目录

[1． 发get请求](https://www.cnblogs.com/zidonghua/p/7476242.html#_label0)

[2． 发post请求（json）](https://www.cnblogs.com/zidonghua/p/7476242.html#_label1)

[3． 发post请求（data）](https://www.cnblogs.com/zidonghua/p/7476242.html#_label2)

[4． data和json区别](https://www.cnblogs.com/zidonghua/p/7476242.html#_label3)

[5． 发https请求（ssl）](https://www.cnblogs.com/zidonghua/p/7476242.html#_label4)

[6． session关联接口](https://www.cnblogs.com/zidonghua/p/7476242.html#_label5)

[7． json数据处理](https://www.cnblogs.com/zidonghua/p/7476242.html#_label7)

[8. token登录](https://www.cnblogs.com/zidonghua/p/7476242.html#_label10)

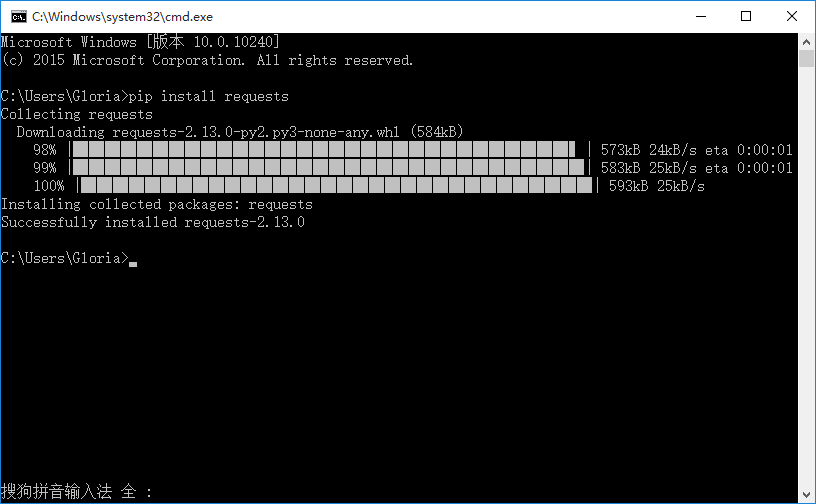
9.携带cookie发送请求

正文

1. 发get请求

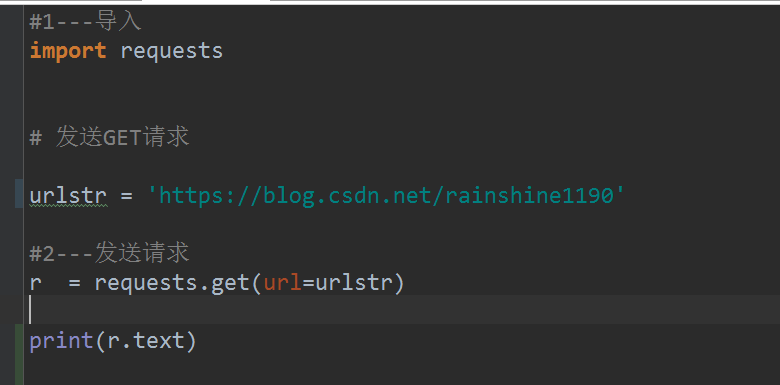
   
一、环境安装

1.用pip安装requests模块  
>>pip install requests



二、get请求

1.导入requests后，用get方法就能直接访问url地址，如：https://blog.csdn.net/rainshine1190，看起来是不是很酷  
1.这里的r也就是response，请求后的返回值，可以调用response里的status\_code方法查看状态码  
3.状态码200只能说明这个接口访问的服务器地址是对的，并不能说明功能OK，一般要查看响应的内容，r.text是返回文本信息

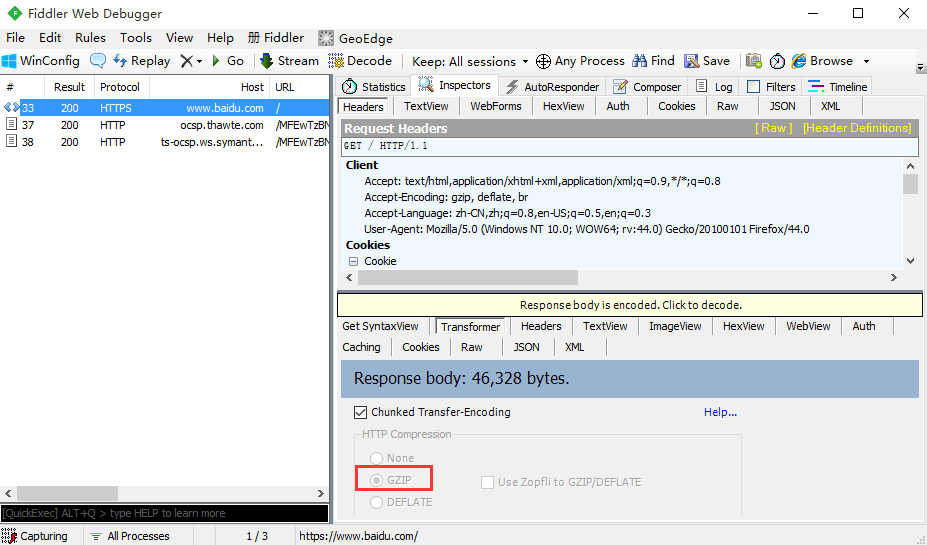


三、params  
1.再发一个带参数的get请求，如玩安卓搜索：Android

地址为：<https://www.wanandroid.com/article/query?k=Android>  
1.请求参数：k=Android，可以以字典的形式传参:{"k": " Android "}  
3.多个参数格式：{"key1": "value1", "key2": "value2", "key3": "value3"}



