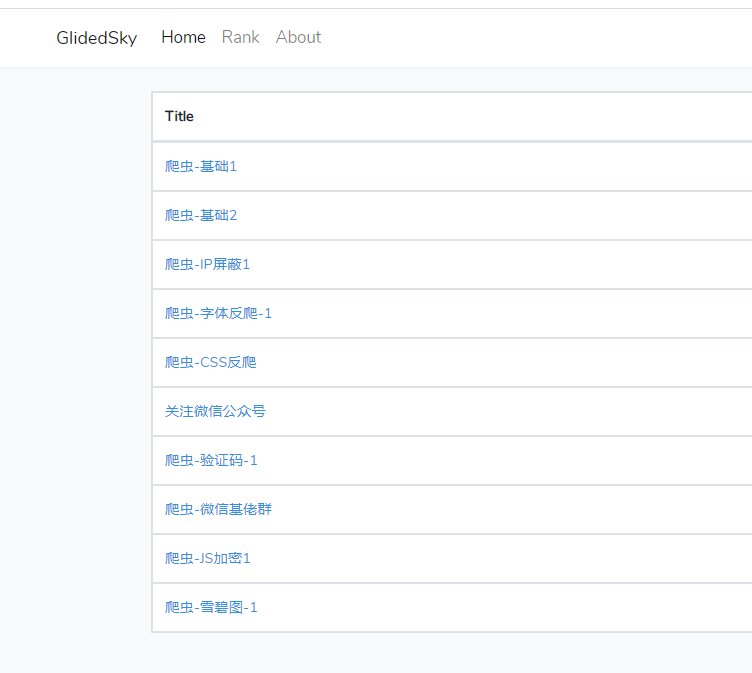
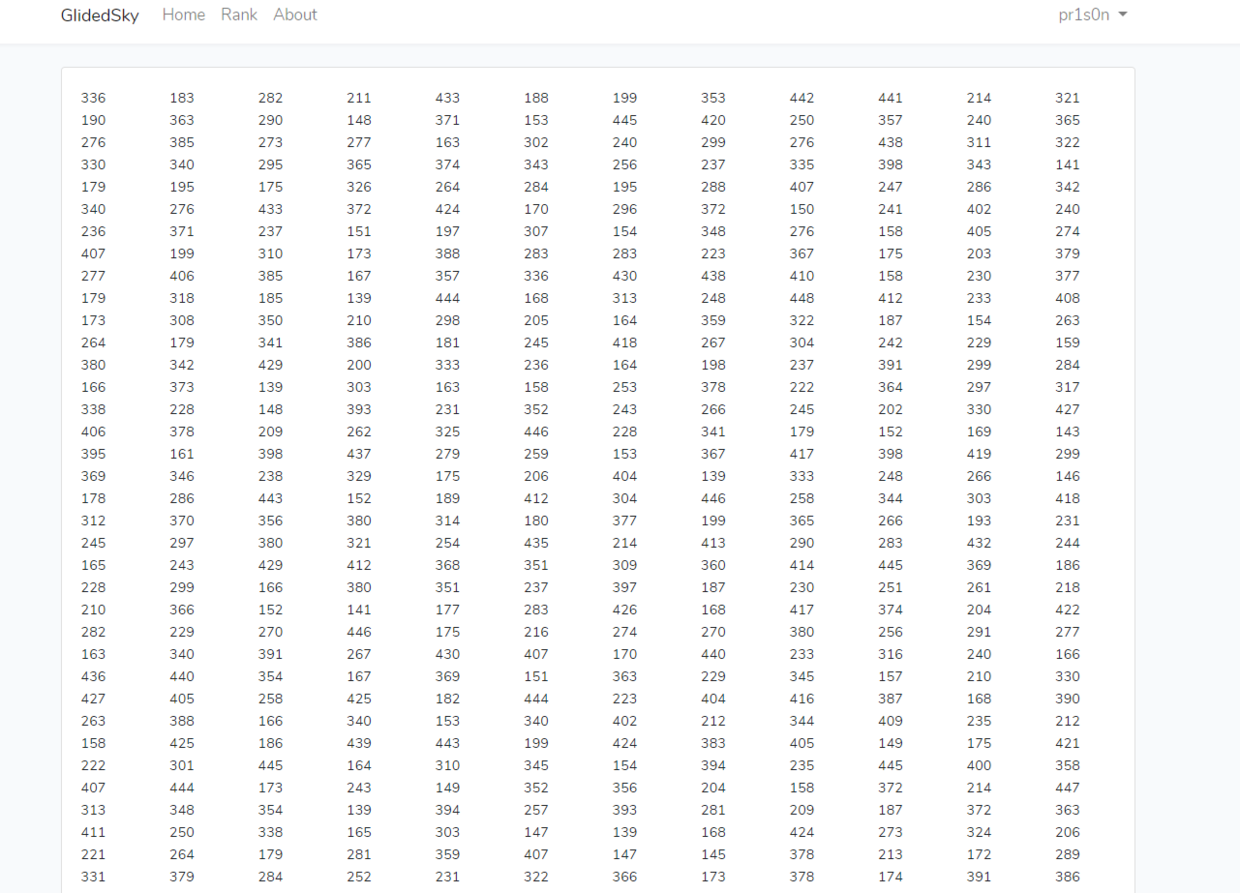
## **前言**

