Python 入门网络爬虫之精华版

Author: LiNing

Email: lining0806@gmail.com

Python 学习网络爬虫主要分 3 个大的版块：抓取，分析，存储

另外，比较常用的爬虫框架 Scrapy，这里最后也详细介绍一下。

首先列举一下本人总结的相关文章，这些覆盖了入门网络爬虫需要的基本概念和技巧：宁

哥的小站-网络爬虫

简单来说这段过程发生了以下四个步骤：

查找域名对应的 IP 地址。

向 IP 对应的服务器发送请求。

服务器响应请求，发回网页内容。

浏览器解析网页内容。









网络爬虫要做的，简单来说，就是实现浏览器的功能。通过指定url，直接返回给用户所需

要的数据，而不需要一步步人工去操纵浏览器获取。

抓取

这一步，你要明确要得到的内容是是什么？是HTML 源码，还是 Json 格式的字符串等。

1. 最基本的抓取

首先，Python 中自带 urllib 及 urllib2 这两个模块，基本上能满足一般的页面抓取。另外，

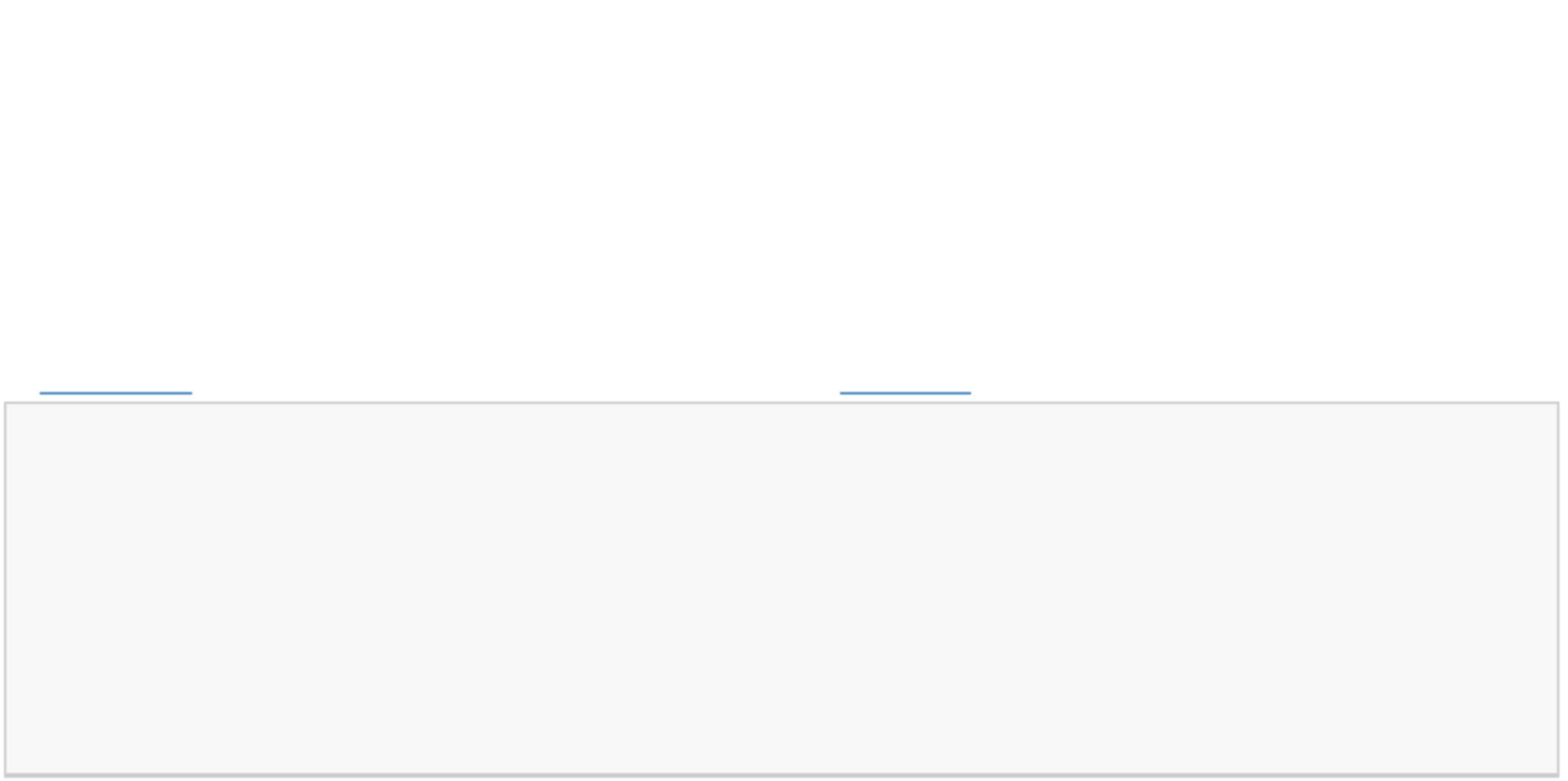
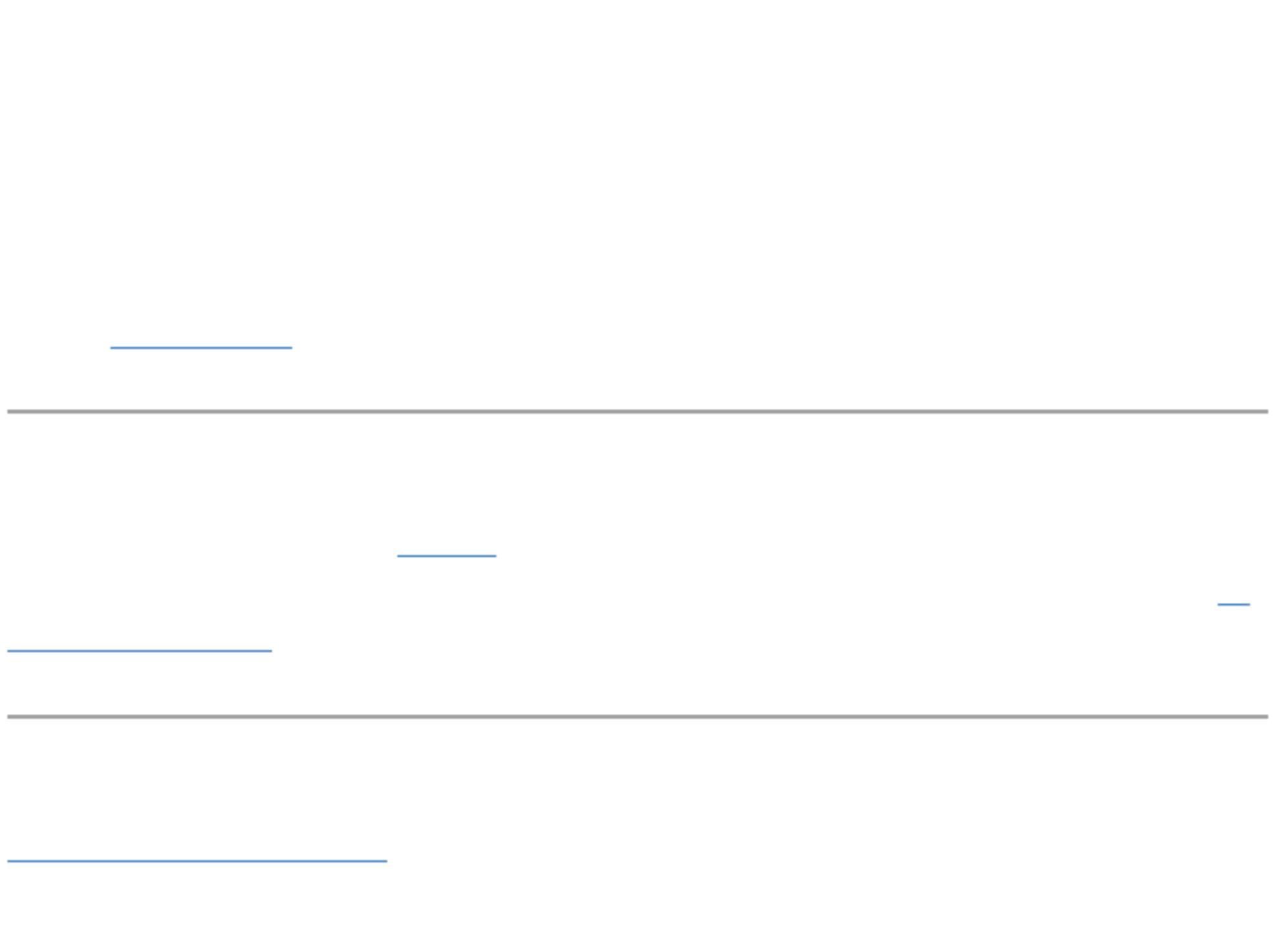
requests 也是非常有用的包，与此类似的，还有 httplib2 等等。