四、content  
1.百度首页如果用r.text会发现获取到的内容有乱码，因为百度首页响应内容是gzip压缩的（非text文本）



 1.如果是在fiddler工具乱码，是可以点击后解码的，在代码里面可以用r.content这个方法，content会自动解码 gzip 和deflate压缩



五、response  
1.response的返回内容还有其它更多信息

-- r.status\_code #响应状态码

-- r.content #字节方式的响应体，会自动为你解码 gzip 和 deflate 压缩

-- r.headers #以字典对象存储服务器响应头，但是这个字典比较特殊，字典键不区分大小写，若键不存在则返回None

-- r.json() #Requests中内置的JSON解码器，处理后对应Python的dict

-- r.url # 获取url

-- r.encoding # 编码格式

-- r.cookies # 获取response返回的cookie

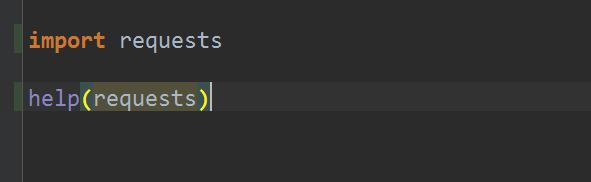
-- r.raw #返回原始响应体

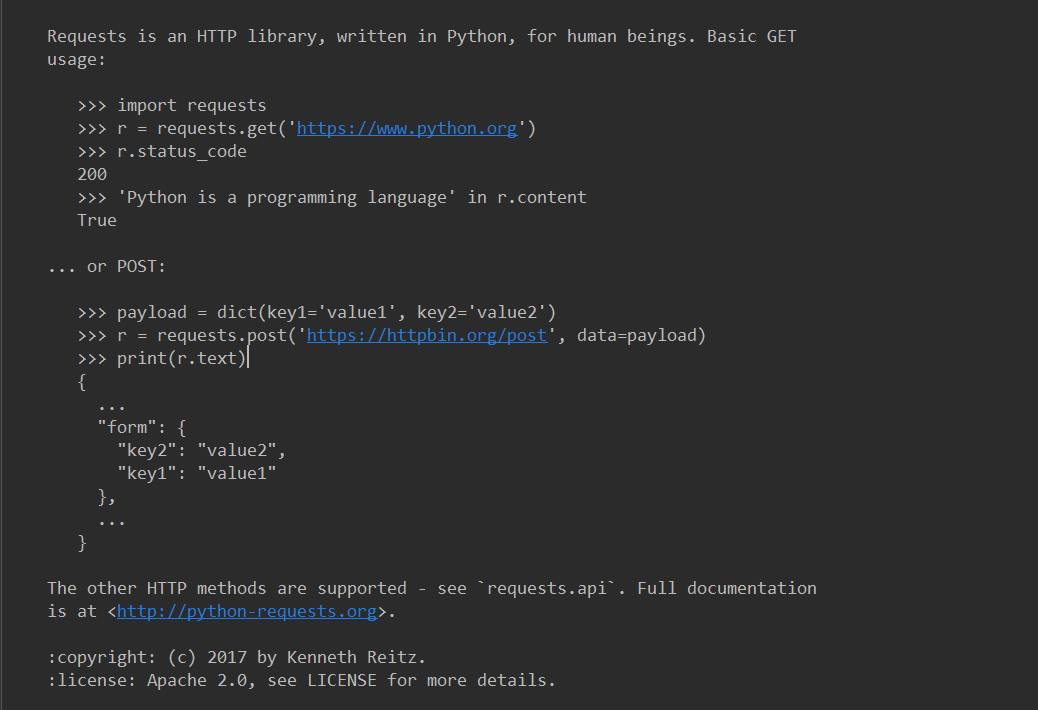
-- r.text #字符串方式的响应体，会自动根据响应头部的字符编码进行解码

-- r.raise\_for\_status() #失败请求(非200响应)抛出异常

2. 发post请求（json）

前言  
发送post的请求参考例子很简单，实际遇到的情况却是很复杂的，首先第一个post请求肯定是登录了，但登录是最难处理的。登录问题解决了，后面都简单了。  
   
一、查看官方文档  
1.学习一个新的模块，直接用help函数就能查看相关注释和案例内容。  
>>import requests  
>>help(requests)



1.查看python发送get和post请求的案例  
   发送post请求(json形式)

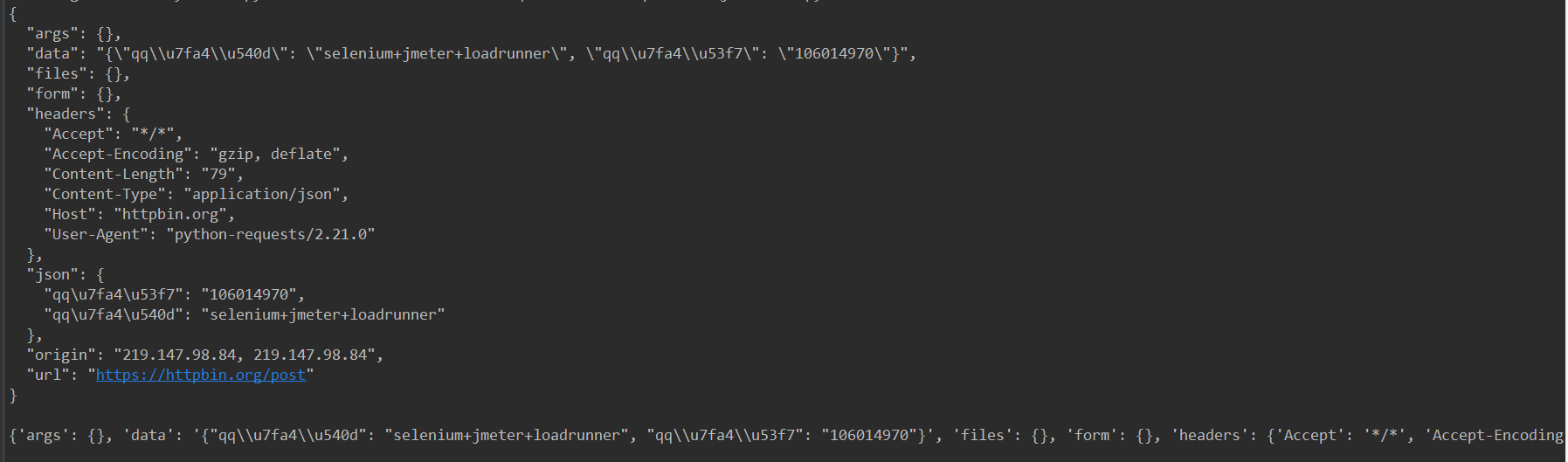
1.post的body是json类型，有两种方法来传递json数据。  
1.第一种：先导入json模块，用dumps方法转化成json格式。