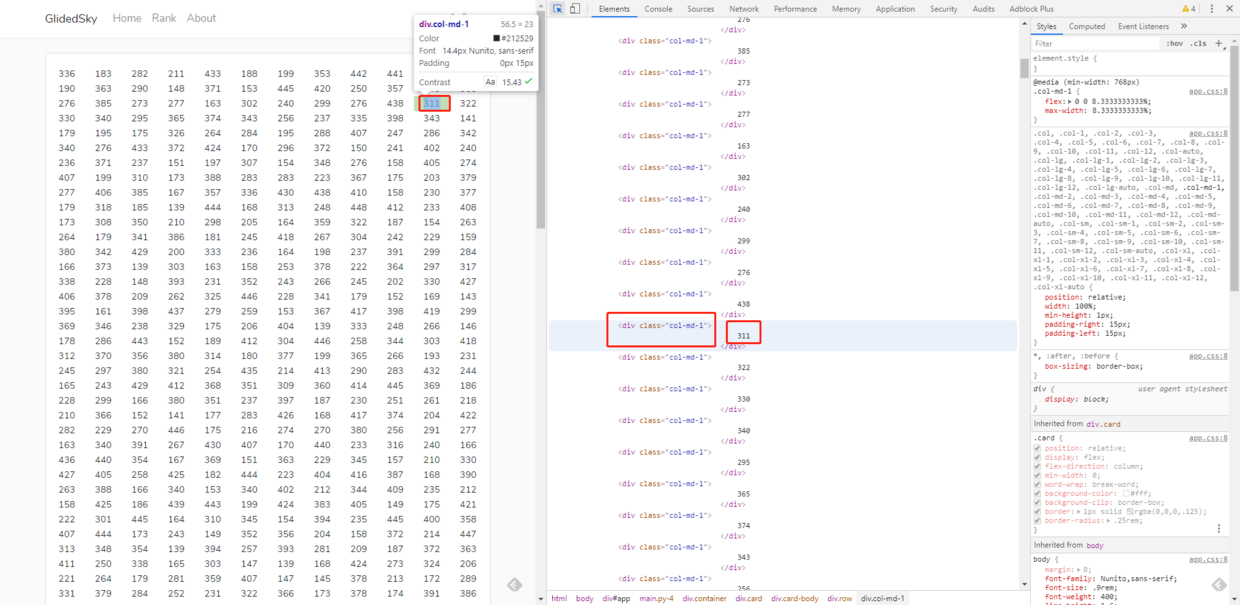
学习Python爬虫技术也是一件需要大量实践的事情，因为并不是所有的网站都对爬虫友好，更多的一种情况是网站为了限制爬虫不得不在最小化影响用户体验的前提下对网站访问做出一定的限制，最常见的就是一些网站的注册和登录页面出现的验证码。  
  
12306网站的验证码在很长一段时间内饱受诟病，最初其复杂程度已经影响到了用户交互体验，但是为什么12306没有选择为了用户体验而放弃验证码？  
因为验证码就像是一个门槛，它主要针对的并不是人，而是可能含有恶意的计算机程序。  
12306网站堪称掌握地表最强开发和维护技术，它每天运行的服务器压力几乎都等同于双十一，高峰时段的操作压力甚至远超双十一十几倍。  
这是个提供刚需服务的网站，稳定的重要性在一定程度上是大于用户体验的，如果没有验证码，那么程序就可以轻而易举的执行登录、注册等操作。  
像我上一篇文章中提到的自动抢票软件，刷新访问的频率越高，对12306服务器的压力就越大。12306也不能分辨屏幕前坐着的到底是不是人，所以只能采取最简单粗暴的人机识别方法:验证码。  


扯远了,不过这也是我们后续会遇到的反爬虫手段之一。

既然是爬虫，当然还是要先确认目标网站，这里推荐一个我认为非常不错的爬虫练习网站：[镀金的天空](http://glidedsky.com/)  


这个网站中从易到难列出了8中反爬虫的挑战，今天要讲的就是其中的基础1：爬取单页数据并计算数据和  


## **分析页面元素**

按下f12调出开发者工具选择我们需要操作的数据，注意这里需要我们记录下数据在html文件中的特征  
  
它是在一个名为col-md-1的div下的数，这个特征为我们之后的数据提取提供了依据。  
可能有的小伙伴不太懂html代码，这里简单提一下html语法的一些特性：