response = requests.get(url)

content = requests.get(url).content

print "response headers:", response.headers

print "content:", content



response = urllib2.urlopen(url)

content = urllib2.urlopen(url).read()

print "response headers:", response.headers

print "content:", content

http = httplib2.Http()

response\_headers, content = http.request(url, 'GET')

print "response headers:", response\_headers

print "content:", content

此外，对于带有查询字段的 url，get 请求一般会将来请求的数据附在 url 之后，以?分割 url

和传输数据，多个参数用&连接。

data = {'data1':'XXXXX', 'data2':'XXXXX'}

Requests：data为 dict，json

import requests

response = requests.get(url=url, params=data)

Urllib2：data为 string

import urllib, urllib2

data = urllib.urlencode(data)

full\_url = url+'?'+data

response = urllib2.urlopen(full\_url)

这种情况属于 post 请求，即先向服务器发送表单数据，服务器再将返回的cookie 存入本

地。

data = {'data1':'XXXXX', 'data2':'XXXXX'}

Requests：data为 dict，json

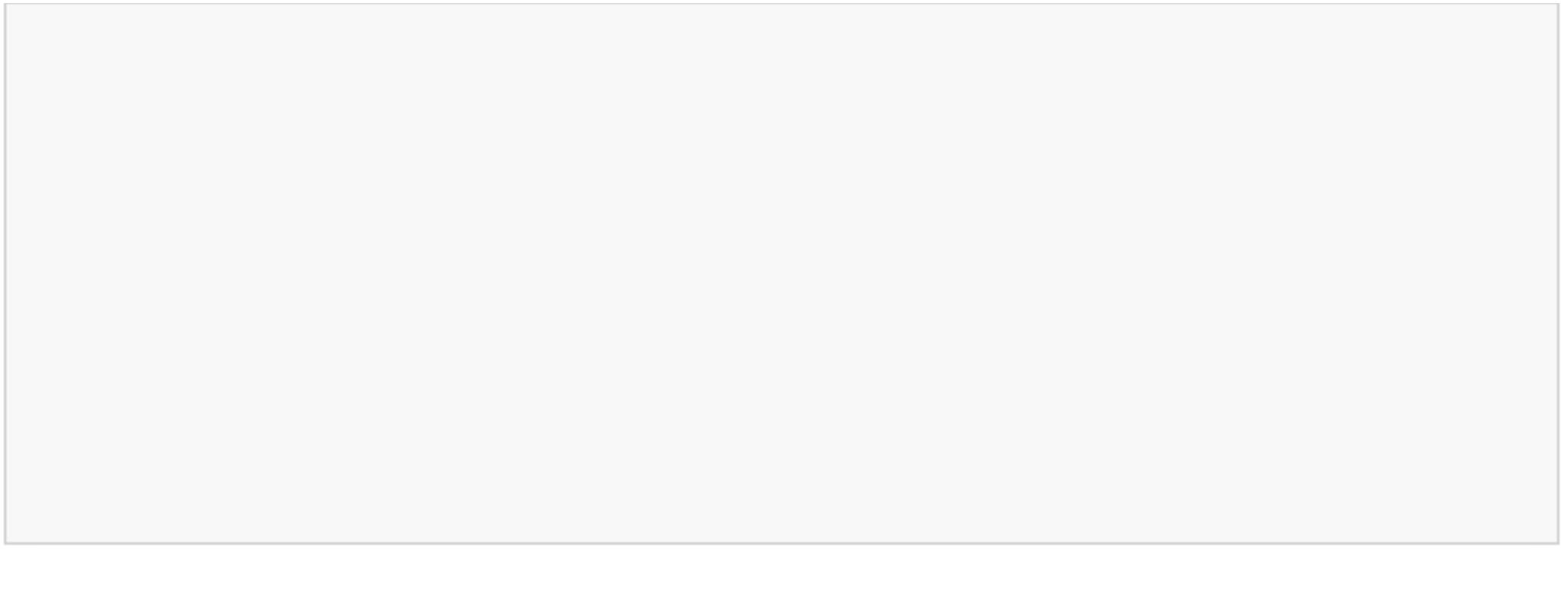
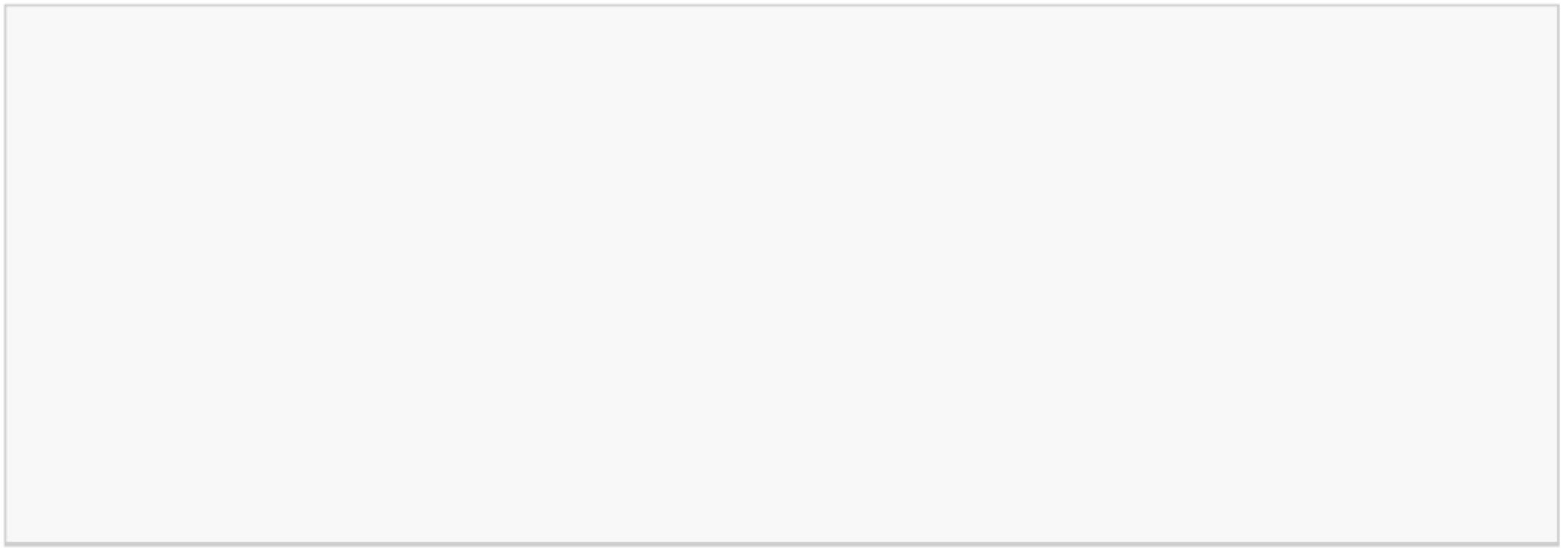
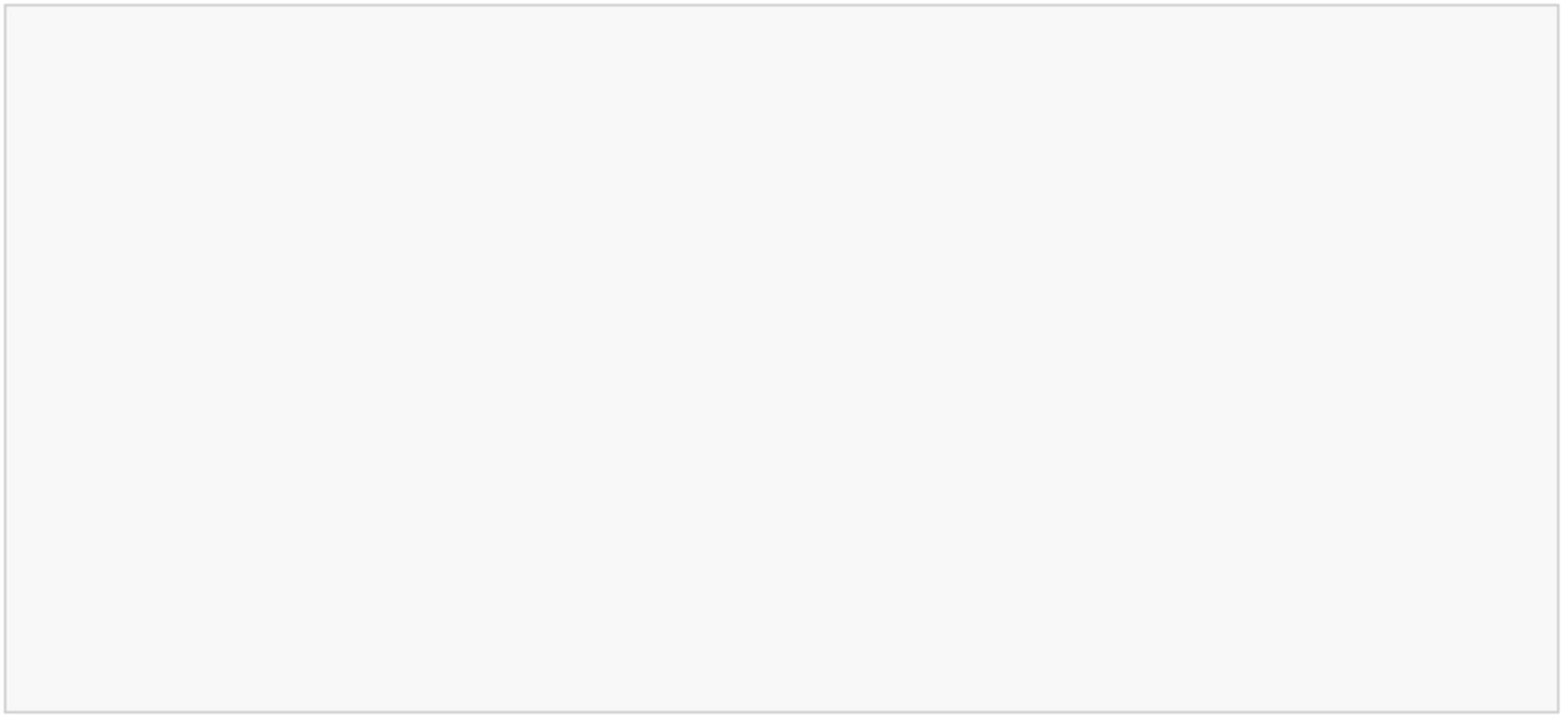
import urllib, urllib2

