
                                        while True:
                                            filename=self.request.recv(1024)
                                            if filename == 'dir_send_done':break
                                            f                                                    =
file('%s%s%s/%s' %(home_path,current_path,dir,filename),'wb')
                                            while True:
                                                data=self.request.recv(1024)
                                                if data == 'file_send_done':break
                                                f.write(data)
                                            f.close()

        self.request.sendall('%s/%s\t\033[32mfile_done\033[m' %(dir,filename))

        self.request.sendall('%s\t\033[32mdir_done\033[m' %(dir))
                                elif msg == 'unknown_file':
                                    continue

                            else:
                                results = cmd.split()[0] + ': Command not found'
                                self.request.sendall(results)

    if __name__ == '__main__':
        HOST,PORT = '10.152.14.85',50007
        server = SocketServer.ThreadingTCPServer((HOST,PORT),MyTCP)
        server.serve_forever()
```

ftpmanage

```
    #!/usr/bin/python
    #manage_ftp.py
    import cPickle
    import sys
    import md5
```

```
import os
import getpass

def filer(file1):
    try:
        f = file(file1,'rb')
        return cPickle.load(f)
    except IOError:
        return {}
    except EOFError:
        return {}
    f.close()

def filew(file1,content):
    f = file(file1,'wb')
    cPickle.dump(content,f)
    f.close()

while True:
    print '''
    1.add user
    2.del user
    3.change password
    4.query user
    0.exit
    '''
    i = raw_input(':').strip()
    userinfo=filer('user.pkl')
    if i == '':
        continue
    elif i == '1':
        while True:
            user=raw_input('user name:').strip()
            if user.isalnum():
                i = 0
                while i<3:
                    passwd=getpass.getpass('passwd:').strip()
                    if passwd == '':
                        continue
                    else:
                        passwd1=getpass.getpass('Confirm
password:').strip()
                        if passwd == passwd1:
                            mpasswd = md5.new(passwd).hexdigest()
```

```
                                userinfo[user] = mpasswd
                                os.system('mkdir -p %s' %user)
                                print '%s creating successful ' %user
                                break
                            else:
                                print "Passwords don't match "
                                i = i + 1
                                continue
                    else:
                        print 'Too many wrong'
                        continue
                    break
                else:
                    print 'user not legal'
                    continue
    elif i == '2':
        user=raw_input('user name:').strip()
        if userinfo.has_key(user):
            del userinfo[user]
            print 'Delete users successfully'
        else:
            print 'user not exist'
            continue
    elif i == '3':
        user=raw_input('user name:').strip()
        if userinfo.has_key(user):
            i = 0
            while i<3:
                passwd=getpass.getpass('passwd:').strip()
                if passwd == '':
                    continue
                else:
                    passwd1=getpass.getpass('Confirm password:').strip()
                    if passwd == passwd1:
                        mpasswd = md5.new(passwd).hexdigest()
                        userinfo[user] = mpasswd
                        print '%s password is changed' %user
                        break
                    else:
                        print "Passwords don't match "
                        i = i + 1
                        continue
            else:
                print 'Too many wrong'
```

```
                continue
        else:
                print 'user not exist'
                continue
    elif i == '4':
        print userinfo.keys()
    elif i == '0':
        sys.exit()
    else:
        print 'select error'
        continue
    filew('user.pkl',content=userinfo)
```

4.3.2 ftpclient

```
#!/usr/bin/python
#ftpclient.py

import socket
import os
import getpass
from time import sleep

HOST='10.152.14.85'
PORT=50007
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect((HOST,PORT))

while True:
    user = raw_input('user:').strip()
    if user.isalnum():
        while True:
            passwd = getpass.getpass('passwd:').strip()
            s.sendall(user + ' ' + passwd)
            servercmd=s.recv(1024)
            if servercmd == 'login successful':
                print '\033[32m%s\033[m' %servercmd
                break
            else:
                print servercmd

        while True:
```

```
                cmd=raw_input('FTP>').strip()
                if cmd == '':
                    continue
                if cmd.split()[0] == 'get':
                    if cmd == 'get':continue
                    for i in cmd.split()[1:]:
                        if os.path.exists(i):
                            confirm = raw_input("\033[31mPlease confirm whether the cover %s(Y/N):\033[m" %(i)).upper().startswith('Y')
                            if not confirm:
                                print '%s cancel' %i
                                continue
                        s.sendall('get ' + i)
                        servercmd=s.recv(1024)
                        if servercmd == 'inexistence':
                            print '%s \t\033[32minexistence\033[m' %i
                            continue
                        elif servercmd == 'ready_file':
                            f = file(i,'wb')
                            while True:
                                data=s.recv(1024)
                                if data == 'get_done':break
                                f.write(data)
                            f.close()
                            print '%s \t\033[32mfile_done\033[m' %(i)
                        elif servercmd == 'ready_dir':
                            try:
                                os.makedirs(i)
                            except:
                                pass
                            while True:
                                serverdir=s.recv(1024)
                                if serverdir == 'get_done':break
                                os.system('mkdir -p %s' %serverdir)
                                print '%s \t\033[32mdir_done\033[m' %(serverdir)
                                while True:
                                    serverfile=s.recv(1024)
                                    if serverfile == 'dir_get_done':break
                                    f = file('%s/%s' %(serverdir,serverfile),'wb')
                                    while True:
                                        data=s.recv(1024)
                                        if data == 'file_get_done':break
                                        f.write(data)
```

```
                                                f.close()
                                                print                                   '%s/%s
\t\033[32mfile_done\033[m' %(serverdir,serverfile)

                                elif cmd.split()[0] == 'send':

                                    if cmd == 'send':continue
                                    for i in cmd.split()[1:]:
                                        if not os.path.exists(i):
                                            print '%s\t\033[31minexistence\033[m' %i
                                            continue

                                        s.sendall('send ' + i)
                                        servercmd=s.recv(1024)
                                        if servercmd == 'existing':
                                            confirm = raw_input("\033[31mPlease confirm
whether the cover %s(Y/N):\033[m" %(i)).upper().startswith('Y')
                                            if confirm:
                                                s.sendall('cover')
                                                servercmd=s.recv(1024)
                                            else:
                                                s.sendall('cancel')
                                                print '%s\tcancel' %i
                                                continue

                                        if os.path.isfile(i):
                                            s.sendall('ready_file')
                                            sleep(0.5)
                                            f = file(i,'rb')
                                            s.send(f.read())
                                            sleep(0.5)
                                            s.sendall('file_send_done')
                                            print                                   '%s\t\033[32mfile
done\033[m' %(cmd.split()[1])
                                            f.close()
                                        elif os.path.isdir(i):
                                            s.sendall('ready_dir')
                                            sleep(0.5)
                                            for dirpath in os.walk(i):

    dir=dirpath[0].replace('%s/' %os.popen('pwd').read().strip(),'',1)
                                                s.sendall(dir)
                                                sleep(0.5)
                                                for filename in dirpath[2]:
```

```
                                s.sendall(filename)
                                sleep(0.5)
                                f = file('%s/%s' %(dirpath[0],filename),'rb')
                                s.send(f.read())
                                f.close()
                                sleep(0.5)
                                s.sendall('file_send_done')
                                msg=s.recv(1024)
                                print msg

                            else:
                                s.sendall('dir_send_done')
                                msg=s.recv(1024)
                                print msg

                    else:
                        s.sendall('unknown_file')
                        print '%s\t\033[31munknown type\033[m' %i
                        continue
                    sleep(0.5)
                    s.sendall('get_done')

            elif cmd.split()[0] == 'cdir':
                if cmd == 'cdir':continue
                s.sendall(cmd)
                data=s.recv(1024)
                print data
                continue
            elif cmd == 'ls':
                list=os.popen(cmd).read().strip().split('\n')
                if list:
                    dirlist,filelist = '',''
                    for i in list:
                        if os.path.isdir(i):
                            dirlist = dirlist + '\033[32m' + i + '\033[m\t'
                        else:
                            filelist = filelist + i + '\t'
                    results = dirlist + filelist
                else:
                    results = '\033[31mnot find\033[m'
                print results
                continue
            elif cmd == 'pwd':
                os.system(cmd)
```

```
            elif cmd.split()[0] == 'cd':
                try:
                    os.chdir(cmd.split()[1])
                except:
                    print '\033[31mcd failure\033[m'
            elif cmd == 'dir':
                s.sendall(cmd)
                data=s.recv(1024)
                print data
                continue
            elif cmd == 'pdir':
                s.sendall(cmd)
                data=s.recv(1024)
                print data
                continue
            elif cmd.split()[0] == 'mdir':
                if cmd == 'mdir':continue
                s.sendall(cmd)
                data=s.recv(1024)
                print data
                continue
            elif cmd.split()[0] == 'help':
                print '''
get [file] [dir]
send [file] [dir]

dir
mdir
cdir
pdir

pwd
md
cd
ls

help
quit
'''
                continue
            elif cmd == 'quit':
                break
            else:
                print '\033[31m%s: Command not found,Please see the
```

```
"help"\033[m' %cmd
                    else:
                        continue
                    break
                s.close()
```

4.4 扫描主机开放端口