* 标签通常成对出现且分别用于闭合彼此（和我们之前讲的Python中的代码块相似，需要明确出元素开始和结束的位置）
* 标签通常可命名(本例中的class就是一中定义名称的方式，类似的还有id,作用相当于一个考场有50份题目一样的试卷，但是只有写了你名字的那个才算是你的试卷，同时也是为了定位到具体的数据)

html代码中不同的标签有不同的定义关键字

* html中可内嵌css和javaScript代码，也可以从外部引入它们  
  内嵌就是直接写在当前html文件中
* html,css,javascript默认代码可直接查看，因此敏感操作不要写在外部js文件中（把敏感信息写在html中也不行）
* html分head、body

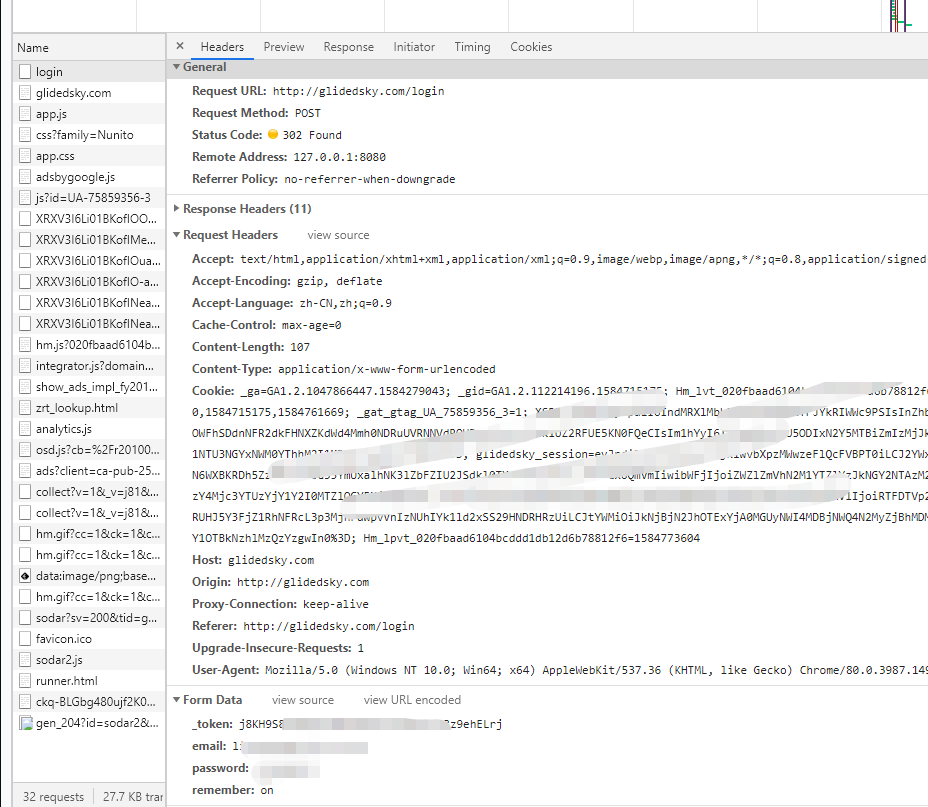
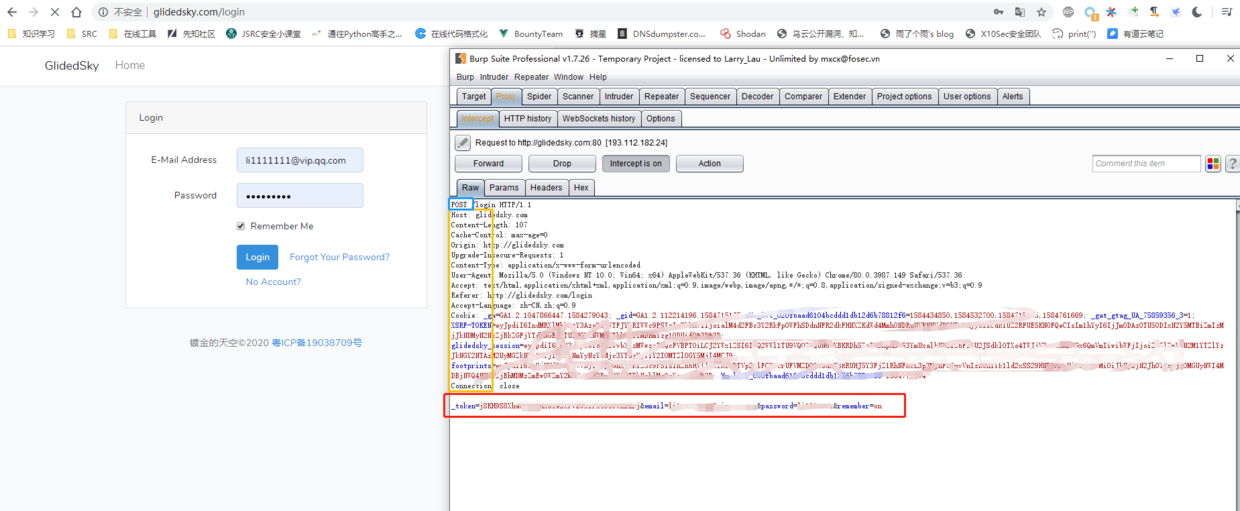
不同标签可能有不同的属性

