Reboot

python基础教程05 函数

woniuppp

Reboot

上节回顾——文件操作

上节课内容非常简单，主要就是几个函数

- open
- read
- readling
- readlines
- write
- writelines
- close

Reboot

上节回顾——文件操作

- 文件的模式
  - 读
  - 写
  - 追加
- 错误处理
  - try
  - except
  - else
  - finaly

Reboot

上节回顾——前端基础

- html css 和js的关系
- 常用标签介绍
  - html
  - head
  - body
  - p
  - input

Reboot

上节回顾——前端基础

css

- width
- height
- background
- float
- margin

Reboot

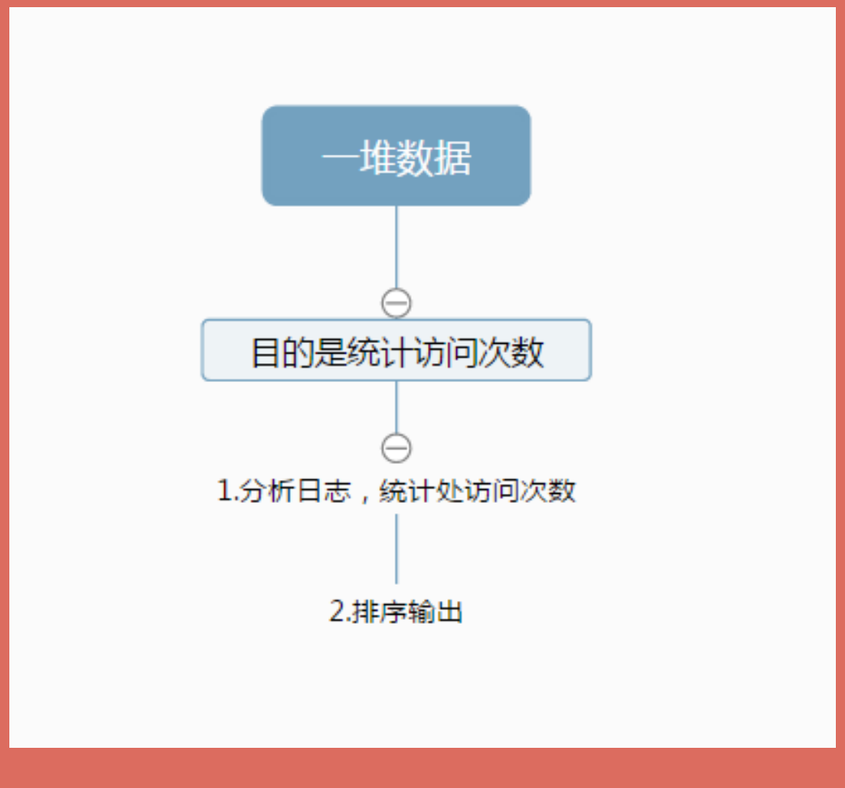
简单的nginx日志分析

学的东西终于好像有点用了

- 题目：日志文件在/home/shre/www_access_20140823.log
- 期待输出一个list，包含里面的http状态，url，ip，访问次数
- 输出按照访问次数排序，输出前十
- 一定要保证自己能独立完成这个题，是后面学习的基础

Reboot

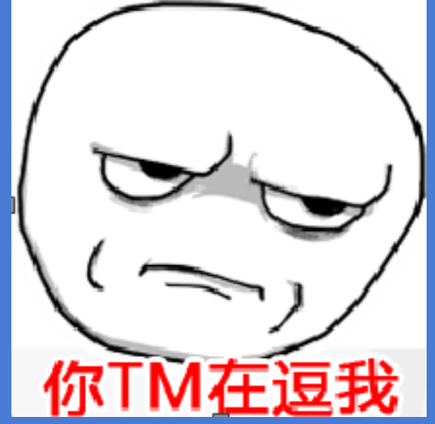
大概的思考步骤



01分析日志，统计出pv

- 读文件，easy
- 需要有一个数据结构来存储统计的数据
- 用list
  - list的每个数据，是一个list，存储着所有的数据
  - 每当有一个新的状态+地址+ip的时候，更新对应list里面的访问次数值
  - 但是问题来了，怎么找是一个问题
  - 每一行数据，都要遍历整个list，明显坑爹

Reboot



查找数据强大的dict

- 状态当一个key，value再来一个dict
- 里面的dict，url当key，同理，嵌套

```
{
    404: {
        'xxx/xxx.img': {
            '192.168.2.1': 2,
            '192.168.2.2': 4
        },
        'aaa/aaa.img':{

        }
    },
    200:{

    }
}
```

