爬取速度过快的时候会被限制，要注意

# 反爬机制1：Referer

图片网站有反爬机制，是识别爬虫的"Refer”字段，这个字段是用来判断请求的来源

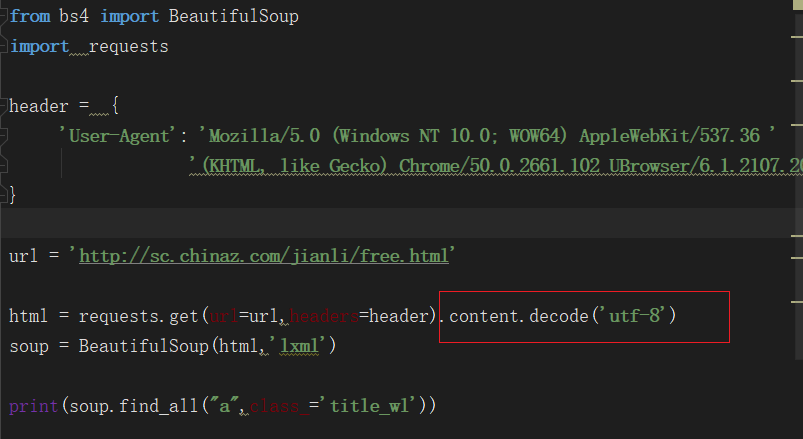
也就是通过headr来判断的，所以我们要模拟正常从主页进入，要在hearder中加入"Referer"键，值为主页网址



# coding=utf-8  
  
import requests  
from bs4 import BeautifulSoup  
import pickle  
  
url = 'http://www.mzitu.com/26685'  
header = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 '  
 '(KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36',  
 'Referer':"https://www.mzitu.com/26685"  
}  
# 反爬机制1：Refer  
# 图片网站有反爬机制，是识别爬虫的"Refer”字段，这个字段是用来判断请求的来源  
# 也就是通过headr来判断的，所以我们要模拟正常从主页进入，要在hearder中加入"Referer"键，值为主页网址  
#  
  
  
html = requests.get(url, headers=header)  
soup = BeautifulSoup(html.text, 'html.parser')  
# print(html.text)  
  
# # 最大页数在span标签中的第10个  
pic\_max = soup.find\_all('span')[9].text  
# print(pic\_max)  
title = soup.find('h2',class\_='main-title').text  
  
# # 输出每个图片页面的地址  
for i in range(1, int(pic\_max) + 1):  
 href = url + '/' + str(i)  
 html = requests.get(href, headers=header)  
 mess = BeautifulSoup(html.text, "html.parser")  
 #虽然html.text也可以显示源码，但是BeautifulSoup(html.text，"html.parser")更快，文档容错能力强  
  
 pic\_url = mess.find('img',alt = title) #获取img标签，其中包含图片地址  
  
 html = requests.get(pic\_url['src'],headers = header)  
 file\_name = pic\_url['src'].split(r'/')[-1]  
 # print(html.content)  
 f = open(file\_name, 'wb')  
 f.write(html.content)  
 f.close()

详解破解referer反爬虫网址：<https://blog.csdn.net/python_neophyte/article/details/82562330>

# 爬取字符串显示乱码问题



看源码的编码方式是什么如果是utf-8的话请求的时候就加上content(获取源码的文本内容)在加上解码的decode(‘utf-8’)就可以在用BeautifulBsoup进行解析操作了

# 爬取图片或者请求其他链接的时候使用了懒加载

**为的是减少服务器负荷**



比如站长素材中，可视化区域内的img标签的属性为



这个src我们对其进行requests请求就可以了

但是如果不在可视化范围内也就是浏览器窗口显示范围内的话



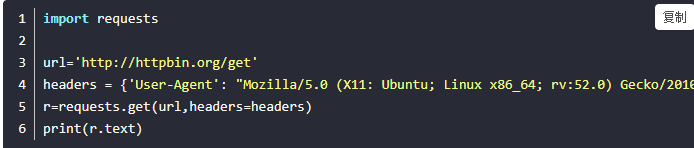
就会发现其并没有src属性只有src2属性，所以我们请求src就会返回空只能去请求src2才可以