3.第二种：使用json参数默认处理成json格式进行传递。

4.返回结果，传到data里。

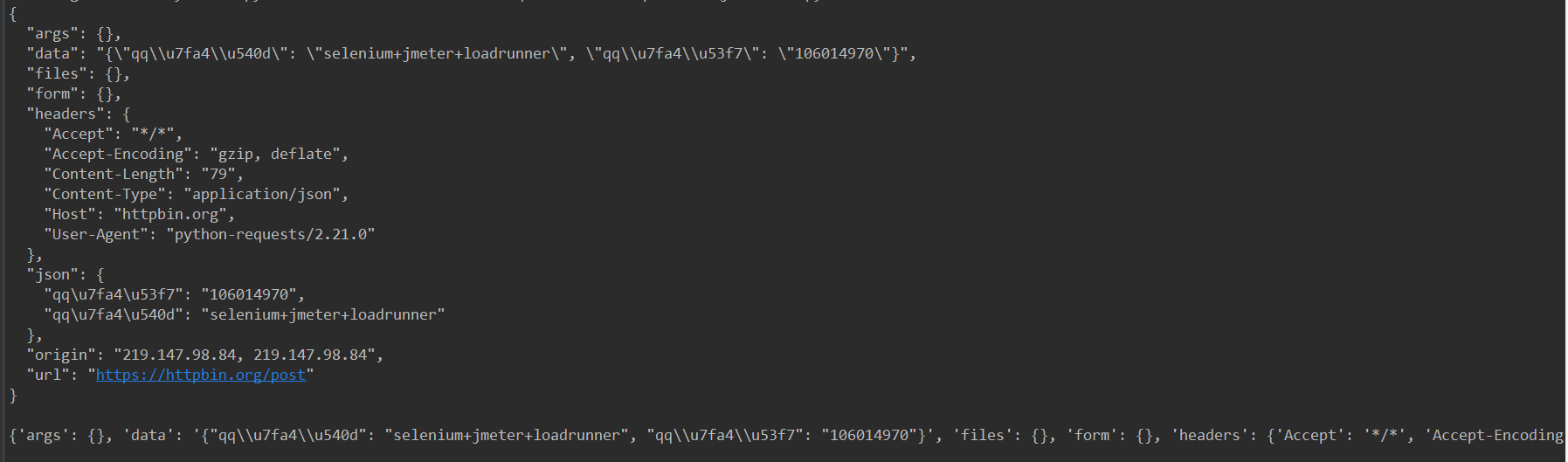
示例1：





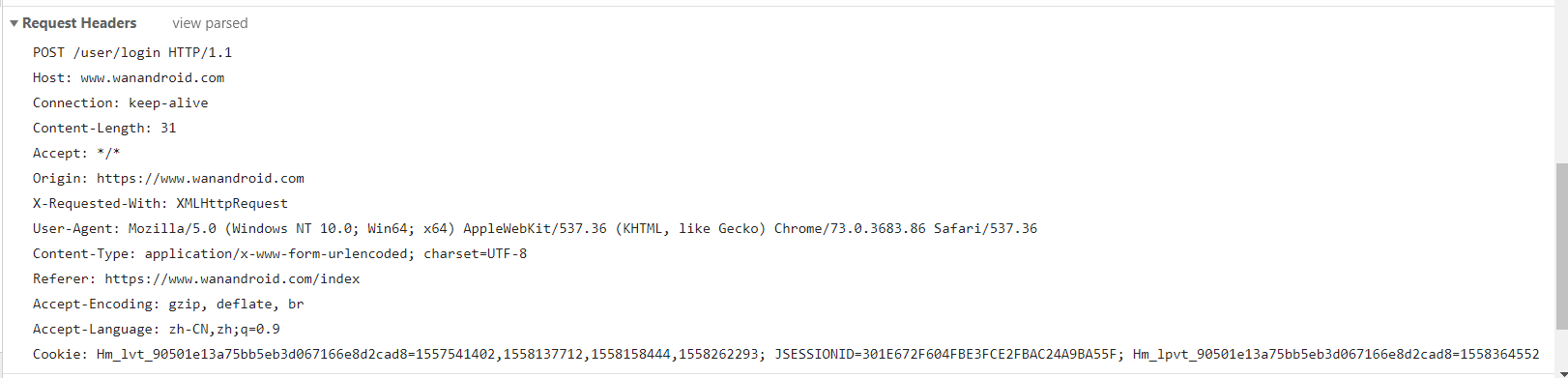
示例2：



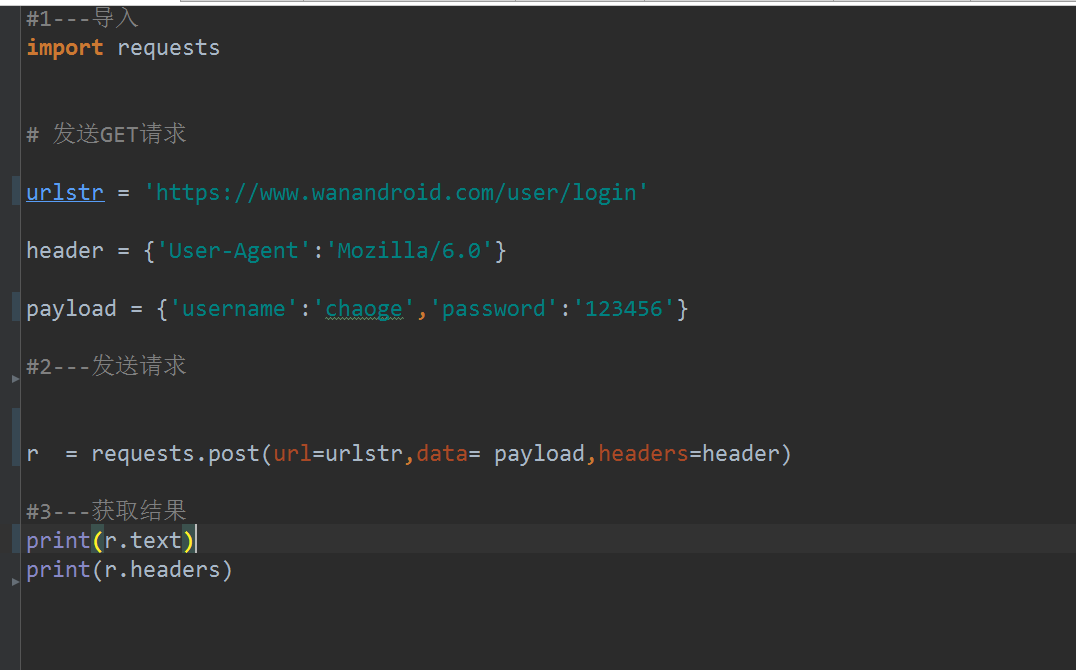


四、headers  
1.以玩Android登录为例，抓包查看请求头，header为Python的字典类型，所以可以支持dict类型的所有操作方式添加header信息，可以直接定义{}，也可以通过dict[key]的方式添加

抓包header如下：



示例代码：



3. 发post请求（data）

前言：  
前面登录博客园的是传json参数，有些登录不是传json的，如jenkins的登录，本篇以jenkins登录为案例，传data参数。

一、发送post请求(表单形式)  
1.用上面给的案例，做个简单修改，发个post请求，payload参数是字典类型，传到如下图的form里（注意post请求参数使用data=），示例中的response返回的是一个json，所以可以通过r.json的方式来查看结果，通过dict引用的方式取值dict[‘key’]

