NSD Python DAY03

1. 案例1：简化除法判断
2. 案例2：分析apache访问日志
3. 案例3：扫描存活主机
4. 案例4：利用多线程实现ssh并发访问

1 案例1：简化除法判断

1.1 问题

编写mydiv.py脚本，主要要求如下：

- 提示用户输入一个数字作为除数
- 如果用户按下Ctrl+C或Ctrl+D则退出程序
- 如果用户输入非数字字符，提示用户应该输入数字
- 如果用户输入0，提示用户0不能作为除数

1.2 方案

使用if语句判断除数是否合适，需要编写多条语句。有了异常处理，可以本着先做，错了再说的逻辑。直接把除法操作放在try语句中执行，根据产生的异常做相应的处理。

另外，Ctrl+C或Ctrl+D只能通过异常捕获。

异常捕获的语法如下：

```
try:
    A
except:
    B
else:
    C
finally:
    D
```

把可能发生异常的语句放在A里面执行，如果出现异常则执行B语句，没有异常则执行C语句。不管是否出现异常都会执行D语句。

捕获异常时，可以使用多个except语句，每个except语句捕获一个异常，每个异常给定不同的处理方法。也可以把多个异常放在同一个except语句后面，但是务必注意，多个异常写在相同的一行，一定要注括号括起来，放在元组中。

1.3 步骤

Top

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

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

import sys

while True:
    try:
        result = 100 / int( raw_input( 'enter a number: ' ) )
    except ( ValueError, ZeroDivisionError ), e:    #将异常原因保存在变量e中
        print "invalid input:", e
        continue
    except ( EOFError, KeyboardInterrupt ):
        sys.exit( 1 )
    break

print result
```

步骤二：测试脚本执行

```
[root@py01 bin]# ./mydiv.py
enter a number: 0
invalid input: integer division or modulo by zero
enter a number: abc
invalid input: invalid literal for int() with base 10: 'abc'
enter a number: 3
33
```

2 案例2：分析apache访问日志

2.1 问题

编写用于分析apache日志的脚本，主要要求如下：

- 统计每个客户端访问apache服务器的次数
- 将统计信息通过字典的方式显示出来
- 分别统计客户端是Firefox和MSIE的访问次数
- 分别使用函数式编程和面向对象编程的方式实现

2.2 方案

涉及到文本处理时，正则表达式将是一个非常强大的工具。匹配客户端的IP地址，可以使用正则表达式的元字符，匹配字符串可以直接使用字符的表面含义。

Top

入门级程序员的写法，使用顺序的结构，直接编写。这种方法虽然可以得出结果，但是代码难以重用。参考步骤一。

进阶的写法可以采用函数式编程，方便日后再次使用。参考步骤二。

最后，还可以使用OOP的编程方法，先定义一个统计类，该类将正则表达式作为它的数据属性。再定义一个方法，从指定的文件中搜索正则表达式出现的次数，并将其存入到一个字典中。参考步骤三。

2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：简单实现

```
[root@py01 bin]# vim countweb.py

#! /usr/bin/env python

import re

logfile = '/var/log/httpd/access_log'
cDict = {}
patt_ip = '^\d+\.\d+\.\d+\.\d+'        #定义匹配IP地址的正则表达式

with open(logfile) as f:
    for eachLine in f:
        m = re.search(patt_ip, eachLine)
        if m is not None:
            ipaddr = m.group()
            #如果IP地址已在字典中，将其值加1，否则初始值设置为1
            cDict[ipaddr] = cDict.get(ipaddr, 0) + 1

print cDict
```

步骤二：使用函数式编程实现

```
[root@py01 bin]# vim countweb2.py

! /usr/bin/env python

import re

def countPatt(patt, fname):    #定义可以在指定文件中搜索指定字符串的函数
```

Top

```
    cDict = {}
    with open(fname) as f:
        for eachLine in f:
            m = re.search(patt, eachLine)
            if m is not None:
                k = m.group()
                cDict[k] = cDict.get(k, 0) + 1
    return cDict

def test():
    logfile = '/var/log/httpd/access_log'
    patt_ip = '^\d+\.\d+\.\d+\.\d+'
    print countPatt(patt_ip, logfile)
    patt_br = 'Firefox|MSIE'
    print countPatt(patt_br, logfile)