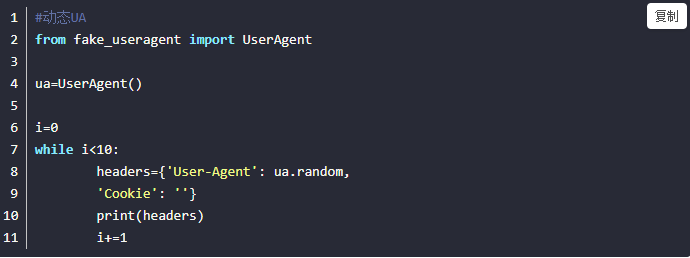
# UA伪装

**防止同一个UA过多访问一个网站**

设置动态UA

‘User-Agent’:当中写入多个头部模拟浏览器

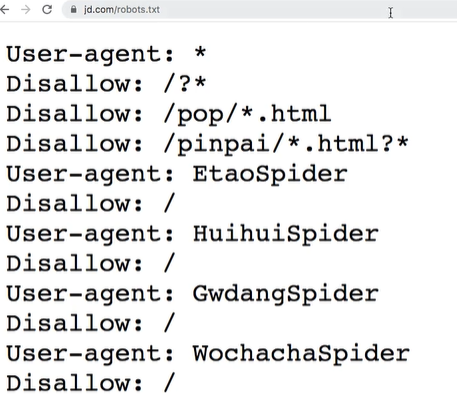




每次爬取的时候随机选择UA模拟浏览器进行爬取

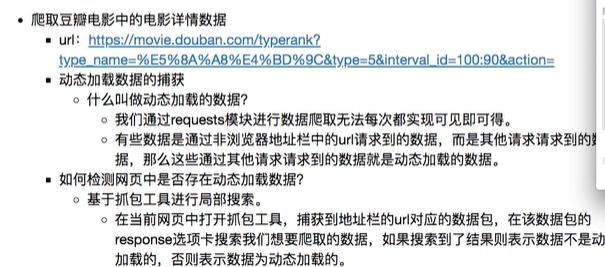
# Robots协议

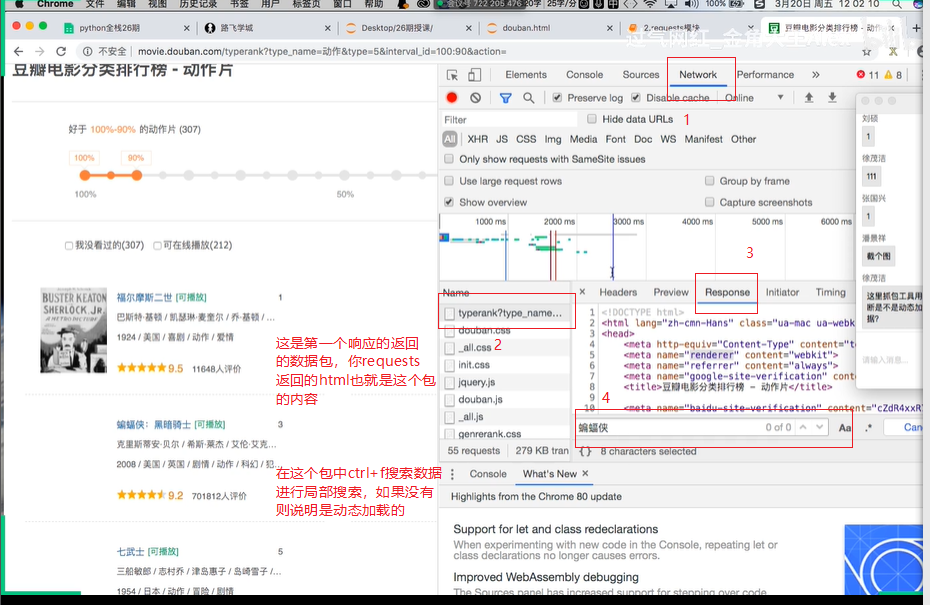
在域名后面加上robots.txt



指的是一个纯文本的协议，协议中规定了该网站中那些数据可以被那些爬虫爬取，哪些不可以被爬取。额 就是一纸书文，防君子不妨小人，而且大都写的是都不可爬，不管它(不爬你养我？)受法律包含的，如果我们是守法的好公民的话就听话，(规则之内’越规’)