data = urllib.urlencode(data)

req = urllib2.Request(url=url, data=data)

response = urllib2.urlopen(req)

2.2 使用 cookie 登陆



使用 cookie 登陆，服务器会认为你是一个已登陆的用户，所以就会返回给你一个已登陆的

内容。因此，需要验证码的情况可以使用带验证码登陆的cookie 解决。

requests\_session = requests.session()

response = requests\_session.post(url=url\_login, data=data)

若存在验证码，此时采用 response = requests\_session.post(url=url\_login, data=data)是

不行的，做法应该如下：

response2 = requests\_session.get(url\_login) # 已登陆，因为之前拿到了 Response Cookie！

response3 = requests\_session.get(url\_results) # 已登陆，因为之前拿到了 Response

Cookie！

相关参考：网络爬虫-验证码登陆

参考项目：爬取知乎网站

3. 对于反爬虫机制的处理

适用情况：限制 IP 地址情况，也可解决由于“频繁点击”而需要输入验证码登陆的情况。

这种情况最好的办法就是维护一个代理 IP 池，网上有很多免费的代理 IP，良莠不齐，可

以通过筛选找到能用的。对于“频繁点击”的情况，我们还可以通过限制爬虫访问网站的频

率来避免被网站禁掉。

response = requests.get(url=url, proxies=proxies)

Urllib2：

proxy\_support = urllib2.ProxyHandler(proxies)

opener = urllib2.build\_opener(proxy\_support, urllib2.HTTPHandler)

