一．正则表达式

1. 核心函数和方法

1.match函数

尝试用正则表达式模式从字符串的开头匹配,如果匹配成功,则返回一个匹配对象;否则返回None

1. search函数

在字符串中查找正则表达式模式的第一次出现,如果匹配成功,则返回一个匹配对象;否则返回None

1. group方法

使用match或search匹配成功后,返回的匹配对象可以通过group方法获得匹配内容

1. findall函数

在字符串中查找正则表达式模式的所有(非重复)出现;返回一个匹配对象的列表

1. finditer函数

和findall()函数有相同的功能,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

1. compile函数

对正则表达式模式进行编译,返回一个正则表达式对象不是必须要用这种方式,但是在大量匹配的情况下,可以提升效率

7.split方法

根据正则表达式中的分隔符把字符分割为一个列表,并返回成功匹配的列表

字符串也有类似的方法,但是正则表达式更加灵活

1. sub方法

把字符串中所有匹配正则表达式的地方替换成新的字符串

# import re

#

# re.match('f..','food') #匹配到，返回匹配对象

# print(re.match('f..','seafood')) #匹配不到，返回None

# m = re.search('f..','seafood is food') #从任意地址匹配，match从开头匹配

# print(m.group()) #返回匹配到的字符

# re.findall('f..','seafood is food') #返回所有匹配到的字符串列表，返回有匹配

# #对象构成的迭代器，迭代器的每个对象都有group方法

# for m in re.finditer('f..','seafood is food'):

# print((m.group))

# re.split('-|\.' ,'hello-world-aaa.tar.gz') #切割成单词列表

# re.sub('x','zzg','hi.x .how are you,x?')

# #把最后字符串中的x，替换成zzg

#

#

# patt = re.compile('f..') #如果有大量匹配，先将模式编译会有更好的执行效率

# patt.search('seafoob') #编译都的对象也有search/match等方法

# m = patt.search('seafood')

# m.group()

分析apache访问日志

编写一个apche日志分析脚本

1. 统计每个客户端访问apache服务器的次数

2. 将统计信息通过字典的方式显示出来

3. 分别统计客户端是Firefox和MSIE的访问次数

4. 分别使用函数式编程和面向对象编程的方式实现

# import re

#

# def count\_patt(fname,patt):

# patt\_dict = {}

# cpatt = re.compile(patt)

#

# with open(fname) as fobj:

# for line in fobj:

# m = cpatt.search(line)

# if m:

# key=m.group()

# patt\_dict[key] = patt\_dict.get(key,0)+1

#

# return patt\_dict

#

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

# fname = 'access\_log'

# ip = '^(\d+\.){3}\d+'

# print(count\_patt(fname,ip))

# #awk '{print $1}' access\_log | sort | uniq -c | sort -nr

# br = 'Firefox|MSIE|Chrome'

# print(count\_patt(fname,br))

# shell = 'bash$|nologin$'

# print(count\_patt('/etc/passwd',shell))

# import re

# from collections import Counter

# class CountPatt:

# def \_\_init\_\_(self,fname):

# self.fname = fname

# def count\_patt(self,patt):

# cpatt = re.compile(patt)

# patt\_obj = Counter()

# with open(self.fname) as fobj:

# for line in fobj:

# m = cpatt.search(line)

# if m:

# item = m.group()

# patt\_obj.update([item])

# return patt\_obj

#

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

# fname = 'access\_log'

# ip = '^(\d+\.){3}\d+'

# web\_log = CountPatt(fname)

# a = web\_log.count\_patt(ip)

# print(a)

# print(a.most\_common(3))

创建TCP服务器

• 创建TCP服务器的主要步骤如下:

1. 创建服务器套接字:s = socket.socket()

2. 绑定地址到套接字:s.bind()

3. 启动监听:s.listen()

4. 接受客户连接:s.accept()

5. 与客户端通信:recv()/send()

6. 关闭套接字:s.close()

# import socket

#

# host = '' #表示监听在0.0.0.0

# port = 12345 #端口号，应该>1024

# addr = (host,port)

# s= socket.socket()

# #如果没有以下一行设置，当程序结束后，系统默认保留该套接字60秒，无法立即在运行

# #加上以下设置，程序可以立即重新启动

# s.setsockopt(socket.SOL\_SOCKET,socket.SO\_REUSEADDR,1)

# s.bind(addr) #绑定地址到套接字

# s.listen(1) #启动监听

# while True:

# try:

# cli\_sock,cli\_addr =s.accept()

# #s.accept()接受客户连接,accept必须绑定一个ip和监听端口

# print('Client commected from:',cli\_addr)

# except KeyboardInterrupt:

# break

# while True:

# data = cli\_sock.recv(1024)

# if data.strip() ==b'quit':

# break

# print(data.decode('utf8'))

# try:

# sdata = input('>')

# except KeyboardInterrupt:

# break

# sdata = '%s\r\n' % sdata

# cli\_sock.send(sdata.encode('utf8'))

# cli\_sock.close()

# s.close()

#客户端访问， telnet 127.0.0.1 12345

客户端

import socket

host = '192.168.4.254'

port = 12345

addr = (host,port)

c = socket.socket()

c.connect(addr)

while True:

data = input('>')

sdata = '%s\r\n' % data

c.send(sdata.encode('utf8'))

if data.strip() == 'quit':

break

rdata = c.recv(1024)

print(rdata.decode('utf8'))

c.close()