```
#!/usr/bin/env python

import socket

def check_server(address,port):
    s=socket.socket()
    try:
        s.connect((address,port))
        return True
    except socket.error,e:
        return False

if __name__=='__main__':
    from optparse import OptionParser
    parser=OptionParser()

parser.add_option("-a","--address",dest="address",default='localhost',help="Address for server",metavar="ADDRESS")
    parser.add_option("-s","--start",dest="start_port",type="int",default=1,help="start port",metavar="SPORT")
    parser.add_option("-e","--end",dest="end_port",type="int",default=1,help="end port",metavar="EPORT")
    (options,args)=parser.parse_args()
    print 'options: %s, args: %s' % (options, args)
    port=options.start_port
    while(port<=options.end_port):
        check = check_server(options.address, port)
        if (check):
            print 'Port    %s is on' % port
        port=port+1
```

5 4 mysql

```
#apt-get install mysql-server
#apt-get install python-MySQLdb
help(MySQLdb.connections.Connection)          # 查看链接参数

conn=MySQLdb.connect(host='localhost',user='root',passwd='123456',db='fortress',port=3306)     # 定义连接
#conn=MySQLdb.connect(unix_socket='/var/run/mysqld/mysqld.sock',user='root',passwd='123456')     # 使用 socket 文件链接
cur=conn.cursor()                                                    # 定义游标
conn.select_db('fortress')                                           # 选择数据库
sqlcmd = 'insert into user(name,age) value(%s,%s)'                   # 定义 sql 命令
cur.executemany(sqlcmd,[('aa',1),('bb',2),('cc',3)])                 # 插入多条值
cur.execute('delete from user where id=20')                          # 删除一条记录
cur.execute("update user set name='a' where id=20")                  # 更细数据
sqlresult = cur.fetchall()                                           # 接收全部返回结果
conn.commit()                                                        # 提交
cur.close()                                                          # 关闭游标
conn.close()                                                         # 关闭连接

import MySQLdb
def mydb(dbcmdlist):
    try:

conn=MySQLdb.connect(host='localhost',user='root',passwd='123456',db='fortress',port=3306)
        cur=conn.cursor()

        cur.execute('create database if not exists fortress;')    # 创建数据库
        conn.select_db('fortress')                                # 选择数据库
        cur.execute('drop table if exists log;')                  # 删除表
        cur.execute('CREATE TABLE log ( id BIGINT(20) NOT NULL AUTO_INCREMENT, loginuser VARCHAR(50) DEFAULT NULL, remoteip VARCHAR(50) DEFAULT NULL, PRIMARY KEY (id) );')    # 创建表

        result=[]
        for dbcmd in dbcmdlist:
            cur.execute(dbcmd)                # 执行 sql
            sqlresult = cur.fetchall()     # 接收全部返回结果
            result.append(sqlresult)
        conn.commit()                         # 提交
```

```
            cur.close()
            conn.close()
            return result
        except MySQLdb.Error,e:
            print 'mysql error msg: ',e
    sqlcmd=[]
    sqlcmd.append("insert into log (loginuser,remoteip)values('%s','%s');" %(loginuser,remoteip))
    mydb(sqlcmd)

    sqlcmd=[]
    sqlcmd.append("select * from log;")
    result = mydb(sqlcmd)
    for i in result[0]:
        print i
```

6 5 处理信号

6.1 信号的概念

信号(signal): 进程之间通讯的方式，是一种软件中断。一个进程一旦接收到信号就会打断原来的程序执行流程来处理信号。

发送信号一般有两种原因:

1(被动式) 内核检测到一个系统事件.例如子进程退出会像父进程发送 SIGCHLD 信号.键盘按下 control+c 会发送 SIGINT 信号

2(主动式) 通过系统调用 kill 来向指定进程发送信号

操作系统规定了进程收到信号以后的默认行为,可以通过绑定信号处理函数来修改进程收到信号以后的行为，有两个信号是不可更改的 SIGTOP 和 SIGKILL

如果一个进程收到一个 SIGUSR1 信号，然后执行信号绑定函数，第二个 SIGUSR2 信号又来了，第一个信号没有被处理完毕的话，第二个信号就会丢弃。

进程结束信号 SIGTERM 和 SIGKILL 的区别: SIGTERM 比较友好，进程能捕捉这个信号，根据您的需要来关闭程序。在关闭程序之前，您可以结束打开的记录文件和完成正在做的任务。在某些情况下，假如进程正在进行作业而且不能中断，那么进程可以忽略这个 SIGTERM 信号。

6.2 常见信号

kill -l # 查看 linux 提供的信号

SIGHUP 1 A # 终端挂起或者控制进程终止
SIGINT 2 A # 键盘终端进程(如 control+c)
SIGQUIT 3 C # 键盘的退出键被按下
SIGILL 4 C # 非法指令
SIGABRT 6 C # 由 abort(3)发出的退出指令
SIGFPE 8 C # 浮点异常
SIGKILL 9 AEF # Kill 信号 立刻停止
SIGSEGV 11 C # 无效的内存引用
SIGPIPE 13 A # 管道破裂: 写一个没有读端口的管道
SIGALRM 14 A # 闹钟信号 由 alarm(2)发出的信号
SIGTERM 15 A # 终止信号,可让程序安全退出 kill -15
SIGUSR1 30,10,16 A # 用户自定义信号 1
SIGUSR2 31,12,17 A # 用户自定义信号 2
SIGCHLD 20,17,18 B # 子进程结束自动向父进程发送 SIGCHLD 信号
SIGCONT 19,18,25 # 进程继续（曾被停止的进程）
SIGSTOP 17,19,23 DEF # 终止进程
SIGTSTP 18,20,24 D # 控制终端（tty）上按下停止键
SIGTTIN 21,21,26 D # 后台进程企图从控制终端读
SIGTTOU 22,22,27 D # 后台进程企图从控制终端写

缺省处理动作一项中的字母含义如下:

A 缺省的动作是终止进程

B 缺省的动作是忽略此信号，将该信号丢弃，不做处理

C 缺省的动作是终止进程并进行内核映像转储(dump core),内核映像转储是指将进程数据在内存的映像和进程在内核结构中的部分内容以一定格式转储到文件系统，并且进程退出执行，这样做的好处是为程序员提供了方便，使得他们可以得到进程当时执行时的数据值，允许他们确定转储的原因，并且可以调试他们的程序。

D 缺省的动作是停止进程，进入停止状况以后还能重新进行下去，一般是在调试的过程中（例如 ptrace 系统调用）

E 信号不能被捕获

F 信号不能被忽略

6.3 Python 提供的信号

```
import signal
dir(signal)
```

['NSIG', 'SIGABRT', 'SIGALRM', 'SIGBUS', 'SIGCHLD', 'SIGCLD', 'SIGCONT', 'SIGFPE', 'SIGHUP', 'SIGILL', 'SIGINT', 'SIGIO', 'SIGIOT', 'SIGKILL', 'SIGPIPE', 'SIGPOLL', 'SIGPROF', 'SIGPWR',

'SIGQUIT', 'SIGRTMAX', 'SIGRTMIN', 'SIGSEGV', 'SIGSTOP', 'SIGSYS', 'SIGTERM', 'SIGTRAP', 'SIGTSTP', 'SIGTTIN', 'SIGTTOU', 'SIGURG', 'SIGUSR1', 'SIGUSR2', 'SIGVTALRM', 'SIGWINCH', 'SIGXCPU', 'SIGXFSZ', 'SIG_DFL', 'SIG_IGN', '__doc__', '__name__', 'alarm', 'default_int_handler', 'getsignal', 'pause', 'signal']

6.4 绑定信号处理函数

```
#encoding:utf8
import os,signal
from time import sleep
def onsignal_term(a,b):
    print 'SIGTERM'          # kill -15
signal.signal(signal.SIGTERM,onsignal_term)        # 接收信号,执行相应函数

def onsignal_usr1(a,b):
    print 'SIGUSR1'          # kill -10
signal.signal(signal.SIGUSR1,onsignal_usr1)

while 1:
    print 'ID',os.getpid()
    sleep(10)
```

6.5 通过另外一个进程发送信号

```
import os,signal
os.kill(16175,signal.SIGTERM)      # 发送信号，16175 是绑定信号处理函数的进程 pid，需要自行修改
os.kill(16175,signal.SIGUSR1)
```

6.6 父进程接收子进程结束发送的 SIGCHLD 信号

```
#encoding:utf8
import os,signal
from time import sleep

def onsigchld(a,b):
    print '收到子进程结束信号'
signal.signal(signal.SIGCHLD,onsigchld)
```

```
pid = os.fork()                    # 创建一个子进程,复制父进程所有资源操作
if pid == 0:                       # 通过判断子进程 os.fork()是否等于 0,分别同时执
行父进程与子进程操作
    print '我是子进程,pid 是',os.getpid()
    sleep(2)
else:
    print '我是父进程,pid 是',os.getpid()
    os.wait()          # 等待子进程结束
```

6.7 接收信号的程序，另外一端使用多线程向这个进程发送信号，会遗漏一些信号

