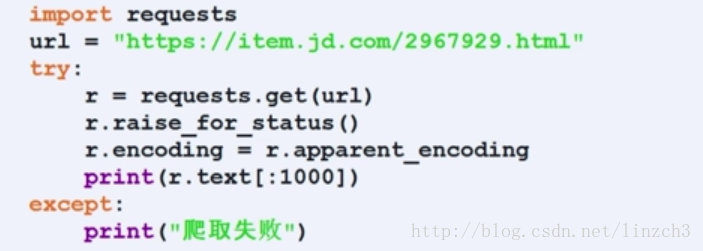
**requests库网络爬虫实战（5个实例）**

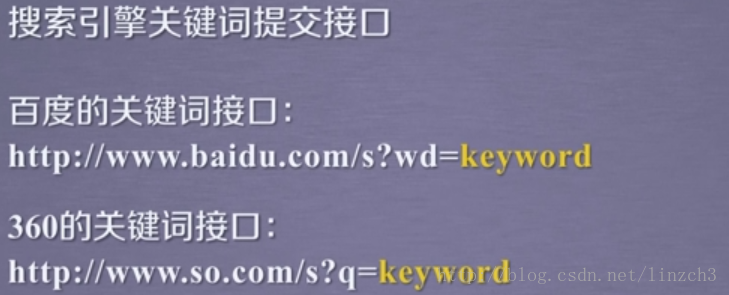
**1.京东商品信息的爬取：**



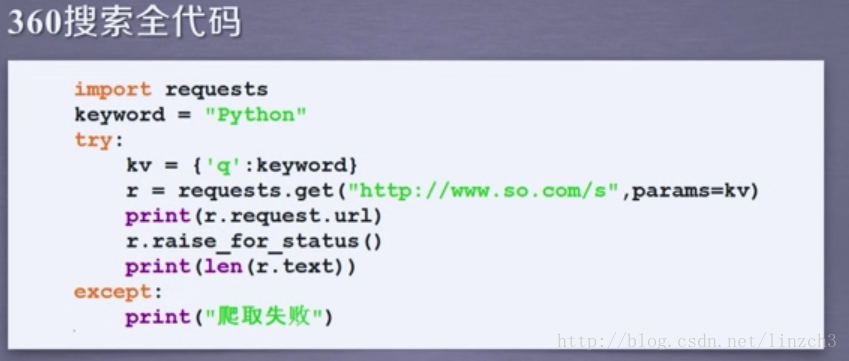
**2.亚马逊商品页面的爬取：**

   
与上一个例子不同的是：这里需要修改headers以让程序模拟浏览器获取数据。   
**注意：这里的调试方法可以参考，可输出r.request.headers查看headers**

**3.百度/360搜索关键词提交**

搜索引擎关键词提交接口如下，可以发现只要替换了keyword这个关键词就可以实现关键词的提交了。   






**4.网络图片的爬取和存储**



例子：   


简单版本：   


详细版本（相比上面的代码，这份代码显得更加稳定）：   
  
**注：这份代码的with…as….语句中的f.close()可以不用加上，该语句会进行相关的文件关闭的处理的。**

**5.IP地址归属地查询**

首先，寻找可查询IP地址的接口，发现在www.ip138.com上有这个接口。通过测试，可发现ip地址查询的接口（下图下方的链接）：类似于上一个例子，只要替换ipaddress这个关键词就可以实现ip地址的查询了。   


下面是测试交互界面和对应代码：   
   
  
注：这里输出respond对象的内容的时候，**都是取r.text的一部分的内容的**，而若直接输出r.text的所有内容，可能会引起对应IDE的失效（或者 输出负担过大）。

而通过这个例子，也可以发现，对于一些可通过用户输入的信息来进行相应的相应的响应的网站，都可以找到其对应的API，从而可用爬虫程序进行自动的信息填入和自动地根据信息来进行相应的处理。比如，对于这一个例子，其API就是：

这里写图片描述

而这也是“以爬虫视角看待网络内容”的表现方式之一。

代码：

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

import requests

def getHTMLText(url):

try:

kv = {'ip': '120.236.174.140'}

r = requests.get(url,params=kv)

r.raise\_for\_status()

r.encoding = r.apparent\_encoding

print(r.url)

return r.text

except:

print("Failed!")

url = "http://www.ip138.com/ips138.asp"

print(getHTMLText(url))