这些内容还是需要去仔细的了解一下，本系列的文章基本不会再对html基础方向的问题进行叙述。

## **程序实现**

与基础系列不同，为了尊重站长意愿，同时也为了大家能够更好的上手实践，涉及到glidedsky的文章将不会给出全部代码，但是会尽量讲清楚所涉及到的技术和部分代码。

### **Python获取网页文本**

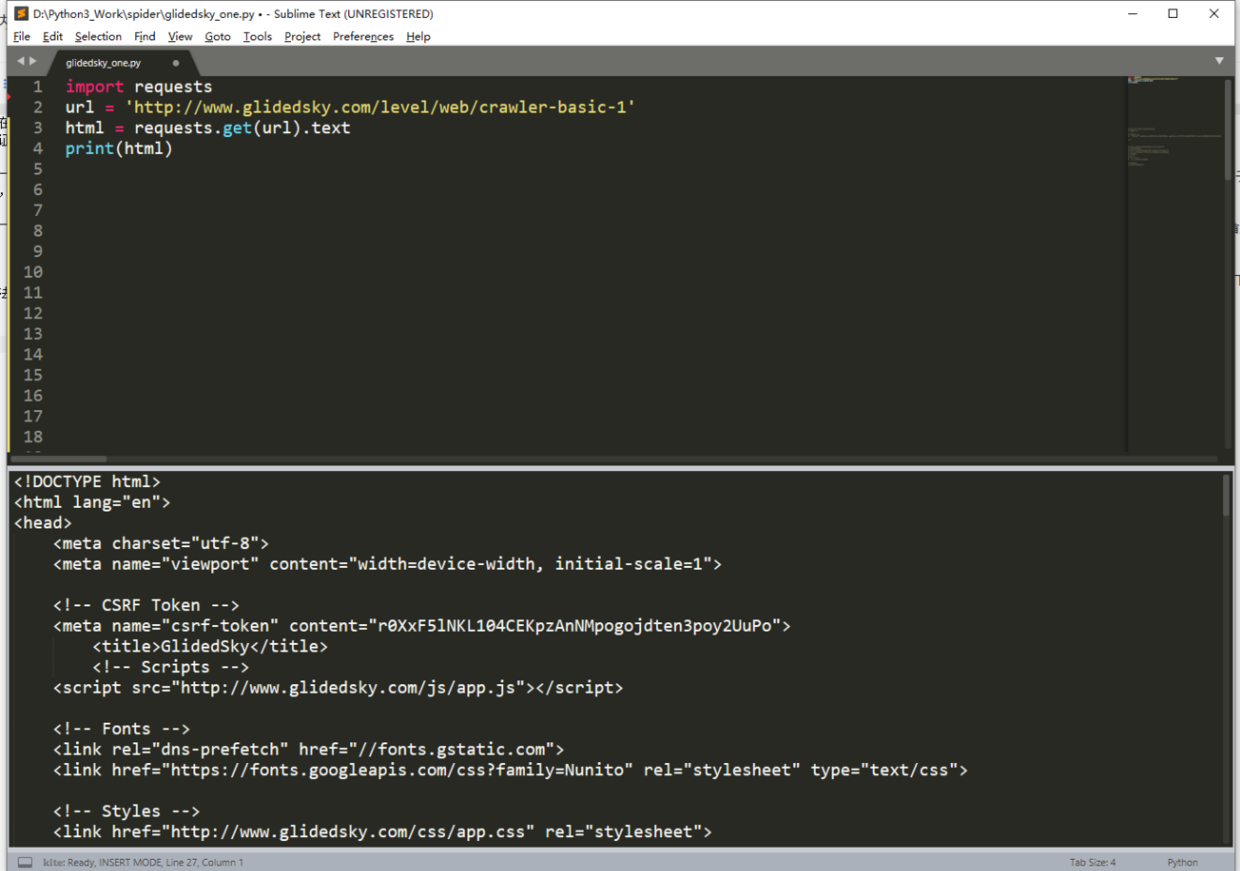
网页访问分GET/POST/DELETE/PUT/OPTION几种不同的类型，最常用的是GET和POST请求。  
get请求的特点是所有参数及数据都显示在URL链接中，主要用于获取数据，对数据本身并不执行操作，所以我们写爬虫的过程中遇到比较多的还是get请求。  
POST请求主要用于向服务器传入数据，POST请求的内容在浏览器页面上是不显示的，需要借助网页抓包工具。  
chrome浏览器的f12开发者工具可以查看到post请求及其内容  
  