```
#encoding:utf8
import os
import signal
from time import sleep
import Queue
QCOUNT = Queue.Queue()   # 初始化队列
def onsigchld(a,b):
    '''收到信号后向队列中插入一个数字 1'''
    print '收到 SIGUSR1 信号'
    sleep(1)
    QCOUNT.put(1)          # 向队列中写入
signal.signal(signal.SIGUSR1,onsigchld)    # 绑定信号处理函数
while 1:
    print '我的 pid 是',os.getpid()
    print '现在队列中元素的个数是',QCOUNT.qsize()
    sleep(2)
```

6.8 多线程发信号端的程序

```
#encoding:utf8
import threading
import os
import signal
def sendusr1():
print '发送信号'
    os.kill(17788, signal.SIGUSR1)       # 这里的进程 id 需要写前一个程序实际
运行的 pid
```

```
WORKER = []
for i in range(1, 7):                          # 开启 6 个线程
    threadinstance = threading.Thread(target = sendusr1)
    WORKER.append(threadinstance)
for i in WORKER:
    i.start()
for i in WORKER:
    i.join()
print '主线程完成'
```

7 6 缓存数据库

7.1 python 使用 memcache

```
easy_install python-memcached     # 安装(python2.7+)
import memcache
mc = memcache.Client(['10.152.14.85:12000'],debug=True)
mc.set('name','luo',60)
mc.get('name')
mc.delete('name1')
```

保存数据

```
set(key,value,timeout)          # 把 key 映射到 value，timeout 指的是什么时候这个映射失效
add(key,value,timeout)          # 仅当存储空间中不存在键相同的数据时才保存
replace(key,value,timeout)    # 仅当存储空间中存在键相同的数据时才保存
```

获取数据

```
get(key)                              # 返回 key 所指向的 value
get_multi(key1,key2,key3)     # 可以非同步地同时取得多个键值， 比循环调用 get 快数十倍
```

7.2 python 使用 mongodb

原文: http://blog.nosqlfan.com/html/2989.html

```
easy_install pymongo          # 安装(python2.7+)
import pymongo
connection=pymongo.Connection('localhost',27017)    # 创建连接
db = connection.test_database                        # 切换数据库
collection = db.test_collection                      # 获取 collection
# db 和 collection 都是延时创建的，在添加 Document 时才真正创建
```

文档添加, _id 自动创建

```
    import datetime
    post = {"author": "Mike",
        "text": "My first blog post!",
        "tags": ["mongodb", "python", "pymongo"],
        "date": datetime.datetime.utcnow()}
    posts = db.posts
    posts.insert(post)
    ObjectId('...')
```

批量插入

```
    new_posts = [{"author": "Mike",
        "text": "Another post!",
        "tags": ["bulk", "insert"],
        "date": datetime.datetime(2009, 11, 12, 11, 14)},
        {"author": "Eliot",
        "title": "MongoDB is fun",
        "text": "and pretty easy too!",
        "date": datetime.datetime(2009, 11, 10, 10, 45)}]
    posts.insert(new_posts)
    [ObjectId('...'), ObjectId('...')]
```

获取所有 collection

```
    db.collection_names()      # 相当于 SQL 的 show tables
```

获取单个文档

```
    posts.find_one()
```

查询多个文档

```
    for post in posts.find():
        post
```

加条件的查询

```
posts.find_one({"author": "Mike"})
```

高级查询

```
posts.find({"date": {"$lt": "d"}}).sort("author")
```

统计数量

```
posts.count()
```

加索引

```
from pymongo import ASCENDING, DESCENDING
posts.create_index([("date", DESCENDING), ("author", ASCENDING)])
```

查看查询语句的性能

```
posts.find({"date": {"$lt": "d"}}).sort("author").explain()["cursor"]
posts.find({"date": {"$lt": "d"}}).sort("author").explain()["nscanned"]
```

7.3 python 使用 redis

```
https://pypi.python.org/pypi/redis
pip install redis    OR easy_install redis
import redis
r = redis.StrictRedis(host='localhost', port=6379, db=0)
r.set('foo', 'bar')
r.get('foo')
r.save()
```

分片 # 没搞懂

redis.connection.Connection(host='localhost', port=6379, db=0, parser_class=<class 'redis.connection.PythonParser'>)

redis.ConnectionPool(connection_class=<class 'redis.connection.Connection'>, max_connections=None, **connection_kwargs)

7.4 python 使用 kestrel 队列

```
# pykestrel
import kestrel
```

```
q = kestrel.Client(servers=['127.0.0.1:22133'],queue='test_queue')
q.add('some test job')
job = q.get()       # 从队列读取工作
job = q.peek()     # 读取下一份工作
# 读取一组工作
while True:
    job = q.next(timeout=10) # 完成工作并获取下一个工作，如果没有工作，则等待 10 秒
    if job is not None:
        try:
            # 流程工作
        except:
            q.abort() # 标记失败工作

q.finish()    # 完成最后工作
q.close()     # 关闭连接
```

kestrel 状态检查

```
    # kestrel 支持 memcache 协议客户端
    #!/usr/local/bin/python
    # 10.13.81.125 22133    10000

    import memcache
    import sys
    import traceback

    ip="%s:%s" % (sys.argv[1],sys.argv[2])
    try:
        mc = memcache.Client([ip,])
        st=mc.get_stats()
    except:
        print "kestrel connection exception"
        sys.exit(2)

    if st:
        for s in st[0][1].keys():
            if s.startswith('queue_') and s.endswith('_mem_items'):
                num = int(st[0][1][s])
                if num > int(sys.argv[3]):
                    print "%s block to %s" %(s[6:-6],num)
                    sys.exit(2)
        print "kestrel ok!"
        sys.exit(0)
    else:
```

```
print "kestrel down"
sys.exit(2)
```

7.5 python 使用 tarantool

```
# pip install tarantool-queue

from tarantool_queue import Queue
queue = Queue("localhost", 33013, 0)        #  连接读写端口  空间 0
tube = queue.tube("name_of_tube")             #
tube.put([1, 2, 3])

task = tube.take()
task.data          # take task and read data from it
task.ack()        # move this task into state DONE
```

8 7 web 页面操作

8.1 urllib2 [网络资源访问]

```
import urllib2
response = urllib2.urlopen('http://baidu.com')
print response.geturl()             # url
headers = response.info()
print headers                           # web 页面头部信息
print headers['date']                #  头部信息中的时间
date = response.read()               #  返回页面所有信息[字符串]
# date = response.readlines() #  返回页面所有信息[列表]

for i in urllib2.urlopen('http://qq.com'):       #  可直接迭代
        print i,
```

8.1.1 下载文件

```
#!/usr/bin/env python
#encoding:utf8
import urllib2

url = 'http://www.01happy.com/wp-content/uploads/2012/09/bg.png'
file("./pic/%04d.png" % i, "wb").write(urllib2.urlopen(url).read())
```

8.1.2 抓取网页解析指定内容

```
#!/usr/bin/env python
#encoding:utf8

import urllib2
import urllib
import random
from bs4 import BeautifulSoup

url='http://www.aaammm.com/aaa/'

ua=["Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)",
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)",
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.2; .NET4.0C; .NET4.0E)",
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36",
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36",
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36",
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36",
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36",
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36",
"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safari/537.36",
```

```
        "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/37.0.2062.120 Safari/537.36",
        "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0",
        "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/36.0.1985.143 Safari/537.36"]

        browser = random.choice(ua)

        req_header = {'User-Agent':browser,
        'Accept':'text/html;q=0.9,*/*;q=0.8',
        'Cookie':'BAIDUID=4C8274B52CFB79DEB4FBA9A7EC76A1BC:FG=1;
BDUSS=1dCdU1WNFdxUll0R09XcnBZTkRrVVVNbWVnSkRKSVRPeVljOUswclBoLUNzVEpVQVFBQU
FBJCQAAAAAAAAAAAEAAADEuZ8BcXVhbnpob3U3MjIAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIIIkC1SCJAtUY;        BD_UPN=123143;
BD_HOME=1',      # 添真实登陆后的 Cookie
        'Accept-Charset':'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
        'Connection':'close',
        }
        #data = urllib.urlencode({'name':'xuesong','id':'30' })          # urllib 的处理参
数的方法，可以再 urllib2 中使用
        data = urllib2.quote("pgv_ref=im.perinfo.perinfo.icon&rrr=pppp")
        req_timeout = 10
        try:
            req = urllib2.Request(url,data=data,headers=req_header)          # data 为
None 则方法为 get，有 date 为 post 方法
            html = urllib2.urlopen(req,data=None,req_timeout).read()
        except urllib2.HTTPError as err:
            print str(err)
        except:
            print "timeout"
        print(html)

        # 百度带 Cookie 后查看自己的用户
        #for i in html.split('\n'):
        #    if 'bds.comm.user=' in i:
        #        print i

        soup = BeautifulSoup(html)
        for i in  soup.find_all(target="_blank",attrs={"class": "usr-pic"}):    # 条件看情
况选择
            if i.img:
                print(i.get('href'))
```

8.1.3 模拟浏览器访问 web 页面 python3

```
#! /usr/bin/env python
# -*- coding=utf-8 -*-
import urllib.request

url = "http://www.baidu.com"
# AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1)',
'Accept':'text/html;q=0.9,*/*;q=0.8',
'Accept-Charset':'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
'Connection':'close',
'Referer':None #注意如果依然不能抓取的话，这里可以设置抓取网站的 host
}

opener = urllib.request.build_opener()
opener.addheaders = [headers]
data = opener.open(url).read()

print(data)
```

8.2 requests [替代 urllib2]