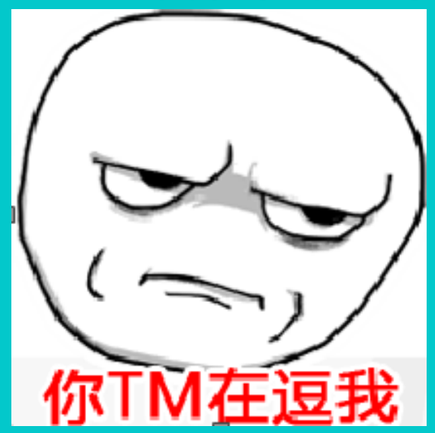
Reboot

第一步搞定

Reboot

第二步

排序---dict好像本来就没啥顺序



Reboot

思路，处理成list 然后冒泡十次，取最后十个

具体看代码

Reboot

函数

- 咱们现在写一些小程序是没问题的
- 如果我们写的log日志分析功能，在其他地方也能用到，该肿么办
- 所以我们需要函数
  - 执行一系列语句
- 代码复用
  - 不用一直复制代码
- 更加易读
  - 通过函数来组织一个功能

Reboot

def 创建一个函数

```
def hello():
    print 'hello'
    print 'world'
hello()
hello()
```

Reboot

return返回一个值

```
def hello():
    return 'hello world'
print hello()
print hello()
```

Reboot

函数的参数,变化的部分

- 参数 只是一个变量

```
def hello(text):
  print 'hello %s' % text
hello('wd')
hello('pc')
```

Reboot

参数 只是一个变量

传递参数，大概就相当于赋值的操作

小练习 ：写一个计算阶乘的函数

Reboot

感受一下两个例子的区别

```
user_name ='wd'
def change_params(name):
    name ='pc'
change_params(user_name)
print user_name
```

```
user_name_list = '[wd','woniu']
def change_params(names):
    names[0] ='pc'
change_params(user_name_list)
print user_name_list
```

Reboot

我们现在所用的参数，都是和位置相关

```
def hello_world(name, word):
    print '%s,%s' % (name,word)
hello_world('wd','hello')
```

```
def hello_world(word,name):
    print '%s,%s' % (name,word)
hello_world('wd','hello')
```

Reboot

我们可以指定参数的名字——关键字参数

代码更易读

```
def hello_world1(name,word):
    print '%s,%s' % (name,word)
def hello_world2(word,name):
    print '%s,%s' % (name,word)

hello_world1(name='wd',word='hello')
hello_world2(name='wd',word='hello')
```

Reboot

定义函数的时候，使用关键字参数，可以指定默认值

```
def hello_world(name='wd',word='hello'):
    print '%s,%s' % (name,word)
hello_world()
hello_world('pc')
hello_world('pc','hehe')
```

Reboot

参数不确定的时候

定义函数的时候，参数前面加一个* 可以收集所有参数

```
def print_params(*params):
    print params
print_params()
print_params(1,2,3,4)
```

Reboot

小练习

函数add_all,把传入的所有参数求和，打印出来

Reboot

先指定位置参数，然后后面的可以用*来匹配

```
def print_params(name,*params):
    print name
    print params
print_params('pc')
print_params('wd',2,3,4)
```

Reboot

收集关键字参数，要用两个 **

```
def print_params(**params):
    print params

print_params()
print_params(name='pc', job='worker')
```

和位置参数一样，也可以先定义确认的参数，不确定的用**

收集

```
def print_params(name,**params):
    print name
    print params

print_params(name='wd',sex='male',location='bj')
print_params(name='pc',job='worker',weight=180)
```

Reboot

尝试和* 一起使用

Reboot

作用域

```
x = 1
def change_global():
    x = 3
change_global()
print x
```

Reboot

作用域

```
x = 1
def change_global():
    global x
    x = 3
change_global()
print x
```

Reboot

列表推倒式

```
print [x*x for x in range(10)]
print [x*x for x in range(10) if x % 3 == 0]
print [(x,y) for x in range(3) for y in range(3)]
```

labmba匿名函数

```
g = lambda x: x*2
print g(3)
```

Reboot

作业1

- 一个list[(1,4),(5,1),(2,3)],根据每个元组中的较大值进行排序
  - 期待结果：[(2,3),(1,4),(5,1)]
  - 要求：用sorted和lambda完成
  - 级别1：用lambda中用max
  - 级别2：lambda中不用max
  - 提示：True乘以4 ==4 Fale乘以2 == 0

```
print True*4
print False*4
```

Reboot

- 用函数，优化log分析的功能
- 实现加减乘除功能的函数
  - 级别1 不支持优先级
  - 级别2 支持优先级，但是没有括号
  - def operate(str):
  - operate('1+2+3-5') == 1

Reboot

微信公众号

Q & A

〈Thank You!〉

Powered By nodePPT v0.9.8-3