使用burpsuite抓包也可以获取到POST数据包的内容  


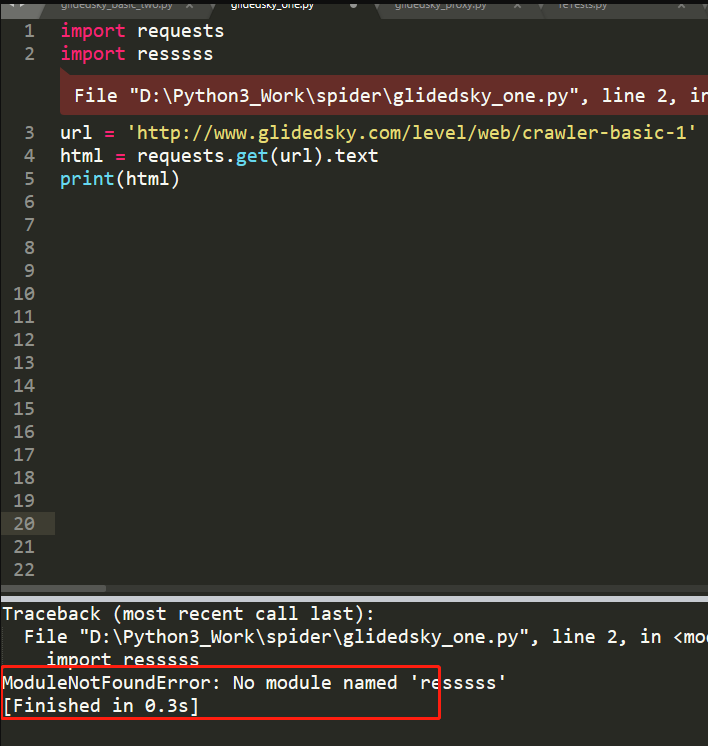
一个请求包通常由请求头和请求主体构成，在这个请求包中，除了POST表单数据之外的数据都是请求头。这里面的数据主要用于和服务器交互，验证数据来源及登录凭证。

首先就是蓝框标注的数据包类型为POST，后一个参数为处理该请求的文件名。  
然后黄框下有几个属性在写爬虫时需要用到，首先是User-Agent代表访问设备信息和浏览器基本信息。

Cookie，这个属性下保存的信息可以相当于一个凭证，任何人拿到你在这个网站的cookie都可以在不知道你的账号密码的情况下进入登录状态。

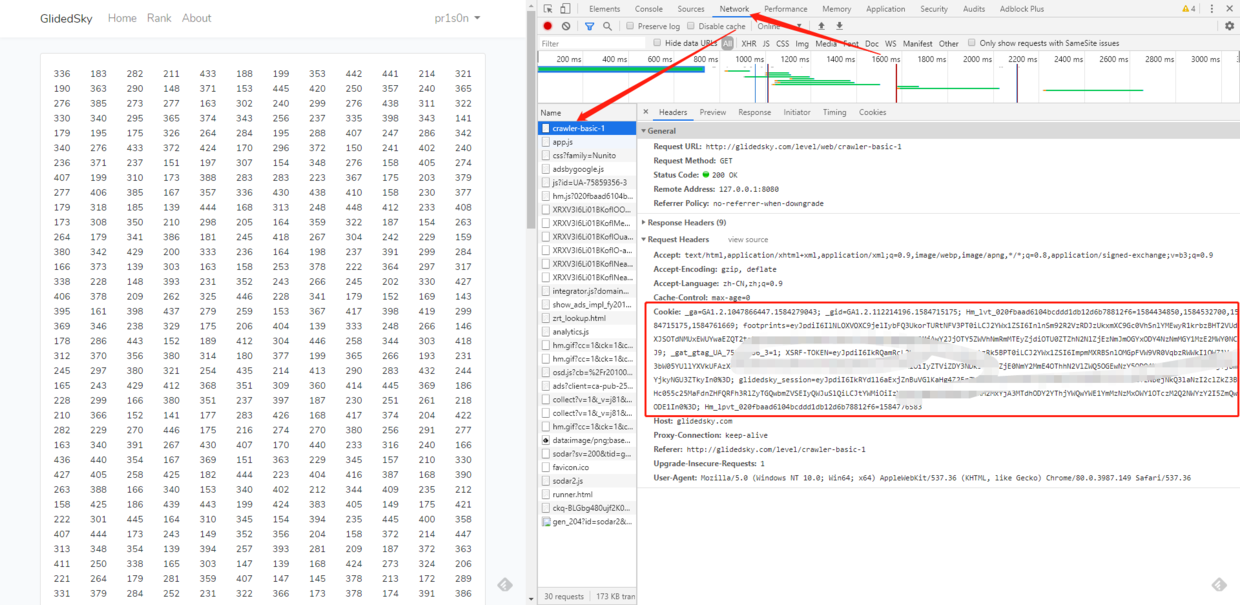
在Python中要实现网页访问功能有很多种方法，我比较喜欢使用requrests这个库。