```
# Requests 是一个 Python 的 HTTP 客户端库
# 官方中文文档 http://cn.python-requests.org/zh_CN/latest/user/quickstart.html#id2
#  安装: sudo pip install requests
import requests

# get 方法提交表单
url = r'http://dict.youdao.com/search?le=eng&q={0}'.format(word.strip())
r = requests.get(url,timeout=2)

# get 方法带参数  http://httpbin.org/get?key=val
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.get("http://httpbin.org/get", params=payload)

# post 方法提交表单
QueryAdd='http://www.anti-spam.org.cn/Rbl/Query/Result'
r = requests.post(url=QueryAdd, data={'IP':'211.211.54.54'})
```

```
# 定制请求头 post 请求
payload = {'some': 'data'}
headers = {'content-type': 'application/json'}
r = requests.post(url, data=json.dumps(payload), headers=headers)

# https 需登录加 auth
r = requests.get('https://baidu.com', auth=('user', 'pass'))

if r.ok:      # 判断请求是否正常
    print r.url                  # u'http://httpbin.org/get?key2=value2&key1=value1'
    print r.status_code          # 状态码
    print r.content              # 获取到的原始内容  可使用 BeautifulSoup4 解析处理判定结果
    print r.text                 # 把原始内容转 unicode 编码
    print r.headers              # 响应头
    print r.headers['content-type']          # 网页头信息 不存在为 None
    print r.cookies['example_cookie_name']   # 查看 cookie
    print r.history              # 追踪重定向 [<Response [301]>]  开启重定向 allow_redirects=True
```

8.2.1 获取 JSON

```
r = requests.get('https://github.com/timeline.json')
r.json()
```

8.2.2 获取图片

```
from PIL import Image
from StringIO import StringIO
i = Image.open(StringIO(r.content))
```

8.2.3 发送 cookies 到服务器

```
url = 'http://httpbin.org/cookies'
cookies = dict(cookies_are='working')
r = requests.get(url, cookies=cookies)
r.text          '{"cookies": {"cookies_are": "working"}}'
```

8.2.4 在同一个 Session 实例发出的所有请求之间保持 cookies

```
s = requests.Session()
s.get('http://httpbin.org/cookies/set/sessioncookie/123456789')
r = s.get("http://httpbin.org/cookies")
print r.text
```

8.2.5 会话对象能够跨请求保持某些参数

```
s = requests.Session()
s.auth = ('user', 'pass')
s.headers.update({'x-test': 'true'})
s.get('http://httpbin.org/headers', headers={'x-test2': 'true'})  # both 'x-test' and 'x-test2' are sent
```

8.2.6 ssl 证书验证

```
requests.get('https://github.com', verify=True)
requests.get('https://kennethreitz.com', verify=False)  # 忽略证书验证
requests.get('https://kennethreitz.com', cert=('/path/server.crt', '/path/key'))  # 本地指定一个证书 正确 <Response [200]> 错误 SSLError
```

8.2.7 流式上传

```
with open('massive-body') as f:
    requests.post('http://some.url/streamed', data=f)
```

8.2.8 流式请求

```
import requests
import json

r = requests.post('https://stream.twitter.com/1/statuses/filter.json',
    data={'track': 'requests'}, auth=('username', 'password'), stream=True)
```

```
for line in r.iter_lines():
    if line: # filter out keep-alive new lines
        print json.loads(line)
```

8.2.9 自定义身份验证

```
from requests.auth import AuthBase
class PizzaAuth(AuthBase):
    """Attaches HTTP Pizza Authentication to the given Request object."""
    def __init__(self, username):
        # setup any auth-related data here
        self.username = username
    def __call__(self, r):
        # modify and return the request
        r.headers['X-Pizza'] = self.username
        return r
requests.get('http://pizzabin.org/admin', auth=PizzaAuth('kenneth'))
```

8.2.10 基本身份认证

```
from requests.auth import HTTPBasicAuth
requests.get('https://api.github.com/user', auth=HTTPBasicAuth('user', 'pass'))
```

8.2.11 摘要式身份认证

```
from requests.auth import HTTPDigestAuth
url = 'http://httpbin.org/digest-auth/auth/user/pass'
requests.get(url, auth=HTTPDigestAuth('user', 'pass'))
```

8.2.12 代理

```
import requests
proxies = {
  "http": "http://10.10.1.10:3128",
  # "http": "http://user:pass@10.10.1.10:3128/",   # 用户名密码
  "https": "http://10.10.1.10:1080",
}
requests.get("http://example.org", proxies=proxies)
```

```
#也可以设置环境变量之间访问
export HTTP_PROXY="http://10.10.1.10:3128"
export HTTPS_PROXY="http://10.10.1.10:1080"
```

8.3 BeautifulSoup [html\xml 解析器]

```
# BeautifulSoup 中文官方文档
# http://www.crummy.com/software/BeautifulSoup/bs3/documentation.zh.html
# http://www.crummy.com/software/BeautifulSoup/bs4/doc/index.zh.html
# Beautiful Soup 将复杂 HTML 文档转换成一个复杂的树形结构,每个节点都是 Python 对象,所有对象可以归纳为 4 种: Tag , NavigableString , BeautifulSoup , Comment

导入模块
    from BeautifulSoup import BeautifulSoup              # For processing HTML 版本 3.0 已停止更新
    from BeautifulSoup import BeautifulStoneSoup       # For processing XML
    import BeautifulSoup                                 # To get everything
    from bs4 import BeautifulSoup                        # 版本 4.0 bs4 安装: pip install BeautifulSoup4

from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc)      # 解析 html 文本 可以是 requests 提交返回的页面 results.content
print(soup.prettify())              # 输出解析后的结构
print(soup.title)                   # 指定标签内容
print(soup.title.name)              # 标签名
print(soup.title.string)            # 标签内容
print(soup.title.parent.name)       # 上层标签名
print(soup.p)                       # <p class="title"><b>The Dormouse's story</b></p>
print(soup.p['class'])              # u'title'  class 属性值
print(soup.a)                       # 找到第一个 a 标签的标签行
print(soup.find_all('a',limit=2)) # 找到 a 标签的行,最多为 limit 个
print(soup.find(id="link3"))        # 标签内 id 为 link3 的标签行
print(soup.get_text())              # 从文档中获取所有文字内容
soup.find_all("a", text="Elsie")  # 从文档中搜索关键字
soup.find(text=re.compile("sisters"))  # 从文档中正则搜索关键字
soup.find_all("a", class_="sister")      # 按 CSS 搜索
soup.find_all(id='link2',"table",attrs={"class":  "status"},href=re.compile("elsie"))      # 搜索方法
for i in  soup.find_all('a',attrs={"class": "usr-pic"}):      # 循环所有 a 标签的标签行
        if i.a.img:
```

```
                print(i.a.img.get("src"))                    # 取出当前 a 标签
中的连接
        Tag
            # find_all 后循环的值是 Tag 不是字符串 不能直接截取
            tag.text                                  # 文本
            tag.name
            tag.name = "blockquote"          # 查找 name 为 blockquote 的
            tag['class']
            tag.attrs                                 # 按熟悉查找
            tag['class'] = 'verybold'

            del tag['class']                    # 删除
            print(tag.get('class'))          # 打印属性值
            print(i.get('href'))               # 打印连接
```

8.4 json

```
#!/usr/bin/python
import json

#json file temp.json
#{ "name":"00_sample_case1", "description":"an example."}

f = file("temp.json");
s = json.load(f)              # 直接读取 json 文件
print s
f.close

d = {"a":1}
j=json.dumps(d)   # 字典转 json
json.loads(j)       # json 转字典

s = json.loads('{"name":"test", "type":{"name":"seq", "parameter":["1", "2"]}}')
print type(s)       # dic
print s
print s.keys()
print s["type"]["parameter"][1]
```

8.5 cookielib [保留 cookie 登录页面]

ck = cookielib.CookieJar() # 通过 这个就可以实现请求带过去的 COOKIE 与发送回来的 COOKIE 值了。

opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(ck)) # 获取到 COOKIE

urllib2.install_opener(opener) # 此句设置 urllib2 的全局 opener

content = urllib2.urlopen(url).read()

8.5.1 登录 cacti 取图片

```
#encoding:utf8
import urllib2
import urllib
import cookielib
def renrenBrower(url,user,password):
    #查找 form 标签中的 action 提交地址
    login_page = "http://10.10.76.79:81/cacti/index.php"
    try:
        #获得一个 cookieJar 实例
        cj = cookielib.CookieJar()
        #cookieJar 作为参数，获得一个 opener 的实例
        opener=urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
        #伪装成一个正常的浏览器，避免有些 web 服务器拒绝访问
        opener.addheaders = [('User-agent','Mozilla/4.0 (compatible; MSIE 6.0;
Windows NT 5.1)')]
        #生成 Post 数据,含有登陆用户名密码,所有表单内的 input 中 name 值
        data =
urllib.urlencode({"action":"login","login_username":user,"login_password":password})
        #以 post 的方法访问登陆页面，访问之后 cookieJar 会自定保存 cookie
        opener.open(login_page,data)
        #以带 cookie 的方式访问页面
        op=opener.open(url)
        #读取页面源码
        data=op.read()
        #将图片写到本地
        #file("1d.png" , "wb").write(data)
        return data
    except Exception,e:
        print str(e)
print
renrenBrower("http://10.10.76.79:81/cacti/graph_image.php?local_graph_id=1630&rra_id=0&vi
```

ew_type=tree&graph_start=1397525517&graph_end=1397611917","admin","admin")

例子 2