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

3 案例3：扫描存活主机

3.1 问题

编写扫描存活主机的脚本，主要要求如下：

- 调用系统的ping命令进行扫描
- 扫描教室环境下所有存活的主机
- 采用多线程的方式编写
- 方案

os模块的system()函数可以调用系统命令，其返回值是系统命令退出码，也就是如果系统命令成功执行，返回0，如果没有成功执行，返回非零值。

本例扫描主机，可以调用系统的ping命令，通过退出码来判断是否ping通了该主机。如果顺序执行，每个ping操作需要消耗数秒钟，全部的254个地址需要10分钟以上。而采用多线程，可以实现对这254个地址同时执行ping操作，并发的结果就是将执行时间缩短到了10秒钟左右。

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```
[root@py01 bin]# vim mtping.py
#! /usr/bin/env python

```

Top

```
04.  import subprocess
05.  import threading
06.  import sys
07.
08.  def ping(ip):
09.      result = subprocess.call("ping -c2 %s &> /dev/null" % ip, shell=True)
10.      if result:
11.          print "%s: down" % ip
12.      else:
13.          print "%s: up" % ip
14.
15.  if __name__ == '__main__':
16.      if len(sys.argv) != 2:
17.          print "Usage: %s subnet" % sys.argv[0]
18.          sys.exit(1)
19.      net_list = sys.argv[1].split('.')
20.      net = '.'.join(net_list[:-1])
21.      ips = ("%s.%s" %(net, i) for i in range(1, 255))
22.      for ip in ips:
23.          t = threading.Thread(target=ping, args=(ip,))
24.          t.start()
```

步骤二：测试脚本执行

```
01.  [root@py01 bin]# python mtping.py 172.40.51.0
```

脚本接受命令行参数，只要给定网段就可以实现对该网段中所有ip地址的ping操作。

4 案例4：利用多线程实现ssh并发访问

4.1 问题

编写ssh客户端脚本，主要要求如下：

- 在文件中取出所有远程主机IP地址
- 在shell命令行中接受远程服务器IP地址文件、远程服务器密码以及在远程主机上执行的命令
- 通过多线程实现在所有的远程服务器上并发执行命令
- 方案

python的paramiko模块可以实现ssh客户端的功能，使用起来也比较简单。但是当服务器非常多的时候，每台服务器上执行完全相同的简单操作，也会花费大量的时间。 Top

通过ssh加上多线程，可以实现并发访问。为了将程序写的灵活性更强，把要执行的命令以位置参数的方式来提供。

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```
[root@py01 bin]# vim remote_comm.py
#! /usr/bin/env python

import paramiko
import os
import sys
import threading

def remote_comm(host, password, comm):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(host, username='root', password=password)
    stdin, stdout, stderr = ssh.exec_command(comm)
    out = stdout.read()
    err = stderr.read()
    if out:
        print "[%s:out]: %s" %(host, out),
    if err:
        print "%s:Error: %s", (host, err),
    ssh.close()

if __name__ == '__main__':
    if len(sys.argv) != 4:
        print "Usage: %s ipfile password 'comm'" %sys.argv[0]
        sys.exit(1)
    ipfile = sys.argv[1]
    if not os.path.isfile(ipfile):
        print "No such file: %s" %ipfile
        sys.exit(2)
    password = sys.argv[2]
    comm = sys.argv[3]
    with open(ipfile) as fobj:
        for line in fobj:
            ip = line.strip()
            t = threading.Thread(target=remote_comm, args=(ip, password, comm))
            t.start()
```

Top

步骤二：测试脚本执行

```
01.  [root@py01 bin]# python remote_comm.py ipaddr.txt tedu.cn 'useradd bob'
```

脚本接受命令行参数，其中ipaddr.txt是存放所有远程主机ip地址的文件，文件中每个ip地址占一行。tedu.cn是远程主机的密码（所有远程主机密码需要是一致的）。最后的命令需要写在引号中。

该示例执行成功后，会在所有远程主机上创建一个名为bob的用户。

Top