使用requests中的get方法完成get请求  


注：如果报了这种错误：  


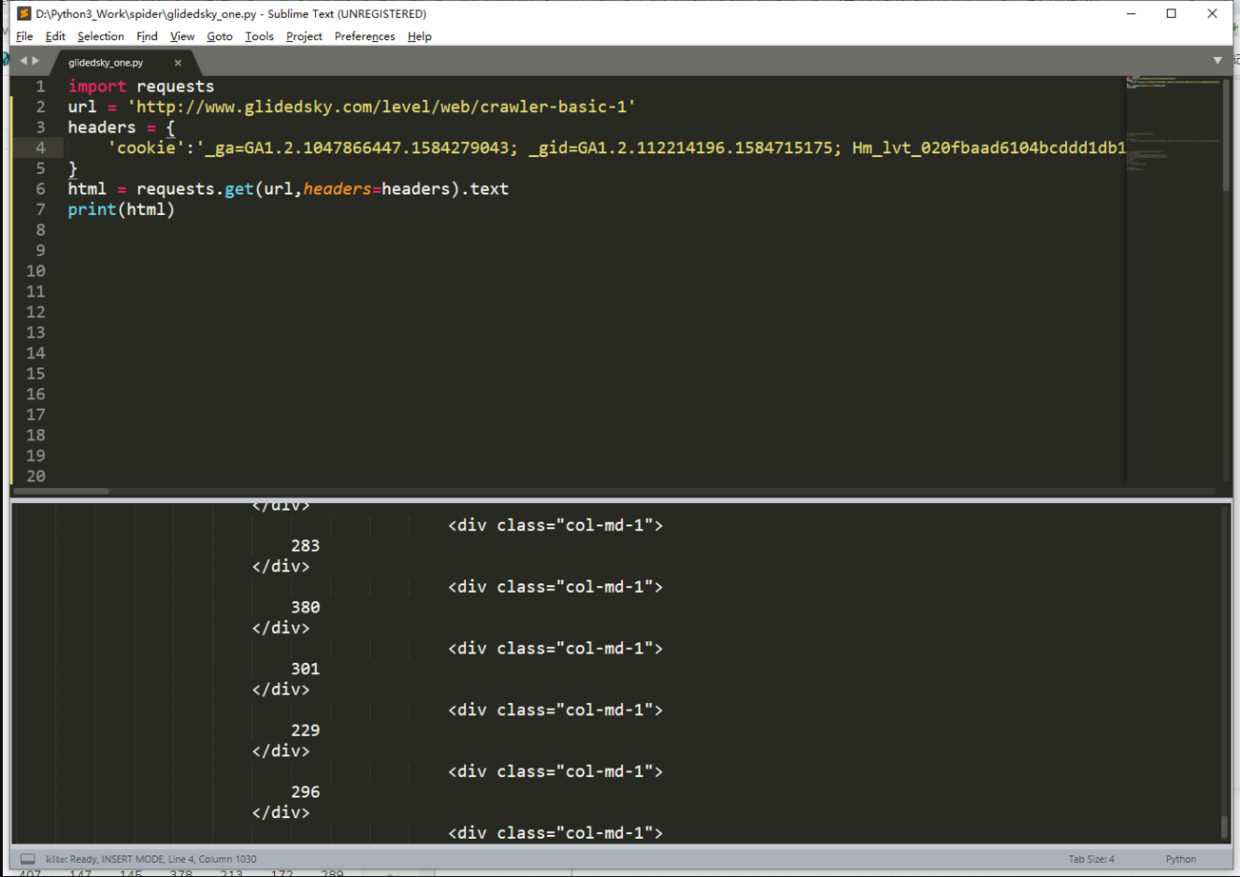
则说明你的电脑中没有安装requests库，在命令行中使用pip install requests命令可进行安装。