```
import urllib, urllib2, cookielib
import os, time

headers = []

def login():
    cj = cookielib.CookieJar()
    opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
    login_url =
r'http://zhixing.bjtu.edu.cn/member.php?mod=logging&action=login&loginsubmit=yes&infloat=yes&lssubmit=yes&inajax=1'
    login_data = urllib.urlencode({'cookietime': '2592000', 'handlekey': 'ls', 'password': 'xxx',
                'quickforward': 'yes', 'username': 'GuoYuan'})
    opener.addheaders = [('Host', 'zhixing.bjtu.edu.cn'),
                ('User-Agent', 'Mozilla/5.0 (Ubuntu; X11; Linux i686; rv:8.0) Gecko/20100101 Firefox/8.0'),
                ('Accept', 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'),
                ('Accept-Language', 'en-us,en;q=0.5'),
                ('Accept-Encoding', 'gzip, deflate'),
                ('Accept-Charset', 'ISO-8859-1,utf-8;q=0.7,*;q=0.7'),
                ('Connection', 'keep-alive'),
                ('Referer', 'http://zhixing.bjtu.edu.cn/forum.php'),]
    opener.open(login_url, login_data)
    return opener

if __name__ == '__main__':
    opener = login()

    url =
r'http://zhixing.bjtu.edu.cn/forum.php?mod=topicadmin&action=moderate&optgroup=2&modsubmit=yes&infloat=yes&inajax=1'
    data = {'fid': '601', 'formhash': '0cdd1596', 'frommodcp': '', 'handlekey': 'mods',
            'listextra': 'page%3D62', 'moderate[]': '496146', 'operations[]': 'type', 'reason': '...',
            'redirect': r'http://zhixing.bjtu.edu.cn/thread-496146-1-1.html', 'typeid': '779'}
    data2 = [(k, v) for k,v in data.iteritems()]
```

```
cnt = 0
for tid in range(493022, 496146 + 1):
    cnt += 1
    if cnt % 20 == 0: print
    print tid,

    data2.append(('moderate[]', str(tid)))
    if cnt % 40 == 0 or cnt == 496146:
        request = urllib2.Request(url=url, data=urllib.urlencode(data2))
        print opener.open(request).read()
        data2 = [(k, v) for k,v in data.iteritems()]
```

8.6 httplib [http 协议的客户端]

```
import httplib
conn3 = httplib.HTTPConnection('www.baidu.com',80,True,10)
```

8.7 查看网页图片尺寸类型

```
#将图片读入内存
#!/usr/bin/env python
#encoding=utf-8
import cStringIO, urllib2, Image
url = 'http://www.01happy.com/wp-content/uploads/2012/09/bg.png'
file = urllib2.urlopen(url)
tmpIm = cStringIO.StringIO(file.read())
im = Image.open(tmpIm)
print im.format, im.size, im.mode
```

8.8 爬虫

```
#!/usr/bin/env python
#encoding:utf-8
#sudo pip install BeautifulSoup

import requests
```

```
from BeautifulSoup import BeautifulSoup
import re

baseurl = 'http://blog.sina.com.cn/s/articlelist_1191258123_0_1.html'

r = requests.get(baseurl)

for url in re.findall('<a.*?</a>', r.content, re.S):
    if url.startswith('<a title='):
        with open(r'd:/final.txt', 'ab') as f:
            f.write(url + '\n')

linkfile = open(r'd:/final.txt', 'rb')
soup = BeautifulSoup(linkfile)
for link in soup.findAll('a'):
    #print link.get('title') + ':        ' + link.get('href')
    ss = requests.get(link.get('href'))
    for content in re.findall('<div id="sina_keyword_ad_area2" class="articalContent
">.*?</div>', ss.content, re.S):
        with open(r'd:/myftp/%s.txt'%link.get('title').strip('<>'), 'wb') as f:
            f.write(content)
            print '%s      has been copied.' % link.get('title')
```

8.9 反垃圾邮件提交申诉

```
#很遗憾，反垃圾邮件联盟改版后加了验证码

#!/usr/bin/env python
#encoding:utf-8
import requests
import re

IpList=['113.212.91.25','113.212.91.23']
QueryAdd='http://www.anti-spam.org.cn/Rbl/Query/Result'
ComplaintAdd='http://www.anti-spam.org.cn/Rbl/Getout/Submit'
data = {
'CONTENT':'''我们是一家正规的 XXX。xxxxxxx。恳请将我们的发送服务器 IP 移出黑
名单。谢谢！
处理措施：
1.XXXX。
2.XXXX。''',
'CORP':'abc.com',
```

```
        'WWW':'www.abc.cm',
        'NAME':'def',
        'MAIL':'def@163.com.cn',
        'TEL':'010-50000000',
        'LEVEL':'0',
        }

        for Ip in IpList:
            query = requests.post(url=QueryAdd, data={'IP':Ip})                    # 黑名
单查询
            if query.ok:
                if re.findall(u'\u7533\u8bc9\u8131\u79bb', query.text, re.S):      # 查找关
键字 申诉脱离 既表明在黑名单中
                    data['IP']=Ip
                    complaint = requests.post(url=ComplaintAdd, data=data)         # 提
交申诉
                    if complaint.ok:
                        if
re.findall(u'\u60a8\u7684\u9ed1\u540d\u5355\u8131\u79bb\u7533\u8bf7\u5df2\u63d0\u4ea4
', complaint.text, re.S):
                            status='申请提交'
                        elif
re.findall(u'\u8131\u79bb\u7533\u8bf7\u5df2\u88ab\u4ed6\u4eba\u63d0\u4ea4',
complaint.text, re.S):
                            status='重复提交'
                        elif
re.findall(u'\u7533\u8bf7\u7531\u4e8e\u8fd1\u671f\u5185\u6709\u88ab\u62d2\u7edd\u7684
\u8bb0\u5f55', complaint.text, re.S):
                            status='近期拒绝'
                        else:
                            status='异常'
                    else:
                        status='正常'
                    print '%s    %s' %(Ip,status)
```

8.10 有道词典

```
        #!/usr/bin/env python
        import requests
        from bs4 import BeautifulSoup
        # bs4 安装: pip install BeautifulSoup4
```

```
def youdao(word):
    url = r'http://dict.youdao.com/search?le=eng&q={0}'.format(word.strip())
    r = requests.get(url)
    if r.ok:
        soup = BeautifulSoup(r.content)
        div = soup.find_all('div', class_='trans-container')[:1]       # find_all 是 bs4 的
方法
        ul = BeautifulSoup(str(div[0]))
        li = ul.find_all('li')
        for mean in li:
            print mean.text

def query():
    print('Created by @littlepy, QQ:185635687')
    while True:
        word = raw_input('>>>')
        youdao(word)

if __name__ == '__main__':
    query()
```

8.11 python 启动 http 服务提供访问或下载

```
python -m SimpleHTTPServer    9900
```

9 8 并发

#线程安全/竞争条件,锁/死锁检测,同步/异步,阻塞/非阻塞,epoll 非阻塞 IO,信号量/事件,线程池,生产消费模型,伪并发,微线程,协程

#Stackless Python 是 Python 编程语言的一个增强版本，它使程序员从基于线程的编程方式中获得好处，并避免传统线程所带来的性能与复杂度问题。Stackless 为 Python 带来的微线程扩展，是一种低开销、轻量级的便利工具

9.1 threading 多线程

9.1.1 thread

```
start_new_thread(function,args kwargs=None)      # 产生一个新的线程
allocate_lock()                                  # 分配一个 LockType 类型的锁对象
exit()                                           # 让线程退出
acquire(wait=None)                               # 尝试获取锁对象
locked()                                         # 如果获取了锁对象返回 True
release()                                        # 释放锁
```

9.1.2 thread 例子

```
#!/usr/bin/env python
#thread_test.py
#不支持守护进程
import thread
from time import sleep,ctime

loops = [4,2]

def loop(nloop,nsec,lock):
    print 'start loop %s at:%s' % (nloop,ctime())
    sleep(nsec)
    print 'loop %s done at: %s' % (nloop, ctime())
    lock.release()                    # 分配已获得的锁,操作结束后释放相应的锁通知主线程

def main():
    print 'starting at:',ctime()
    locks = []
    nloops = range(len(loops))

    for i in nloops:
        lock = thread.allocate_lock()      # 创建一个锁
        lock.acquire()                     # 调用各个锁的 acquire()函数获得锁

        locks.append(lock)                 # 把锁放到锁列表 locks 中
    for i in nloops:
        thread.start_new_thread(loop,(i,loops[i],locks[i]))    # 创建线程
    for i in nloops:
```