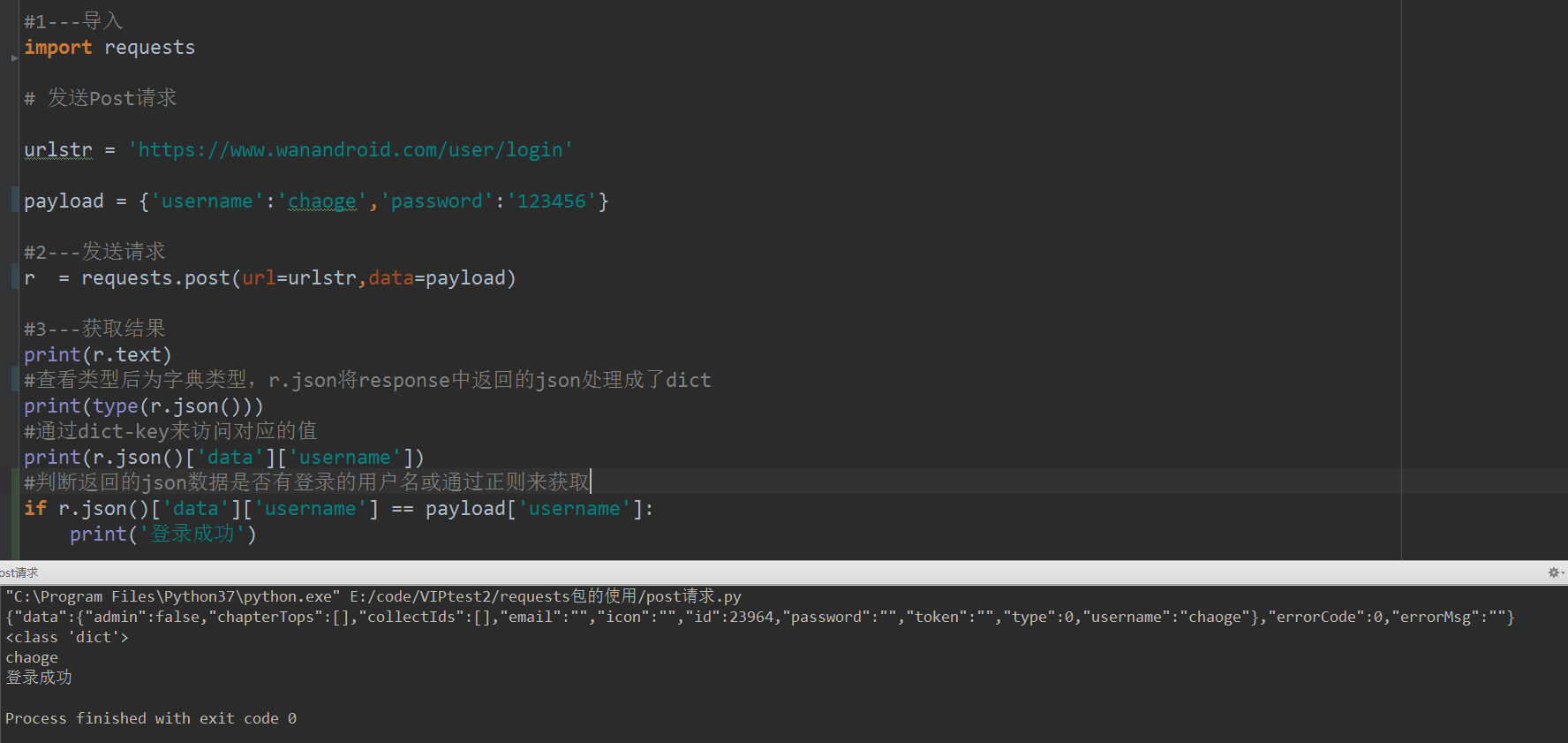


有时候body参数并不是json格式，是key=value格式，也就是前面介绍post请求四种数据类型里面的第二种

  
五、判断登录成功  
1.判断登录成功，可以抓取页面上的关键元素，比如：账号名称，注销按钮

2.通过正则表达式提出这2个关键字

六、参考代码

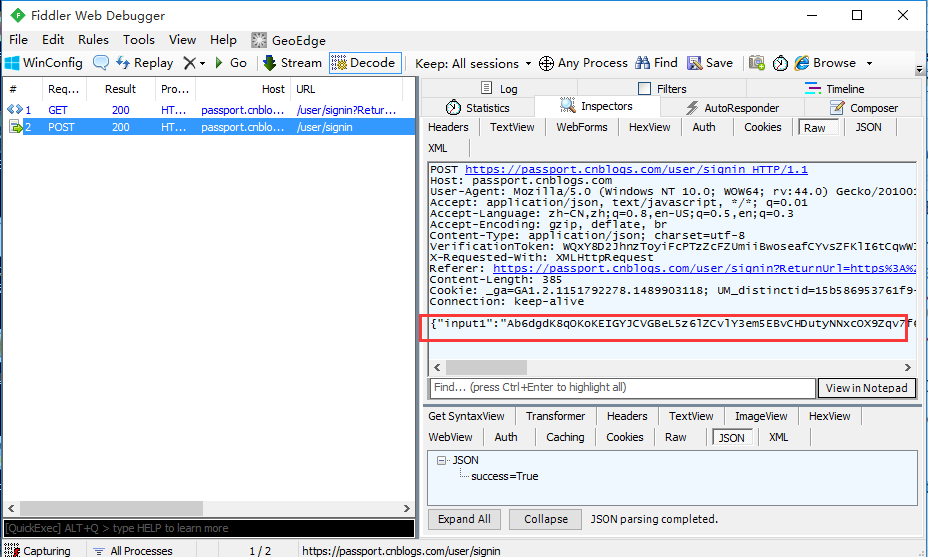


4. data和json区别

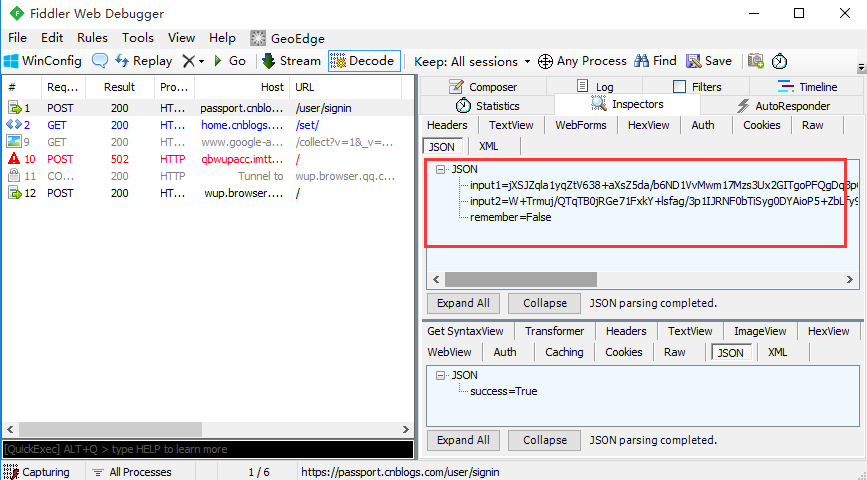
前言  
在发post请求的时候，有时候body部分要传data参数，有时候body部分又要传json参数，那么问题来了：到底什么时候该传json，什么时候该传data?  
   