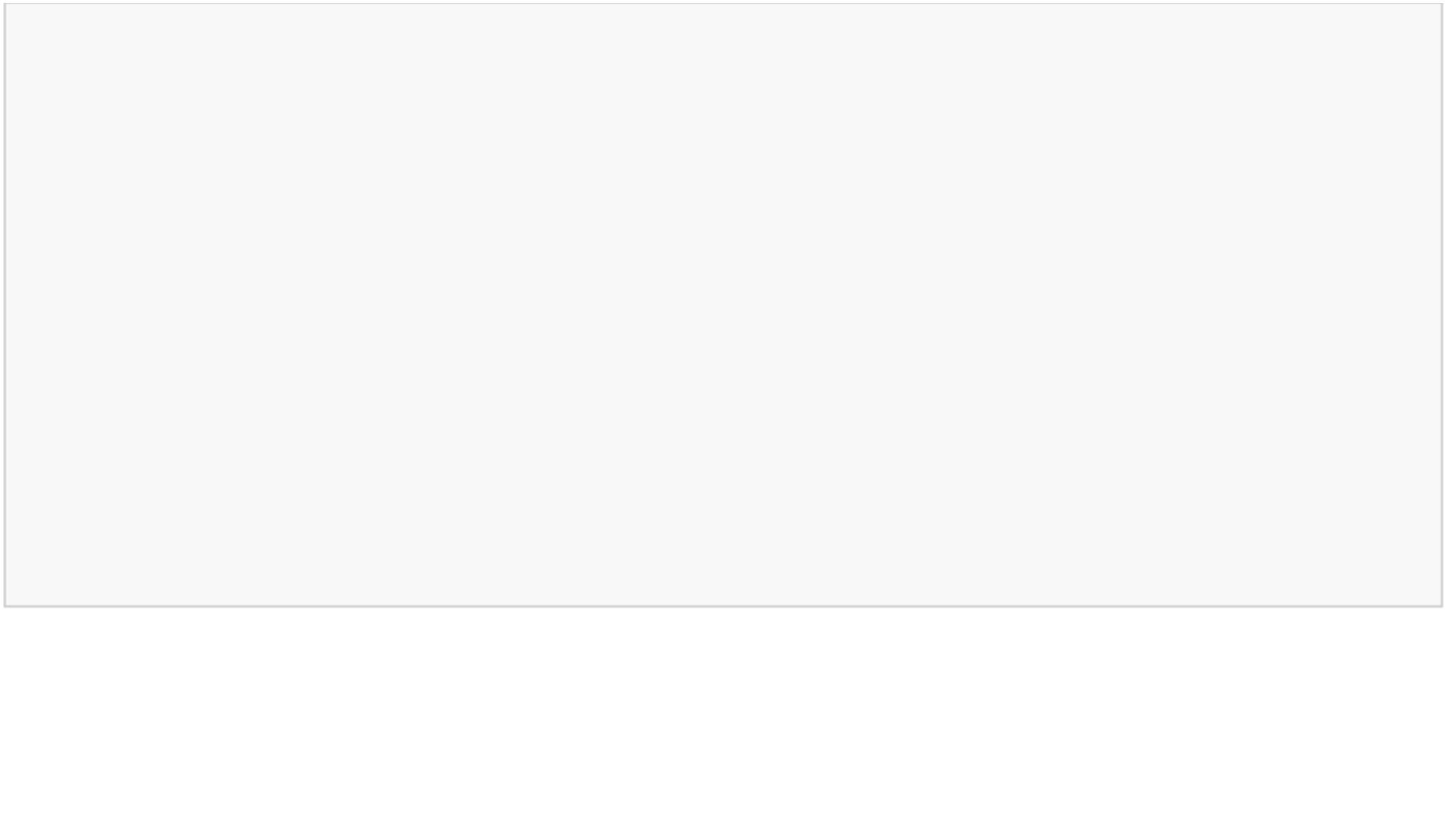
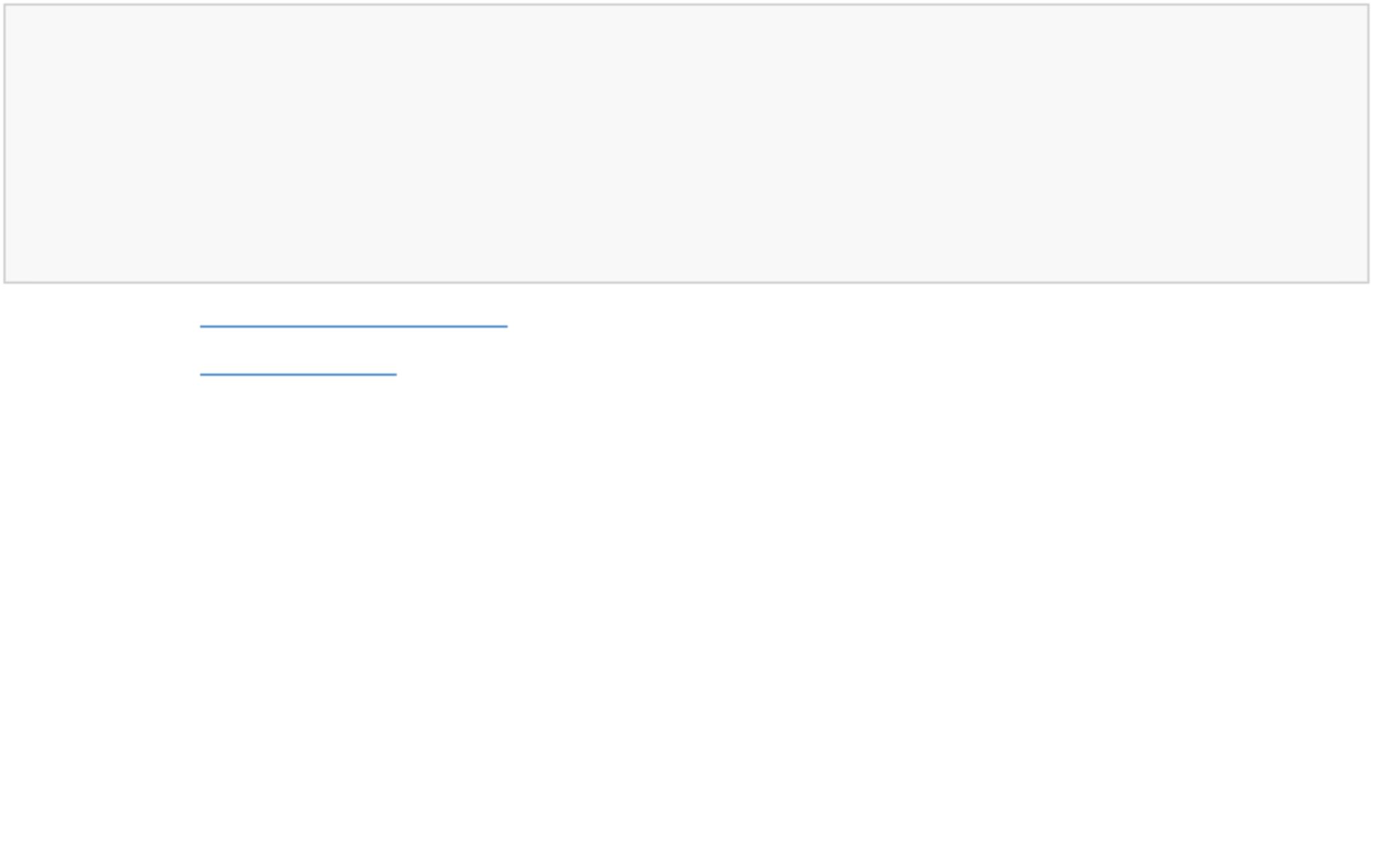
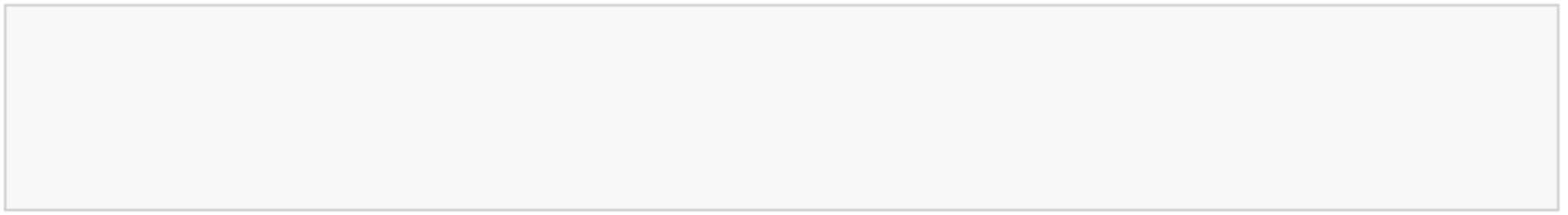
urllib2.install\_opener(opener) #安装 opener，此后调用 urlopen()时都会使用安装过的

opener对象

response = urllib2.urlopen(url)