```
            while locks[i].locked():pass          # 等待全部解锁才继续运行
        print 'all DONE at:',ctime()

if __name__ == '__main__':
    main()
```

thread 例子 1

```
#coding=utf-8
import thread,time,os

def f(name):
        i =3
        while i:
                time.sleep(1)
                print name
                i -= 1
        # os._exit()     会把整个进程关闭
        os._exit(22)

if __name__ == '__main__':
        thread.start_new_thread(f,("th1",))
        while 1:
                pass
        os._exit(0)
```

9.1.3 threading

```
Thread                          # 表示一个线程的执行的对象
    start()                     # 开始线程的执行
    run()                       # 定义线程的功能的函数(一般会被子类重写)
    join(timeout=None)     # 允许主线程等待线程结束,程序挂起,直到线程结束;如果给了 timeout,则最多等待 timeout 秒.
    getName()                   # 返回线程的名字
    setName(name)               # 设置线程的名字
    isAlive()                   # 布尔标志,表示这个线程是否还在运行中
    isDaemon()                  # 返回线程的 daemon 标志
    setDaemon(daemonic)    # 后台线程,把线程的 daemon 标志设置为 daemonic(一定要在调用 start()函数前调用)
    # 默认主线程在退出时会等待所有子线程的结束。如果希望主线程不等待子线程，而是在退出时自动结束所有的子线程，就需要设置子线程为后台线程(daemon)
    Lock                        # 锁原语对象
    Rlock                       # 可重入锁对象.使单线程可以在此获得已获得了的锁(递
```

归锁定)

Condition # 条件变量对象能让一个线程停下来,等待其他线程满足了某个条件.如状态改变或值的改变

Event # 通用的条件变量.多个线程可以等待某个事件的发生,在事件发生后,所有的线程都会被激活

Semaphore # 为等待锁的线程提供一个类似等候室的结构

BoundedSemaphore # 与 Semaphore 类似,只是不允许超过初始值

Time # 与 Thread 相似,只是他要等待一段时间后才开始运行

activeCount() # 当前活动的线程对象的数量

currentThread() # 返回当前线程对象

enumerate() # 返回当前活动线程的列表

settrace(func) # 为所有线程设置一个跟踪函数

setprofile(func) # 为所有线程设置一个 profile 函数

threading 例子 1

```
#!/usr/bin/env python
#encoding:utf8
import threading
from Queue import Queue
from time import sleep,ctime

class ThreadFunc(object):
    def __init__(self,func,args,name=''):
        self.name=name
        self.func=func                  # loop
        self.args=args                  # (i,iplist[i],queue)
    def __call__(self):
        apply(self.func,self.args)      # 函数 apply() 执行 loop 函数并传递元组参数
def loop(nloop,ip,queue):
    print 'start',nloop,'at:',ctime()
    queue.put(ip)
    sleep(2)
    print 'loop',nloop,'done at:',ctime()
if __name__ == '__main__':
    threads = []
    queue = Queue()
    iplist = ['192.168.1.2','192.168.1.3','192.168.1.4','192.168.1.5','192.168.1.6','192.168.1.7','192.168.1.8']
    nloops = range(len(iplist))

    for i in nloops:
        t =
```

```
threading.Thread(target=ThreadFunc(loop,(i,iplist[i],queue),loop.__name__))
                threads.append(t)
        for i in nloops:
                threads[i].start()
        for i in nloops:
                threads[i].join()
        for i in nloops:
                print queue.get()
```

threading 例子 2

```
#!/usr/bin/env python
#encoding:utf8
from Queue import Queue
import random,time,threading

class Producer(threading.Thread):
    def __init__(self, t_name, queue):
        threading.Thread.__init__(self, name=t_name)
        self.data=queue
    def run(self):
        for i in range(5):
            print "%s: %s is producing %d to the queue!\n" %(time.ctime(), self.getName(), i)
            self.data.put(i)
            self.data.put(i*i)
            time.sleep(2)
        print "%s: %s finished!" %(time.ctime(), self.getName())

class Consumer(threading.Thread):
    def __init__(self, t_name, queue):
        threading.Thread.__init__(self, name=t_name)
        self.data=queue
    def run(self):
        for i in range(10):
            val = self.data.get()
            print "%s: %s is consuming. %d in the queue is consumed!\n" %(time.ctime(), self.getName(), val)
        print "%s: %s finished!" %(time.ctime(), self.getName())

if __name__ == '__main__':
    queue = Queue()
    producer = Producer('Pro.', queue)
    consumer = Consumer('Con.', queue)
```

```
        producer.start()
        consumer.start()
        producer.join()
        consumer.join()
```

threading 例子 3

```
    # 启动线程后自动执行 run 函数其他不可以
    import threading
    import time

    class Th(threading.Thread):
        def __init__(self,name):
            threading.Thread.__init__(self)
            self.t_name=name
            self.daemon = True        # 默认为 false，让主线程等待处理完成
        def run(self):
            time.sleep(1)
            print "this is " + self.t_name

    if __name__ == '__main__':
        thread1 = Th("Th_1")
        thread1.start()
```

threading 例子 4

```
    import threading
    import time
    class Th(threading.Thread):
    def __init__(self,thread_name):
        threading.Thread.__init__(self)
        self.setName(thread_name)
    def run(self):
        threadLock.acquire()
        print self.getName()
        for i in range(3):
            time.sleep(1)
            print str(i)
        print self.getName() +    " is over"
        threadLock.release()

    if __name__ == '__main__':
        threadLock = threading.Lock()
        thread1 = Th("Th_1")
```

```
    thread2 = Th("Th_2")
    thread1.start()
    thread2.start()
```

9.1.4 后台线程

```
import threading
import time,random

class MyThread(threading.Thread):
    def run(self):
        wait_time=random.randrange(1,10)
        print "%s will wait %d seconds" % (self.name, wait_time)
        time.sleep(wait_time)
        print "%s finished!" % self.name

if __name__=="__main__":
    for i in range(5):
        t = MyThread()
        t.setDaemon(True)       # 设置为后台线程,主线程完成时不等待子线程完成就结束
        t.start()
```

9.1.5 threading 控制最大并发_查询日志中 IP 信息

```
#!/usr/bin/env python
#coding:utf-8
import urllib2
import json
import threading
import time

'''
by:某大牛
QQ:185635687
这个是多线程并发控制.如果要改成多进程，只需把 threading 换成 mulitprocessing.Process ， 对， 就是换个名字而已.
'''
```

```
#获取 ip 及其出现次数
def ip_dic(file_obj, dic):
    for i in file_obj:
        if i:
            ip=i.split('-')[0].strip()
            if ip in dic.keys():
                dic[ip]=dic[ip] + 1
            else:
                dic[ip]=1
    return dic.iteritems()

#目标函数
def get_data(url, ipcounts):
    data=urllib2.urlopen(url).read()
    datadict=json.loads(data)
    fdata =
u"ip:%s---%s,%s,%s,%s,%s"  %(datadict["data"]["ip"],ipcounts,datadict["data"]["country"],datadict
["data"]["region"],datadict["data"]["city"],datadict["data"]["isp"])
    print fdata

#多线程
def threads(iters):
    thread_pool = []
    for k in iters:
        url = "http://ip.taobao.com/service/getIpInfo.php?ip="
        ipcounts = k[1]
        url = (url + k[0]).strip()
        t = threading.Thread(target=get_data, args=(url, ipcounts))
        thread_pool.append(t)
    return thread_pool

#控制多线程
def startt(t_list, max,second):
    l = len(t_list)
    n = max
    while l > 0:
        if l > max:
            nl = t_list[:max]
            t_list = t_list[max:]
            for t in nl:
                t.start()
            time.sleep(second)
            for t in nl:
                t.join()
```

```
                print '*'*15,    str(n)+ ' ip has been queried'+'*'*15
                n += max
                l = len(t_list)
                continue
            elif l <= max:
                nl = t_list
                for t in nl:
                    t.start()
                for t in nl:
                    t.join()
                print '>>> Totally ' + str(n+l ) + ' ip has been queried'
                l = 0

if __name__ =="__main__":
    dic={}
    with open('access.log') as file_obj:
        it = ip_dic(file_obj, dic)
        t_list= threads(it)
        startt(t_list, 15, 1)
```

9.1.6 多线程取队列

```
#!/usr/bin/python

import Queue
import threading
import time

exitFlag = 0

class myThread (threading.Thread):
    def __init__(self, threadID, name, q):
        threading.Thread.__init__(self)
        self.threadID = threadID
        self.name = name
        self.q = q
    def run(self):
        print "Starting " + self.name
        process_data(self.name, self.q)
        print "Exiting " + self.name

def process_data(threadName, q):
```

```
        while not exitFlag:           # 死循环等待
            queueLock.acquire()
            if not q.empty():       # 判断队列是否为空
                data = q.get()
                print "%s processing %s" % (threadName, data)
            queueLock.release()
            time.sleep(1)

    threadList = ["Thread-1", "Thread-2", "Thread-3"]
    nameList = ["One", "Two", "Three", "Four", "Five"]
    queueLock = threading.Lock()        # 锁与队列并无任何关联，其他线程也进行取锁操作的时候就会检查是否有被占用，有就阻塞等待解锁为止
    workQueue = Queue.Queue(10)
    threads = []
    threadID = 1

    # Create new threads
    for tName in threadList:
        thread = myThread(threadID, tName, workQueue)
        thread.start()
        threads.append(thread)
        threadID += 1