# 通过JS或者ajax动态加载的数据我们通过requests模块进行数据爬取无法每次都实现可见即可得

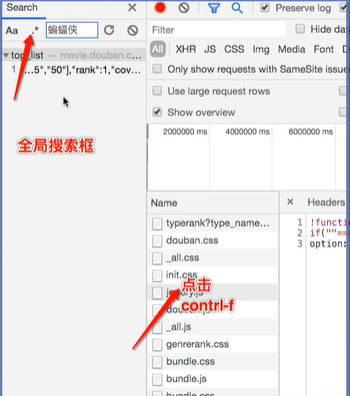


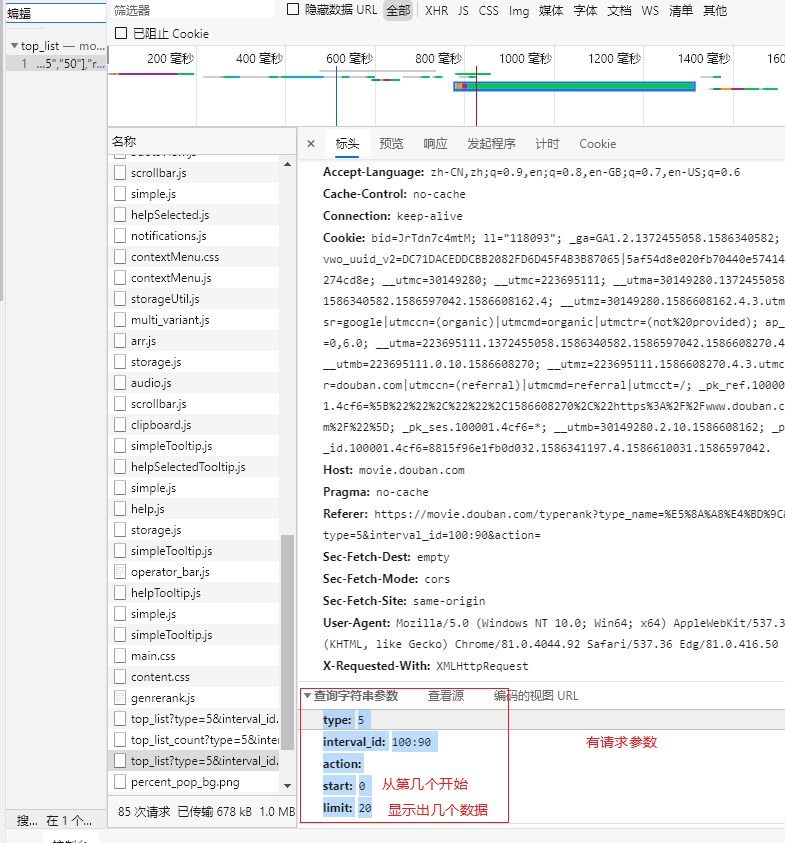


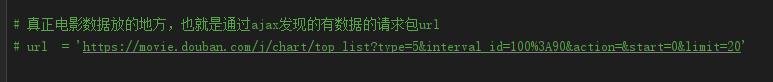


要进行全局搜索来获取那个数据包，下次直接请求这个数据包就可以了而不是初始url

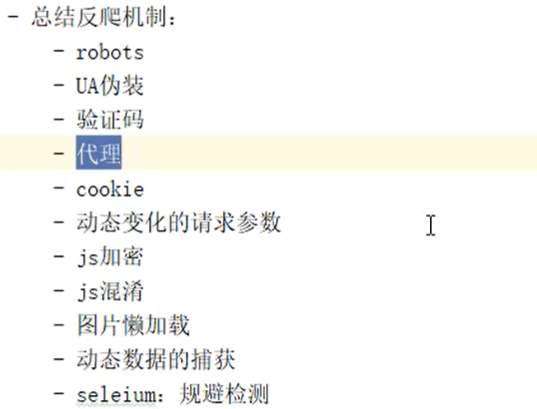
(因为初始url里面并没有想要的数据)











Referer：防盗链