适用情况：限制频率情况。

Requests，Urllib2 都可以使用 time 库的 sleep()函数：



有些网站会检查你是不是真的浏览器访问，还是机器自动访问的。这种情况，加上User-

Agent，表明你是浏览器访问即可。有时还会检查是否带Referer 信息还会检查你的

Referer 是否合法，一般再加上 Referer。

headers = {'User-Agent':'XXXXX', 'Referer':'XXXXX'}

Requests：

response = requests.get(url=url, headers=headers)

Urllib2：

import urllib, urllib2

req = urllib2.Request(url=url, headers=headers)

response = urllib2.urlopen(req)

不多说。

def multi\_session(session, \*arg):

while True:

retryTimes = 20

while retryTimes>0:

try:

return session.post(\*arg)

except:

retryTimes -= 1

或者

def multi\_open(opener, \*arg):

while True:

retryTimes = 20

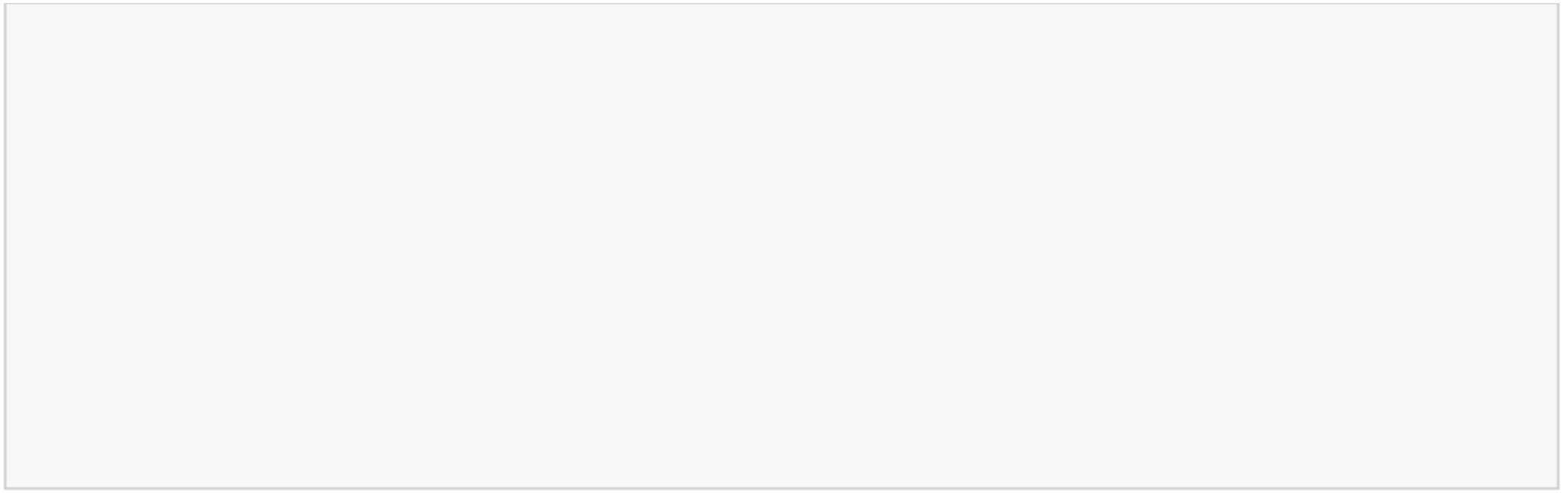
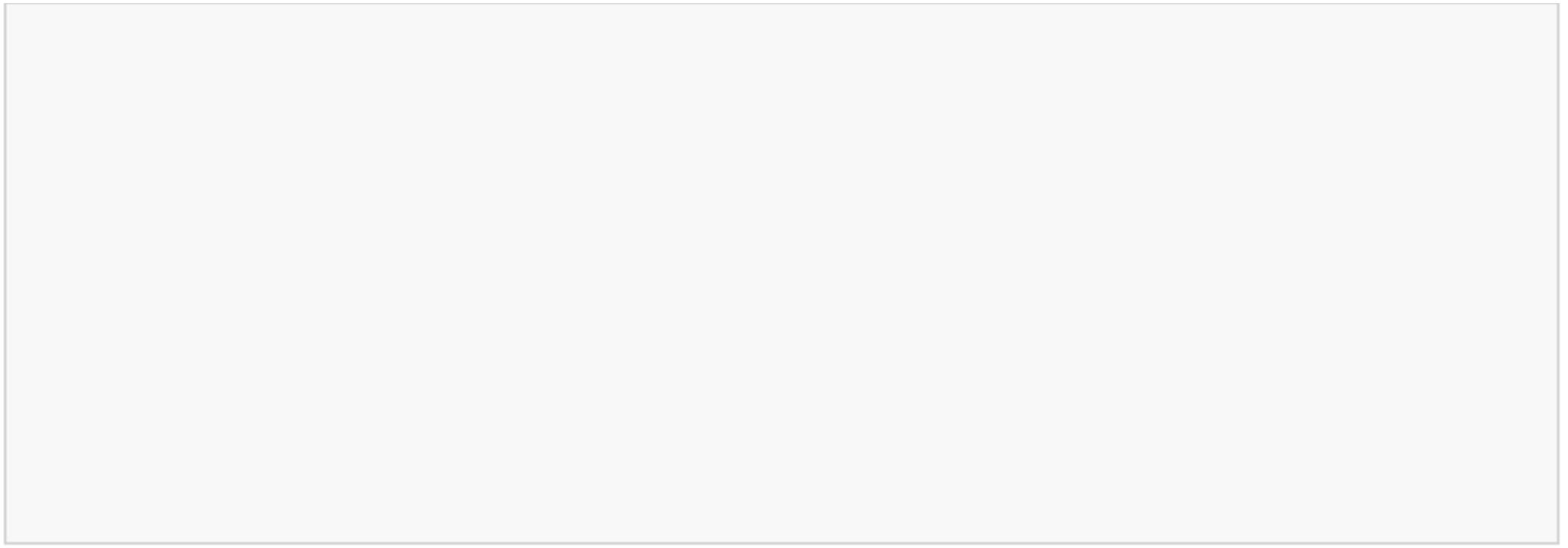
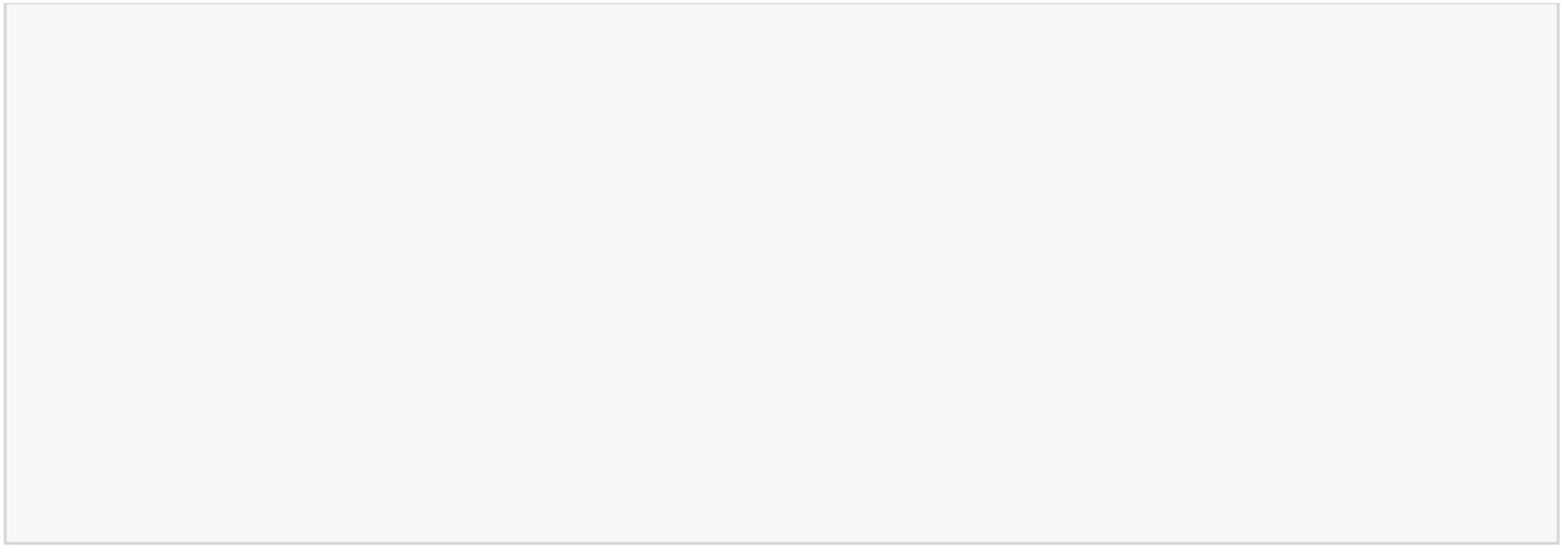
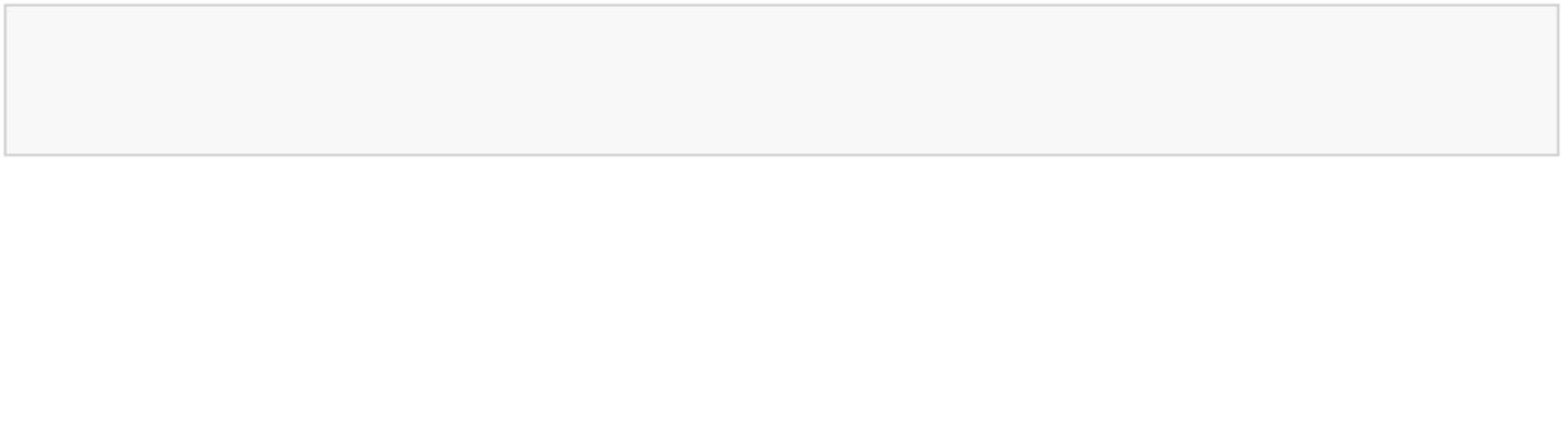
while retryTimes>0:

try:

return opener.open(\*arg)

except:

print '.',



retryTimes -= 1

这样我们就可以使用 multi\_session 或 multi\_open 对爬虫抓取的 session 或 opener 进行保

持。

5. 多进程抓取

这里针对华尔街见闻进行多进程抓取的实验对比：Python 多进程抓取 与 Java 多进程抓取

相关参考：关于 Python 和 Java 的多进程多线程计算方法对比

6. 对于 Ajax 请求的处理

它的工作原理是：从网页的 url 加载网页的源代码之后，会在浏览器里执行JavaScript 程

序。这些程序会加载更多的内容，“填充”到网页里。这就是为什么如果你直接去爬网页本

身的 url，你会找不到页面的实际内容。

这里，若使用 Google Chrome 分析”请求“对应的链接(方法：右键→审查元素→Network→

清空，点击”加载更多“，出现对应的 GET 链接寻找 Type 为 text/html 的，点击，查看 get

参数或者复制 Request URL)，循环过程。

如果“请求”之前有页面，依据上一步的网址进行分析推导第1 页。以此类推，抓取抓 Ajax

地址的数据。

对返回的 json 格式数据(str)进行正则匹配。json 格式数据中，需从'\uxxxx'形式的

unicode\_escape 编码转换成 u'\uxxxx'的 unicode 编码。

7. 自动化测试工具 Selenium

Selenium 是一款自动化测试工具。它能实现操纵浏览器，包括字符填充、鼠标点击、获取

元素、页面切换等一系列操作。总之，凡是浏览器能做的事，Selenium 都能够做到。

这里列出在给定城市列表后，使用 selenium 来动态抓取去哪儿网的票价信息的代码。

相关参考：网络爬虫之 Selenium 使用代理登陆：爬取去哪儿网站

8. 验证码识别

对于网站有验证码的情况，我们有三种办法：

使用代理，更新 IP。

使用 cookie 登陆。

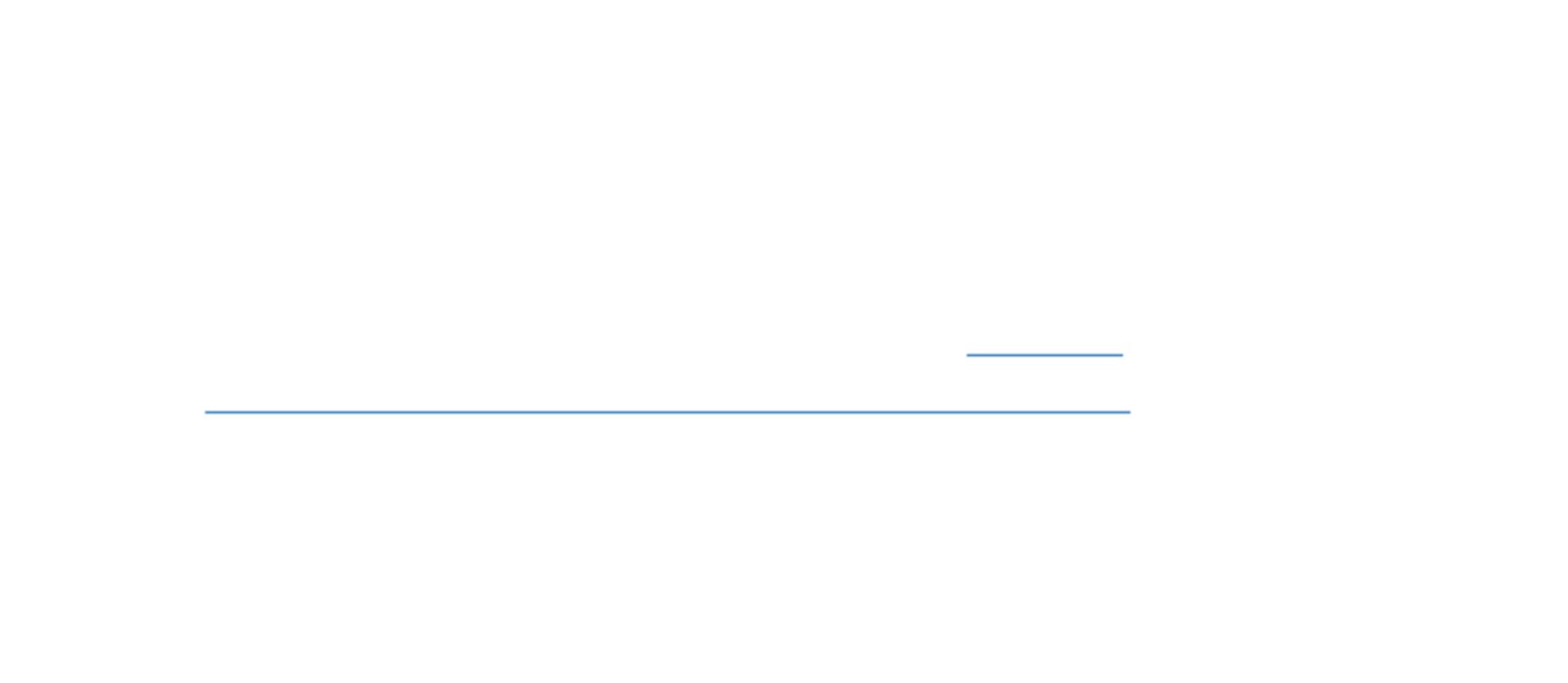
验证码识别。







使用代理和使用 cookie 登陆之前已经讲过，下面讲一下验证码识别。



可以利用开源的 Tesseract-OCR 系统进行验证码图片的下载及识别，将识别的字符传到爬

虫系统进行模拟登陆。如果不成功，可以再次更新验证码识别，直到成功为止。

参考项目：Captcha1

爬取还有一个需要注意的问题：

如何监控一系列网站的更新情况，也就是说，如何进行增量式爬取？



分析

抓取之后就是对抓取的内容进行分析，你需要什么内容，就从中提炼出相关的内容来。

常见的分析工具有正则表达式，BeautifulSoup，lxml 等等。

存储

分析出我们需要的内容之后，接下来就是存储了。

我们可以选择存入文本文件，也可以选择存入MySQL 或 MongoDB 数据库等。





如何进行内容去重？

Scrapy

Scrapy 是一个基于 Twisted 的开源的 Python 爬虫框架，在工业中应用非常广泛。

相关内容可以参考基于 Scrapy 网络爬虫的搭建，同时给出这篇文章介绍的微信搜索爬取

的项目代码，给大家作为学习参考。

参考项目：使用 Scrapy 或 Requests 递归抓取微信搜索结果