一、识别json参数  
1.在前面1.8章节讲过，post请求的body通常有四种类型，最常见的就是json格式的了，这个还是很好识别的



1.用抓包工具查看，首先点开Raw去查看body部分，如下图这种，参数最外面是大括号{   }包起来的，这种已经确诊为json格式了。



 3.再一次确认，可以点开Json这一项查看，点开之后可以看到这里的几组参数是json解析后的（记住它的样子）

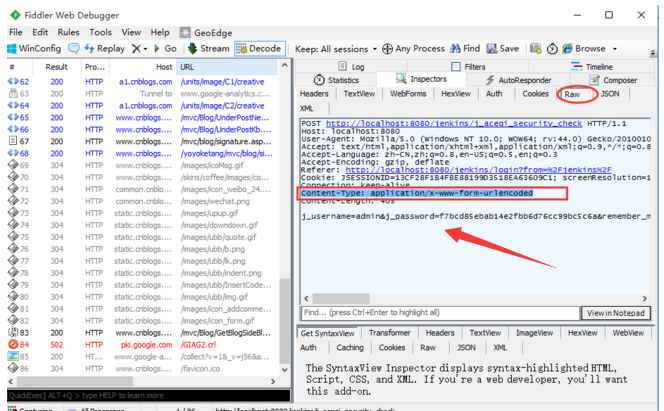


4.这时候，就可以用前面1.2讲的传json参数

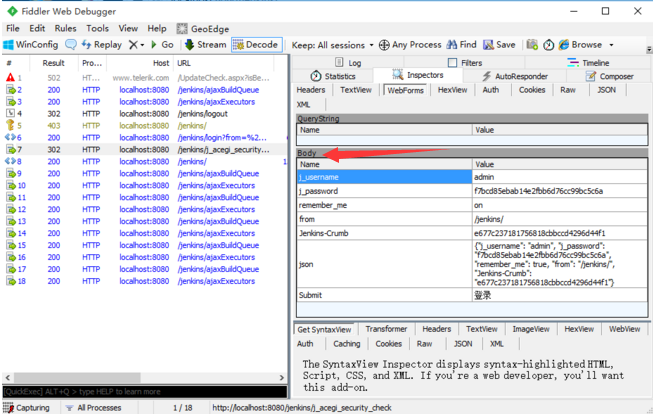


二、识别data参数  
1.data参数也就是这种格式：key1=value1&key2=value1...这种格式很明显没有大括号

点开Raw查看，跟上面的json区别还是很大的



1.因为这个是非json的，所以点开Json这个菜单是不会有解析的数据的，这种数据在WebForms里面查看



3.可以看到这种参数显示在Body部分，左边的Name这项就是key值，右边的Value就是对应的value值，像这种参数转化从python的字典格式就行了



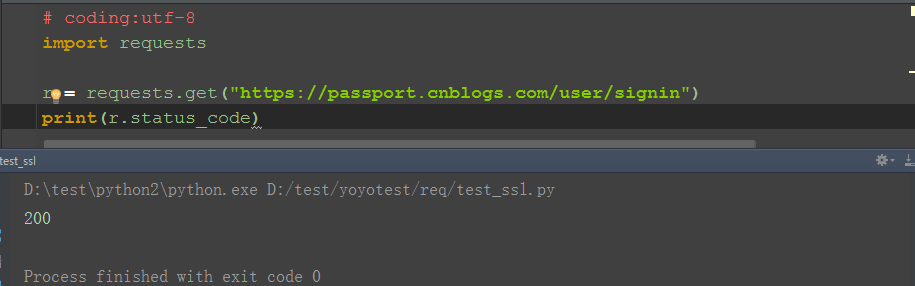
4.这一种发post时候就传data参数就可以了，格式如下：

s = requests.session()

r = s.post(url, headers=headers, data=d) # 这里的d就是上一步的字典格式的参数

5. 发https请求（ssl）

前言  
本来最新的requests库V1.13.0是支持https请求的，但是一般写脚本时候，我们会用抓包工具fiddler，这时候会 报：requests.exceptions.SSLError: [SSL: CERTIFICATE\_VERIFY\_FAILED] certificate verify failed (\_ssl.c:590)  
  
   
一、SSL问题  
1.不启用fiddler，直接发https请求，不会有SSL问题（也就是说不想看到SSL问题，关掉fiddler就行）



1.启动fiddler抓包，会出现这个错误：requests.exceptions.SSLError: [SSL: CERTIFICATE\_VERIFY\_FAILED] certificate verify failed (\_ssl.c:590)



解决方法：

错误提示就是上面这样的。首先我找了很多的资料，有很多人说关闭证书验证（verify=False））可以解决这个问题或者说是在进行GET时,指定SSL证书.

response = requests.get('http://www.baidu.com/', headers = header, verify=False)

但我用以上两种方法都没有完美解决此问题，而且有些还有后续错误比如InsecureRequestWarning

只要安装一下几个requests依赖包就可以解决此问题

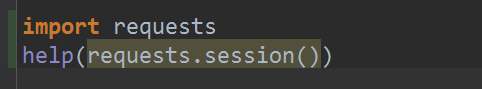
pip install cryptography

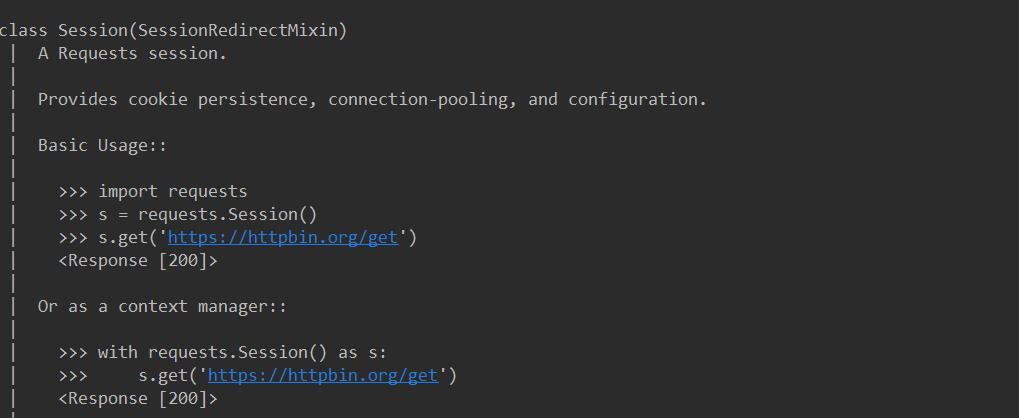
pip install pyOpenSSL

pip install certifi

6. session关联接口

前言  
上一篇模拟登录csdn博客，但这只是第一步，一般登录后，还会有其它的操作，如发帖，评论等，这时候如何保持会话呢？  
   
一、session简介  
1.查看帮助文档



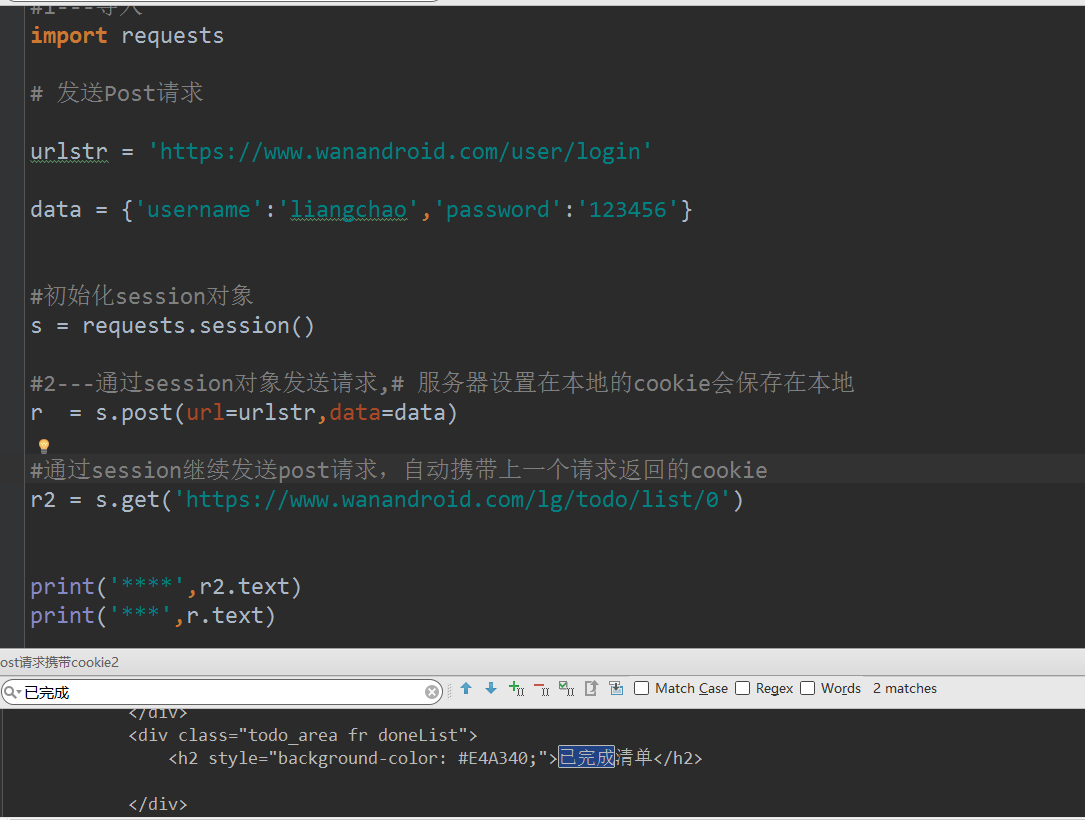


二、使用session登录  
1.使用session登录 s = requests.session()