    # Fill the queue
    queueLock.acquire()
    for word in nameList:
        workQueue.put(word)
    queueLock.release()

    # Wait for queue to empty
    while not workQueue.empty():     # 死循环判断队列被处理完毕
        pass

    # Notify threads it's time to exit
    exitFlag = 1

    # Wait for all threads to complete
    for t in threads:
        t.join()
    print "Exiting Main Thread"
```

9.2 Queue 通用队列

```
q=Queue(size)          # 创建大小 size 的 Queue 对象
qsize()                # 返回队列的大小(返回时候,可能被其他进程修改,近似值)
empty()                # 如果队列为空返回 True，否则 Fales
full()                 # 如果队列已满返回 True，否则 Fales
put(item,block0)       # 把 item 放到队列中,如果给了 block(不为 0),函数会一直阻塞到队列中有空间为止
get(block=0)           # 从队列中取一个对象,如果给了 block(不为 0),函数会一直阻塞到队列中有对象为止
get_nowait             # 默认 get 阻塞，这个不阻塞
```

9.3 multiprocessing [多进程并发]

9.3.1 多线程

```
import urllib2
from multiprocessing.dummy import Pool as ThreadPool

urls=['http://www.baidu.com','http://www.sohu.com']

pool=ThreadPool(4)     # 线程池
results=pool.map(urllib2.urlopen,urls)
pool.close()
pool.join()
```

9.3.2 多进程并发

```
#!/usr/bin/env python
#encoding:utf8
from multiprocessing import Process
import time,os
def f(name):
    time.sleep(1)
```

```
    print 'hello ',name
    print os.getppid()     # 取得父进程 ID
    print os.getpid()      # 取得进程 ID
process_list = []

for i in range(10):
    p = Process(target=f,args=(i,))
    p.start()
    process_list.append(p)
for j in process_list:
    j.join()
```

9.3.3 Queue 进程间通信

```
from multiprocessing import Process,Queue
import time
def f(name):
    time.sleep(1)
    q.put(['hello'+str(name)])
process_list = []
q = Queue()
if __name__ == '__main__':
    for i in range(10):
        p = Process(target=f,args=(i,))
        p.start()
        process_list.append(p)
    for j in process_list:
        j.join()
    for i in range(10):
        print q.get()
```

9.3.4 Pipe 管道

```
from multiprocessing import Process,Pipe
import time
import os

def f(conn,name):
    time.sleep(1)
```

```
        conn.send(['hello'+str(name)])
        print os.getppid(),'----------',os.getpid()
process_list = []
parent_conn,child_conn = Pipe()
if __name__ == '__main__':
        for i in range(10):
                p = Process(target=f,args=(child_conn,i))
                p.start()
                process_list.append(p)
        for j in process_list:
                j.join()
        for p in range(10):
                print parent_conn.recv()
```

9.3.5 进程间同步

```
#加锁，使某一时刻只有一个进程 print
from multiprocessing import Process,Lock
import time
import os

def f(name):
        lock.acquire()
        time.sleep(1)
        print 'hello--'+str(name)
        print os.getppid(),'----------',os.getpid()
        lock.release()
process_list = []
lock = Lock()
if __name__ == '__main__':
        for i in range(10):
                p = Process(target=f,args=(i,))
                p.start()
                process_list.append(p)
        for j in process_list:
                j.join()
```

9.3.6 共享内存

```
# 通过使用 Value 或者 Array 把数据存储在一个共享的内存表中
```

'd'和'i'参数是 num 和 arr 用来设置类型，d 表示一个双精浮点类型，i 表示一个带符号的整型。

```
from multiprocessing import Process,Value,Array
import time
import os

def f(n,a,name):
    time.sleep(1)
    n.value = name * name
    for i in range(len(a)):
        a[i] = -i
process_list = []
if __name__ == '__main__':
    num = Value('d',0.0)
    arr = Array('i',range(10))
    for i in range(10):
        p = Process(target=f,args=(num,arr,i))
        p.start()
        process_list.append(p)
    for j in process_list:
        j.join()
    print num.value
    print arr[:]
```

9.3.7 manager

```
# 比共享内存灵活,但缓慢
# 支持list,dict,Namespace,Lock,Semaphore,BoundedSemaphore,Condition,Event,Queue,Value,Array
from multiprocessing import Process,Manager
import time
import os

def f(d,name):
    time.sleep(1)
    d[name] = name * name
    print d
process_list = []
if __name__ == '__main__':
    manager = Manager()
    d = manager.dict()
    for i in range(10):
```

```
            p = Process(target=f,args=(d,i))
            p.start()
            process_list.append(p)
        for j in process_list:
            j.join()
            print d
```

9.3.8 最大并发数

```
import multiprocessing
import time,os

result = []
def run(h):
    print 'threading:' ,h,os.getpid()
p = multiprocessing.Pool(processes=20)

for i in range(100):
    result.append(p.apply_async(run,(i,)))
p.close()

for res in result:
    res.get(timeout=5)
```

10 9 框架

10.1 flask [微型网络开发框架]

```
# http://dormousehole.readthedocs.org/en/latest/
# html 放在 ./templates/    js 放在 ./static/

request.args.get('page', 1)             # 获取参数 ?page=1
request.json                                # 获取传递的整个 json 数据
request.form.get("host",'127')          # 获取表单值
```

10.1.1 简单实例 # 接收数据和展示

```
import MySQLdb as mysql
from flask import Flask, request

app = Flask(__name__)
db.autocommit(True)
c = db.cursor()

"""
CREATE TABLE `statusinfo` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `hostname` varchar(32) NOT NULL,
  `load` float(10) NOT NULL DEFAULT 0.00,
  `time` int(15) NOT NULL,
  `memtotal` int(15) NOT NULL,
  `memusage` int(15) NOT NULL,
  `memfree` int(15) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=161 DEFAULT CHARSET=utf8;
"""

@app.route("/collect", methods=["GET", "POST"])
def collect():
    sql = ""
    if request.method == "POST":
        data = request.json                                    # 获取传递的 json
        hostname = data["Host"]
        load = data["LoadAvg"]
        time = data["Time"]
        memtotal = data["MemTotal"]
        memusage = data["MemUsage"]
        memfree = data["MemFree"]

        try:
            sql = "INSERT INTO `statusinfo` (`hostname`,`load`,`time`,`memtotal`,`memusage`,`memfree`) VALUES('%s', %s, %s, %s, %s, %s);" % (hostname, load,time,memtotal,memusage,memfree)
            ret = c.execute(sql)
            return 'ok'
        except mysql.IntegrityError:
            return 'errer'
```

```
@app.route("/show", methods=["GET", "POST"])
def show():
    try:
        hostname = request.form.get("hostname")        # 获取表单方式的变量值
        sql = "SELECT `load` FROM `statusinfo` WHERE hostname = '%s';" %(hostname)
        c.execute(sql)
        ones = c.fetchall()
        return render_template("sysstatus.html", data=ones, sql = sql)
    except:
        print 'hostname null'

from flask import render_template
@app.route("/xxx/<name>")
def hello_xx(name):
    return render_template("sysstatus.html", name='teach')

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=50000, debug=True)
```

10.2 twisted [非阻塞异步服务器框架]

用来进行网络服务和应用程序的编程。虽然 Twisted Matrix 中有大量松散耦合的模块化组件，但该框架的中心概念还是非阻塞异步服务器这一思想。对于习惯于线程技术或分叉服务器的开发人员来说，这是一种新颖的编程风格，但它却能在繁重负载的情况下带来极高的效率。

```
pip install twisted

from twisted.internet import protocol, reactor, endpoints

class Echo(protocol.Protocol):
    def dataReceived(self, data):
        self.transport.write(data)
class EchoFactory(protocol.Factory):
    def buildProtocol(self, addr):
        return Echo()

endpoints.serverFromString(reactor, "tcp:1234").listen(EchoFactory())
reactor.run()
```

10.3 greenlet [微线程/协程框架]

更加原始的微线程的概念,没有调度,或者叫做协程。这在你需要控制你的代码时很有用。你可以自己构造微线程的 调度器；也可以使用"greenlet"实现高级的控制流。例如可以重新创建构造器；不同于 Python 的构造器，我们的构造器可以嵌套的调用函数，而被嵌套的函数也可以 yield 一个值。

pip install greenlet

10.4 tornado [极轻量级 Web 服务器框架]

高可伸缩性和 epoll 非阻塞 IO,响应快速,可处理数千并发连接,特别适用用于实时的 Web 服务

```
# http://www.tornadoweb.cn/documentation
pip install tornado

import tornado.ioloop
import tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")

application = tornado.web.Application([
    (r"/", MainHandler),
])

if __name__ == "__main__":
    application.listen(8888)
    tornado.ioloop.IOLoop.instance().start()
```

10.5 Scrapy [web 抓取框架]

Python 开发的一个快速,高层次的屏幕抓取和 web 抓取框架，用于抓取 web 站点并从页面中提取结构化的数据。Scrapy 用途广泛，可以用于数据挖掘、监测和自动化测试。