可以看到程序成功返回了网页源码，但是仔细看会发现使用Python获取到的源码中并没有包含我们想要的数据。

这是因为该网站做了限制，只有登录后在可以看到数据，所以在请求时我们必须带上cookie才可以。  


按下f12，点击network网络选项，找到主要请求的网页，将cookie后的内容全部复制下来。

requests的get()方法允许我们传入一个header参数



成功获取到目标数据

### **处理获取到的文本数据**

接下来先理下思路：  
目的：提取所有数字并求和  
现已完成：获取到全部文本  
接下来要做的：

1. 提取所有数值
2. 将这些数值累加

确实就是这么朴实无华，想要从这么长的文本中准确获取到数值，我们需要用到两个库：BeautifulSoup和re

BeautifulSoup属于外部库，需要在命令行中使用pip install BeautifulSoup4进行安装。

引入方法：

from bs4 import BeautifulSoup

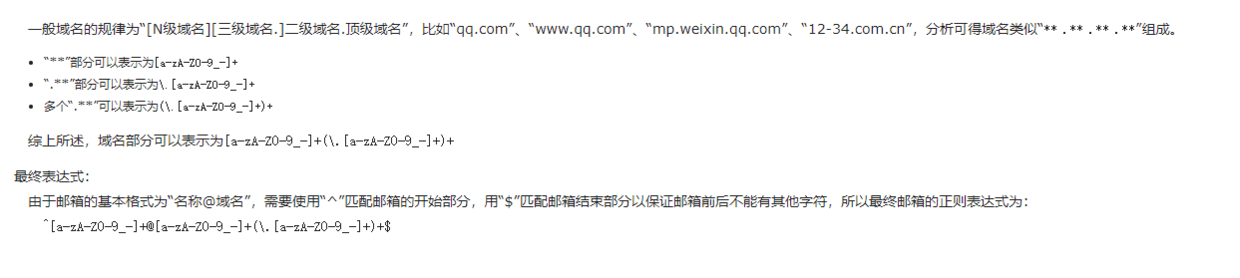
re为正则表达式支持库，Python自带，但是同样需要使用import引入

Beautiful Soup 是一个可以从HTML或XML文件中提取数据的Python库.它能够通过你喜欢的转换器实现惯用的文档导航,查找,修改文档的方式。

这是官方的解释，通俗点说就是它可以帮助你更方便的提取数据。

正则表达式是用来匹配具有某种特征数据的模式，它定义了一些字符代表不同的数据。

虽然普通人真正见到正则表达式本体的机会不多，但是肯定都用过。

例如在注册一个网站时，会要求你填写邮箱和手机号，程序就是通过正则表达式首先判断你输入的邮箱和手机号格式是否正确。  


正则表达式语法同样比较复杂，但是非常有效，非常建议学习。

推荐看这篇文章了解正则表达式： [三十分钟入门正则表达式](http://www.cnblogs.com/deerchao/archive/2006/08/24/zhengzhe30fengzhongjiaocheng.html)

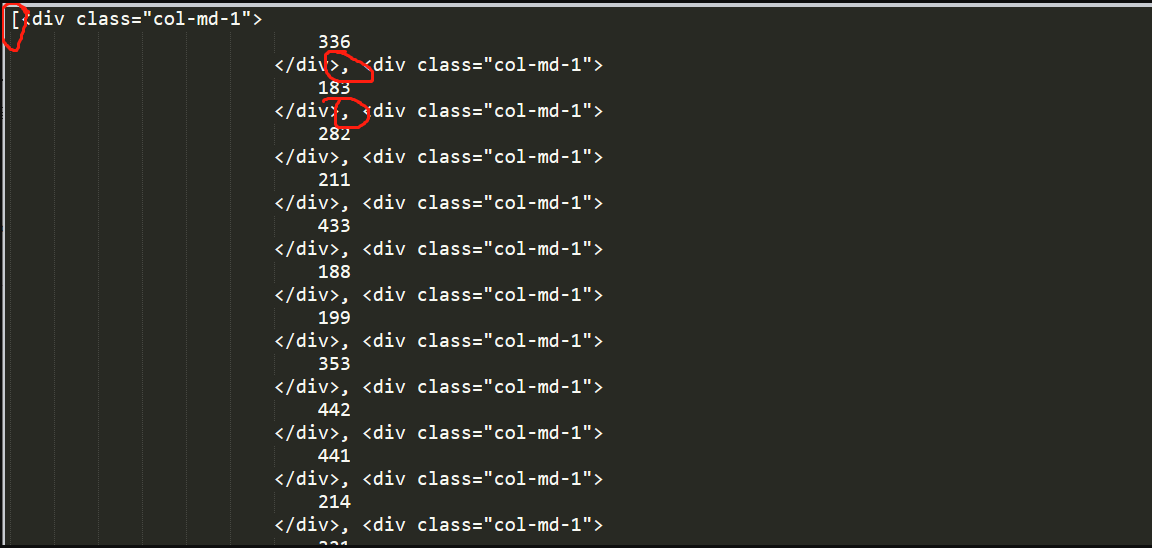
理解这两个库的作用之后我们就可以开始使用它们了。

首先使用BeautifulSoup对获取到的网页内容进行处理

soup = BeautifulSoup(html,'html.parser')

在处理后的数据中提取出所有class=col-md-1的div的内容

nums = soup.find\_all('div',class\_='col-md-1')