2.通过session函数自动携带上次请求返回的cookie信息，发送二次post请求

s.post(xxx)

示例代码：



7. json数据处理

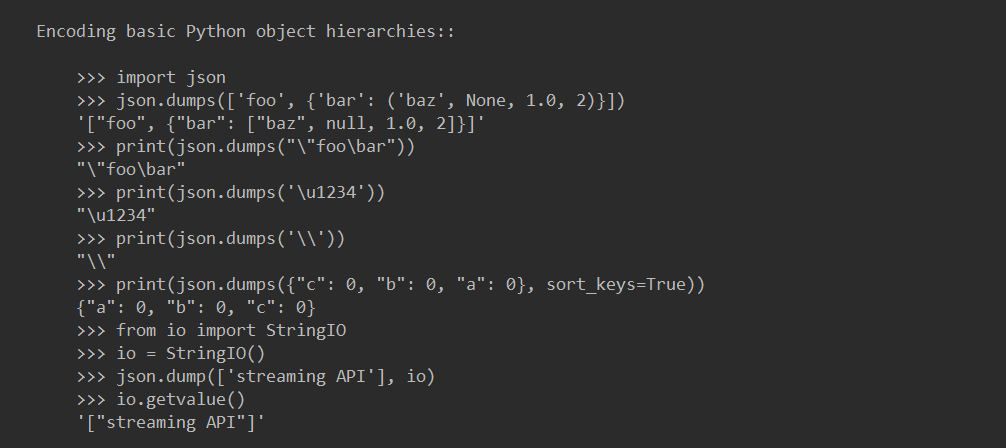
前言  
有些post的请求参数是json格式的，这个前面post请求里面提到过，需要导入json模块处理。  
一般常见的接口返回数据也是json格式的，我们在做处理时候，往往只需要提取其中几个关键的参数就行，这时候就需要json来解析返回的数据了。

前提：import json

json.dumps 将 Python 对象编码成 JSON 字符串---encode编码

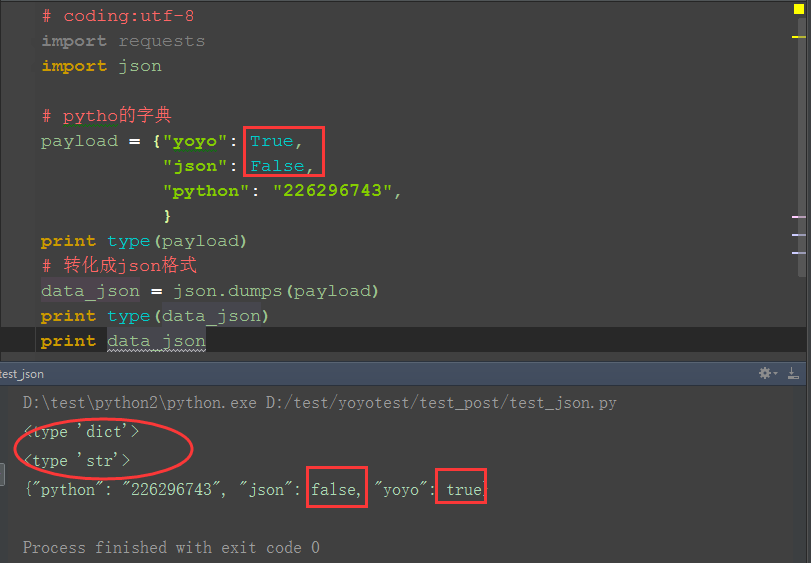
json.loads 将已编码的 JSON 字符串解码为 Python 对象---decode解码

一、json模块简介  
1.Json简介：Json，全名 JavaScript Object Notation，是一种轻量级的数据交换格式,常用于http请求中  
1.可以用help(json),查看对应的源码注释内容：

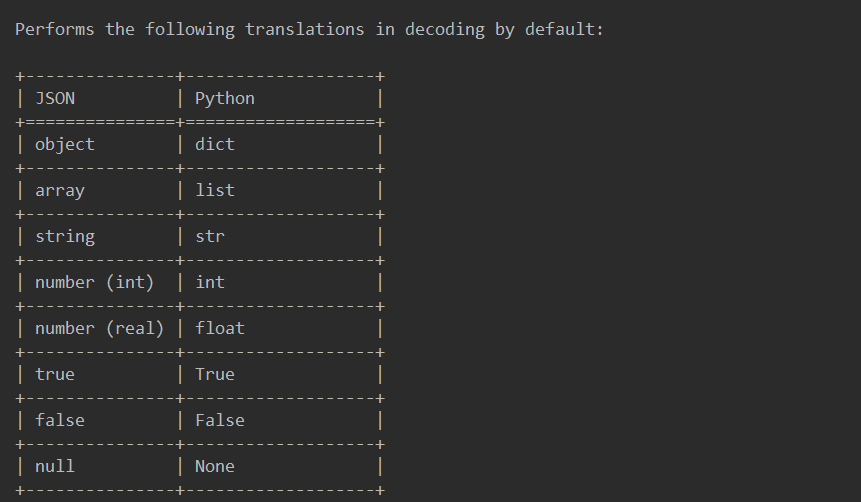


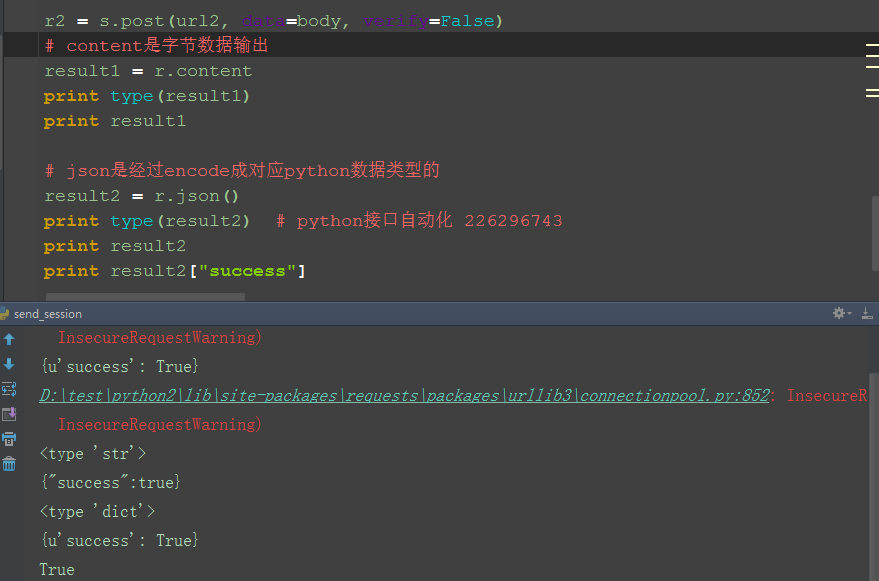
补充json模块，

二、Encode（python->json）  
1.首先说下为什么要encode，python里面bool值是True和False,json里面bool值是true和false,并且区分大小写，这就尴尬了，明明都是bool值。  
在python里面写的代码，传到json里，肯定识别不了，所以需要把python的代码经过encode后成为json可识别的数据类型。  
1.举个简单例子，下图中dict类型经过json.dumps（）后变成str，True变成了true,False变成了fasle