```
pip install scrapy

from scrapy import Spider, Item, Field
```

```
class Post(Item):
    title = Field()

class BlogSpider(Spider):
    name, start_urls = 'blogspider', ['http://blog.scrapinghub.com']

    def parse(self, response):
        return [Post(title=e.extract()) for e in response.css("h2 a::text")]

scrapy runspider myspider.py
```

10.6 django [重量级 web 框架]

10.7 bottle [轻量级的 Web 框架]

11 10 例子

11.1 小算法

11.1.1 斐波那契

```
#将函数结果作为列表可用于循环
def fab(max):
n, a, b = 0, 0, 1
while n < max:
    yield b
    a, b = b, a + b
    n = n + 1
for n in fab(5):
    print n
```

11.1.2 乘法口诀

```
#!/usr/bin/python
for i in range(1,10):
    for j in range(1,i+1):
        print j,'*',i,'=',j*i,
    else:
        print ''
```

11.1.3 最小公倍数

```
# 1-70 的最小公倍数
def c(m,n):
        a1=m
        b1=n
        r=n%m
        while r!=0:
                n=m
                m=r
                r=n%m
        return (a1*b1)/m
d=1
for i in range(3,71,2):
        d = c(d,i)
print d
```

11.1.4 排序算法

11.1.4.1 插入排序

```
def insertion_sort(sort_list):
    iter_len = len(sort_list)
    if iter_len < 2:
        return sort_list
    for i in range(1, iter_len):
```

```
        key = sort_list[i]
        j = i - 1
        while j>=0 and sort_list[j]>key:
            sort_list[j+1] = sort_list[j]
            j -= 1
        sort_list[j+1] = key
    return sort_list
```

11.1.4.2 选择排序

```
def selection_sort(sort_list):
    iter_len = len(sort_list)
    if iter_len < 2:
        return sort_list
    for i in range(iter_len-1):
        smallest = sort_list[i]
        location = i
        for j in range(i, iter_len):
            if sort_list[j] < smallest:
                smallest = sort_list[j]
                location = j
        if i != location:
            sort_list[i], sort_list[location] = sort_list[location], sort_list[i]
    return sort_list
```

11.1.4.3 冒泡排序算法

```
def bubblesort(numbers):
    for j in range(len(numbers)-1,-1,-1):
        for i in range(j):
            if numbers[i]>numbers[i+1]:
                numbers[i],numbers[i+1] = numbers[i+1],numbers[i]
            print(i,j)
            print(numbers)
```

11.1.5 二分算法

```
#python 2f.py 123456789 4
```

```
# list('123456789')    =    ['1', '2', '3', '4', '5', '6', '7', '8', '9']
#!/usr/bin/env python
import sys

def search2(a,m):
    low = 0
    high = len(a) - 1
    while(low <= high):
        mid = (low + high)/2
        midval = a[mid]

        if midval < m:
            low = mid + 1
        elif midval > m:
            high = mid - 1
        else:
            print mid
            return mid
    print -1
    return -1

if __name__ == "__main__":
    a = [int(i) for i in list(sys.argv[1])]
    m = int(sys.argv[2])
    search2(a,m)
```

11.2 将字典中所有 time 去掉

```
a={'version01': {'nba': {'timenba': 'valuesasdfasdf', 'nbanbac': 'vtimefasdf', 'userasdf':
'vtimasdf'}}}
eval(str(a).replace("time",""))
```

11.3 PIL 图像处理

```
import Image
im = Image.open("j.jpg")                    # 打开图片
print im.format, im.size, im.mode     # 打印图像格式、像素宽和高、模式
# JPEG (440, 330) RGB
im.show()                                        # 显示最新加载图像
```

```
box = (100, 100, 200, 200)
region = im.crop(box)                    # 从图像中提取出某个矩形大小的图像
```

11.4 图片等比缩小

```
# -*- coding: cp936 -*-
import Image
import glob, os

#图片批处理
def timage():
    for files in glob.glob('D:\\1\\*.JPG'):
        filepath,filename = os.path.split(files)
        filterame,exts = os.path.splitext(filename)
        #输出路径
        opfile = r'D:\\22\\'
        #判断 opfile 是否存在，不存在则创建
        if (os.path.isdir(opfile)==False):
            os.mkdir(opfile)
        im = Image.open(files)
        w,h = im.size
        #im_ss = im.resize((400,400))
        #im_ss = im.convert('P')
        im_ss = im.resize((int(w*0.12), int(h*0.12)))
        im_ss.save(opfile+filterame+'.jpg')

if __name__=='__main__':
    timage()
```

11.5 取系统返回值赋给序列

```
cmd = os.popen("df -Ph|awk 'NR!=1{print $5}'").readlines();
cmd = os.popen('df -h').read().split('\n')
cmd = os.popen('lo 2>&1').read()

#取磁盘使用空间
import commands
df = commands.getoutput("df -hP")
[ x.split()[4] for x in df.split("\n") ]
```

```
[ (x.split()[0],x.split()[4]) for x in df.split("\n") if x.split()[4].endswith("%") ]
```

11.6 打印表格

```
map = [["a","b","c"],
       ["d","e","f"],
       ["g","h","i"]]
def print_board():
    for i in range(0,3):
        for j in range(0,3):
            print "|",map[i][j],
            #if j != 2:
        print '|'
```

11.7 生成 html 文件表格

```
log_file = file('check.html', 'w')
log_file.write("""
<!DOCTYPE HTML>
<html lang="utr-8">
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
<table align='center' border='0' cellPadding='0'  style='font-size:24px;'><tr ><td>状态统计</td></tr></table>
<style>.font{font-size:13px}</style>
<table  align='center' border='1' borderColor=gray cellPadding=3 width=1350 class='font'>
<tr style='background-color:#666666'>
  <th width=65>IP</th>
  <th width=65>状态</th>
</tr>
""")
for i in list:
    log_file.write('<tr><td>%s</td><td>%s</td></tr>\n' %(i.split()[0],i.split()[1]) )
log_file.write("""
</table>
```

```
</body>
</html>
""")
log_file.flush()
log_file.close()
```

11.8 井字游戏

```
#!/usr/bin/python
# http://www.admin10000.com/document/2506.html
def print_board():
    for i in range(0,3):
        for j in range(0,3):
            print map[2-i][j],
            if j != 2:
                print "|",
        print ""

def check_done():
    for i in range(0,3):
        if map[i][0] == map[i][1] == map[i][2] != " " \
        or map[0][i] == map[1][i] == map[2][i] != " ":
            print turn, "won!!!"
            return True

    if map[0][0] == map[1][1] == map[2][2] != " " \
    or map[0][2] == map[1][1] == map[2][0] != " ":
        print turn, "won!!!"
        return True

    if " " not in map[0] and " " not in map[1] and " " not in map[2]:
        print "Draw"
        return True

    return False

turn = "X"
map = [[" "," "," "],
       [" "," "," "],
       [" "," "," "]]
done = False
```

```
while done != True:
    print_board()

    print turn, "'s turn"
    print

    moved = False
    while moved != True:
        print "Please select position by typing in a number between 1 and 9, see below for which number that is which position..."
        print "7|8|9"
        print "4|5|6"
        print "1|2|3"
        print

        try:
            pos = input("Select: ")
            if pos <=9 and pos >=1:
                Y = pos/3
                X = pos%3
                if X != 0:
                    X -=1
                else:
                     X = 2
                     Y -=1

                if map[Y][X] == " ":
                    map[Y][X] = turn
                    moved = True
                    done = check_done()

                    if done == False:
                        if turn == "X":
                            turn = "O"
                        else:
                            turn = "X"

        except:
            print "You need to add a numeric value"
```

11.9 网段划分

题目

192.168.1
192.168.3
192.168.2
172.16.3
192.16.1
192.16.2
192.16.3
10.0.4

输出结果：
192.16.1-192.16.3
192.168.1-192.168.3
172.16.3
10.0.4

答案

```
#!/usr/bin/python

f = file('a.txt')
c = f.readlines()
dic={}

for i in c:
    a=i.strip().split('.')
    if a[0]+'.'+a[1] in dic.keys():
        key=dic["%s.%s" %(a[0],a[1])]
    else:
        key=[]
    key.append(a[2])
    dic[a[0]+'.'+a[1]]=sorted(key)

for x,y in dic.items():
    if y[0] == y[-1]:
        print '%s.%s' %(x,y[0])
    else:
        print '%s.%s-%s.%s' %(x,y[0],x,y[-1])
```

11.10 统计日志 IP

打印出独立 IP，并统计独立 IP 数

219.140.190.130 - - [23/May/2006:08:57:59 +0800] "GET /fg172.exe HTTP/1.1" 200 2350253

221.228.143.52 - - [23/May/2006:08:58:08 +0800] "GET /fg172.exe HTTP/1.1" 206 719996

221.228.143.52 - - [23/May/2006:08:58:08 +0800] "GET /fg172.exe HTTP/1.1" 206 713242

```
#!/usr/bin/python
dic={}
a=open("a").readlines()
for i in a:
    ip=i.strip().split()[0]
    if ip in dic.keys():
        dic[ip] = dic[ip] + 1
    else:
        dic[ip] = 1
for x,y in dic.items():
    print x," ",y
```

不定期更新，更新下载地址：
http://hi.baidu.com/quanzhou722/item/cf4471f8e23d3149932af2a7

请勿删除信息，植入广告，抵制不道德行为。