此是我们的数据变为：  


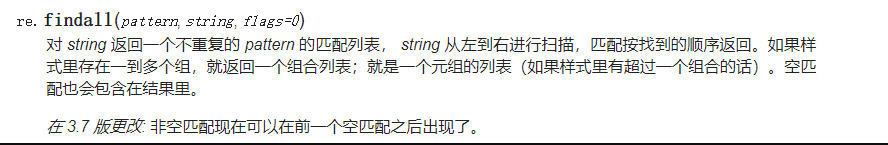
注意这里我圈出的细节，这说明经过处理后的数据存储到了一个列表中。

接下来就是需要把除了数字之外的字符全部剔除

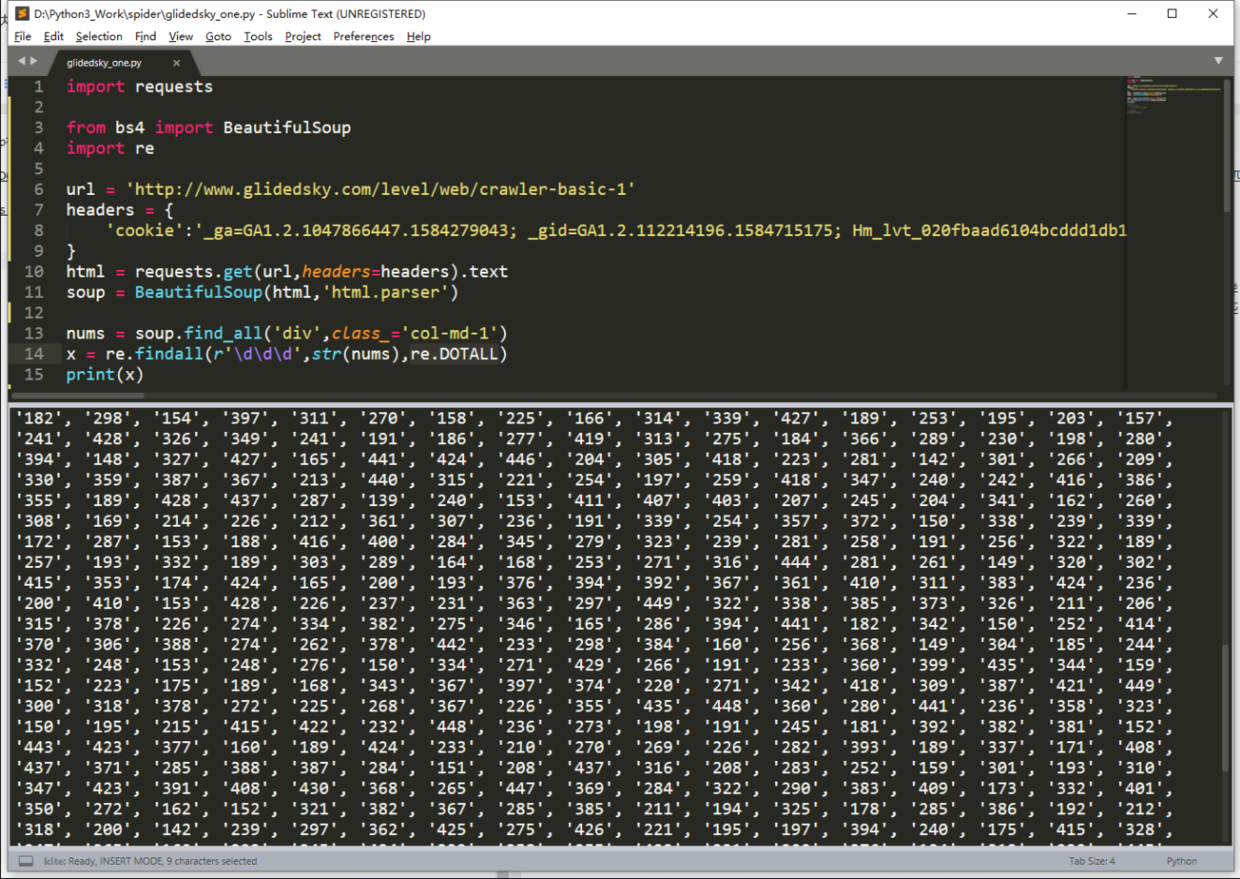
x = re.findall(r'\d\d\d',str(nums),re.DOTALL)

因为正则表达式是针对字符串的，所以我们需要用str方法将刚获取到的数据转化为字符串类型，findall方法的第一个参数即是匹配三位数字的正则表达式，re.DOTALL表示匹配所有字符，包括换行符等特殊符号。

re模块中的Findall方法，和BeautifulSoup模块的find\_all方法完全不一样。



Python re模块的[官方文档](http://docs.python.org/zh-cn/3/library/re.html)中详细罗列了各个方法的使用方法和一些简单的正则表达式知识。

至此，我们就已经将所有的数值匹配出来了  


接下来就可以使用for循环通过下标对这个列表进行遍历，然后将数据累加求和，这部分的代码不再给出。

多尝试，三行代码就能解决。

另外，其实BeautifulSoup和re任意一个模块都可以完成数据筛选工作，我没用的原因是因为我懒。

最后给出最终运行结果