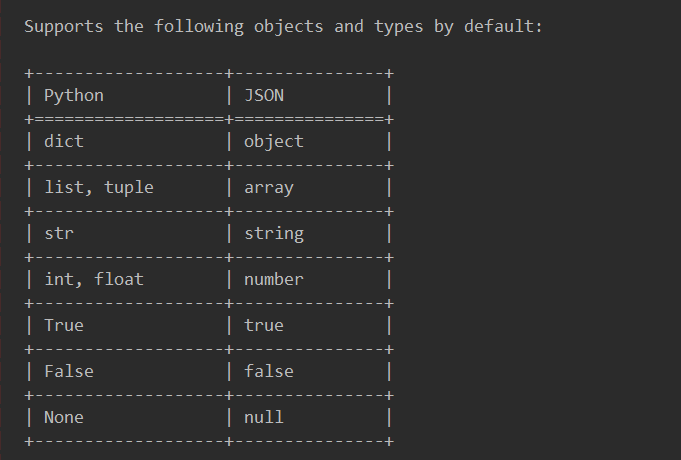


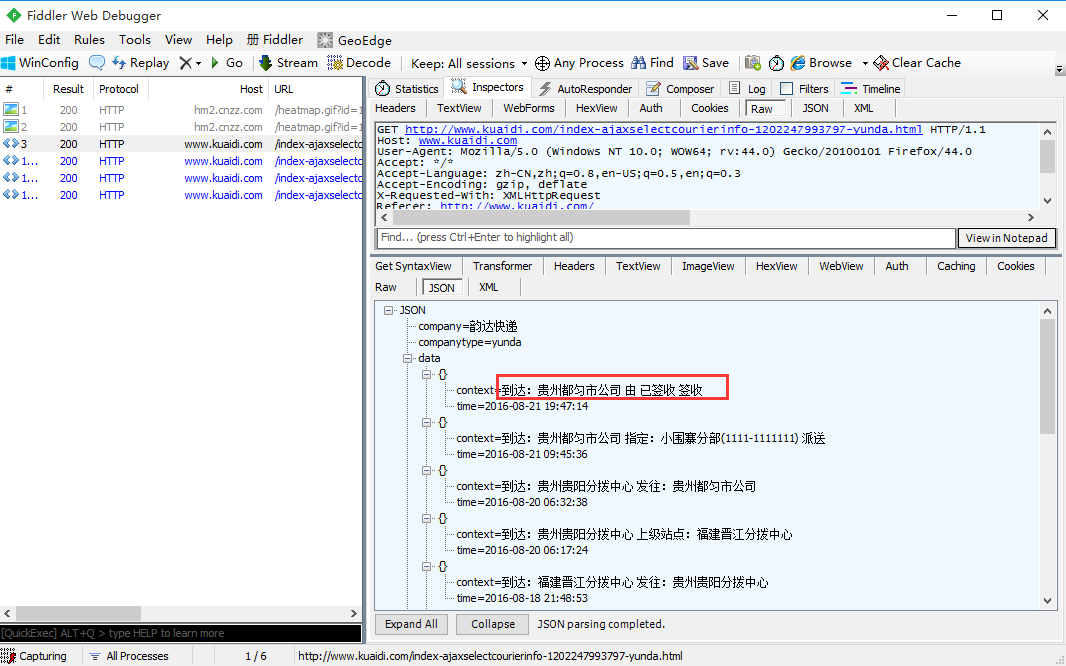
3.以下对应关系表是从json模块的源码里面爬出来的.python的数据类，经过encode成json的数据类型，对应的表如下：

三、decode(json->python)  
1.以第三篇的登录成功结果：{"success":true}为例，我们其实最想知道的是success这个字段返回的是True还是False  
1.如果以content字节输出，返回的是一个字符串：{"success":true}，这样获取后面那个结果就不方便了  
3.如果经过json解码后，返回的就是一个字典：{u'success': True}，这样获取后面那个结果，就用字典的方式去取值：result2["success"]



4.同样json数据转化成python可识别的数据，对应的表关系如下

四、案例分析  
1.比如打开快递网：http://www.kuaidi.com/，搜索某个单号，判断它的状态是不是已签收

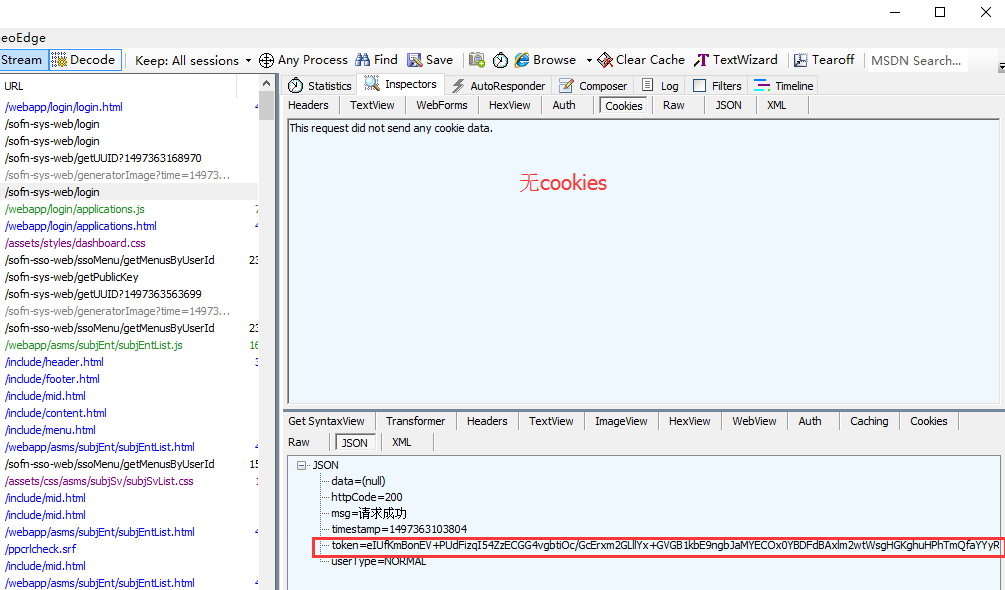


1. 实现代码如下

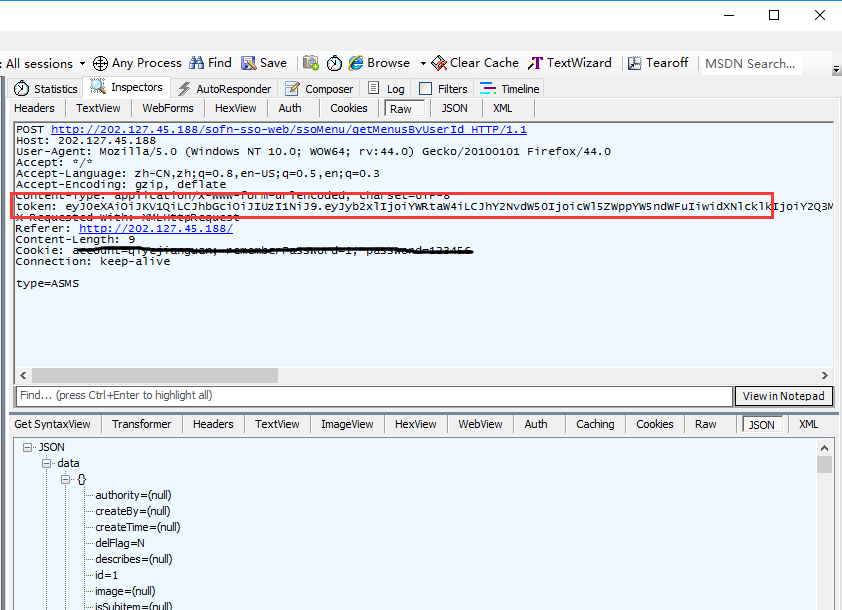


8. token登录

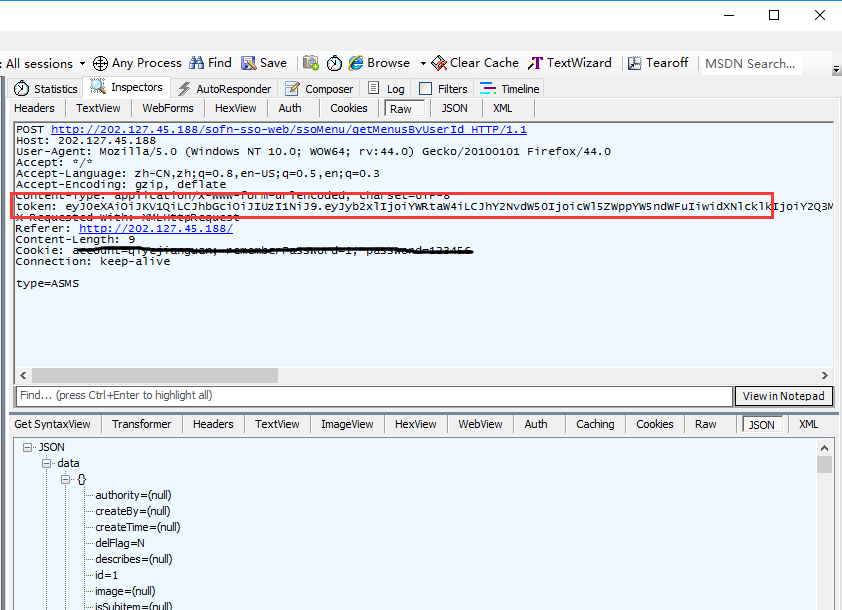
前言  
有些登录不是用cookie来验证的，是用token参数来判断是否登录。  
token传参有两种一种是放在请求头里，本质上是跟cookie是一样的，只是换个单词而已；另外一种是在url请求参数里，这种更直观。  
   
一、登录返回token  
1.如下图的这个登录，无cookies



1.但是登录成功后有返回token



二、请求头带token  
1.登录成功后继续操作其它页面，发现post请求的请求头，都会带token参数



1.这种请求其实比cookie更简单，直接把登录后的token放到头部就行

三、token关联

1.用脚本实现登录，获取token参数，获取后传参到请求头就可以了  
1.如果登录有验证码，前面的脚本登录步骤就省略了，自己手动登录后获取token



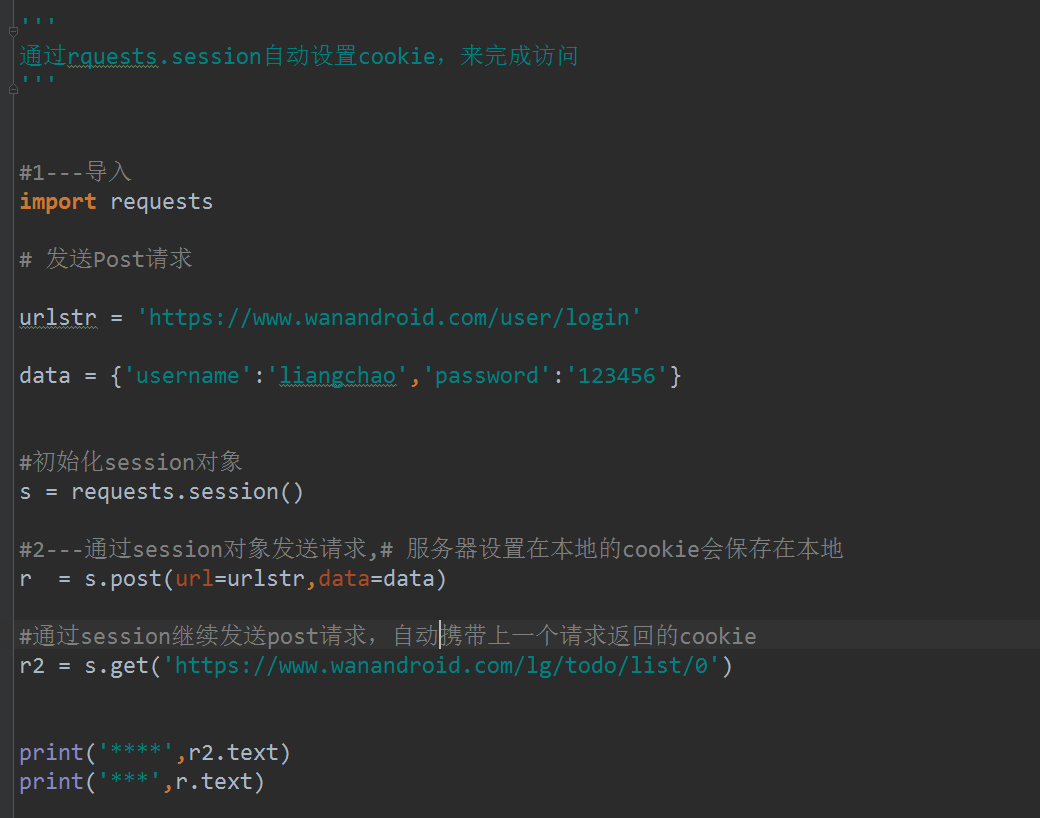
9.携带cookie发送请求

这里整理了4种携带cookie的方法：

1、通过r.cookies手动获取上一请求返回的cookie来设置下次请求的cookie



2、通过rquests.session自动设置cookie，来完成访问



3、通过定制cookie，单独设置cookie来访问目标网址



4、通过定制cookie，放入header来访问目标网址



