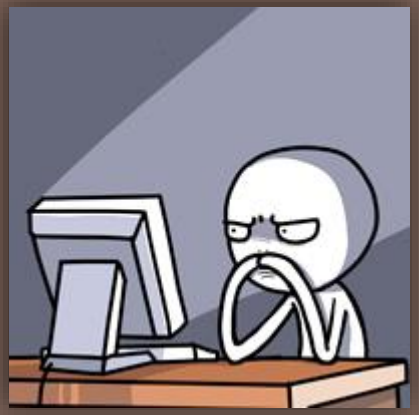
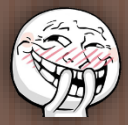
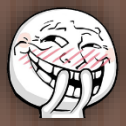
By Yanju Chen

网络爬虫浅入浅出

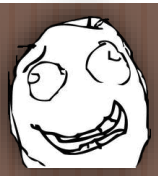


又一个关于王尼玛的故事……

第一话：邂逅

很久很久以前，王尼玛  邂逅了自己的女神，王尼美  ，为了得到女神的芳心，王尼玛  绞尽脑汁，终于知道了女神有一个个人网站，上面放了很多“有趣”  的随笔资料，王尼玛决定先去看看这些随笔资料。

第一话：邂逅

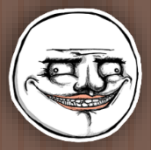
先不管女神智商问题，王尼玛  发现女神资料按照如下规律排布：

http://nimei.xxx/1.html

http://nimei.xxx/2.html

……

http://nimei.xxx/1000.html

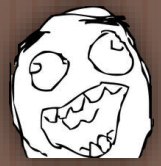
（一共1000个页面，同时请忽略域名  ）

王尼玛发现自己一个个点一个个修改数字位置太麻烦，有没有简单一点的方法呢？ 

第一话：邂逅

然后王尼玛就去求助他的好朋友，也是Python大牛的曹尼玛  ……

 （曹尼玛）：天了噜，这么简单的问题，用Python库就可以了！

 （王尼玛）：真的吗？太好了！

然后王尼玛就拿到了下面这个代码片段并成功拿到了女神网站资料：

```
# Story1
import urllib2
s1_opener = urllib2.build_opener()
for i in range(1001):
    s1_response = s1_opener.open("http://nimei.xxx/%s.html"%i)
    # Read the Content
```

为什么这个代码就能拿到女神网站资料呢？

曹尼玛的小黑屋时间#1

使用Python的urllib2库完成丰富的网络相关操作：

```
1   # Story1
2   import urllib2
3   s1_opener = urllib2.build_opener()
4   for i in range(1001):
5       s1_response = s1_opener.open("http://nimei.xxx/%s.html"%i)
6       # Read the Content
```

考虑网络的不稳定性，曹尼玛给你的小建议：添加异常处理

```
 8   # Story1:Suggested
 9   import urllib2
10   s1_opener = urllib2.build_opener()
11   for i in range(1001):
12       try:
13           s1_response = s1_opener.open("http://nimei.xxx/%s.html"%i)
14           # Read the Content
15       except Exception,e:
16           # Do Something
```

异常处理还可以解决更多扩展的问题哦，比如未知序列……

第二话：初识

经过曹尼玛的指点，王尼玛  凭借对女神的了解，成功认识了王尼美。王尼美  对于王尼玛为什么这么了解自己非常好奇。

 （王尼美）：难道你是来自星星的欧巴?

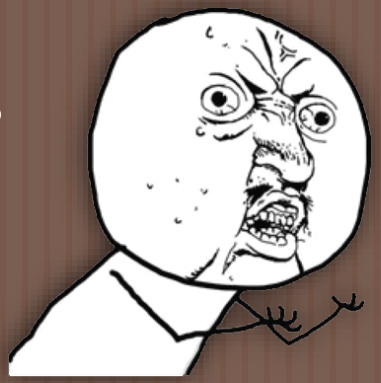
 （王尼玛）：我不会告诉你我什么都知道的……

 （王尼美）：那你一定知道我喜欢吃什么吧?

 （王尼玛）：当……当然知道了……

第二话：初识

回到家里，王尼玛赶紧打开王尼美的个人网站，王尼美的网站新增了日记的栏目，可是令他吃惊的是，点开首页的一篇日记之后，王尼玛发现网站的结构变了，原来有规律的网址变成了下面这种形式：



http://nimei.xxx/C4CA4238A0B923820DCC509A6F75849B

http://nimei.xxx/C81E728D9D4C2F636F067F89CC14862C

http://nimei.xxx/ECCBC87E4B5CE2FE28308FD9F2A7BAF3

……

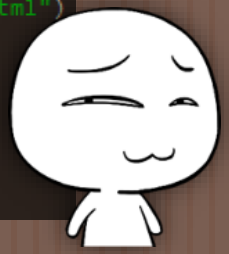
虽然页面上有所有日记的链接可以点，但是要一个个点也是醉了……

没办法，王尼玛只能再去求助曹尼玛  了……

曹尼玛的小黑屋时间#2

对于无规律或者散列处理后的网址，能够得到接入链接的，可以通过Python库BeautifulSoup获得页面的链接地址：

```
19  # Story2
20  import urllib2
21  import BeautifulSoup
22  s1_opener = urllib2.build_opener()
23  try:
24      s1_response = s1_opener.open("http://nimei.xxx/index.html")
25      # Read the Content
26      # Deal with BeautifulSoup, Extract An UrlList
27      urlList = []
28      for i in urlList:
29          s1_response = s1_opener.open(i)
30  except Exception,e:
31      # Do Something
```



鉴于BS库不是默认的标准库，可以手动用字符串检索匹配的方式获得超链接，不需要额外库的支持，超链接的一般正则表达式如下：

<a.+?href=(['"]?)([^>\s]+)\1.*?>([\S\s]+?)<\/a>

代码for部分可以写成递归形式，这也是大部分蜘蛛（爬虫）的基本运行形式。

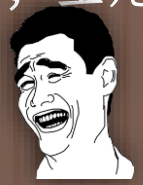
但是其实，如果你有X雷，可以直接点“下载全部”……

第三话：心事

经过曹尼玛  的又一次帮助，王尼玛终于成功阅读了尼美的日记，知道了她喜欢吃汉堡  ，在从XX基吃完汉堡回来的路上，王尼玛发现王尼美闷闷不乐的样子  ，不知道为什么，问了也不肯说。不过，王尼玛记得尼美说过自己有一本私密日记本  ，上面写着一些自己最近发生的事情，或许可以去看看在尼美网站能不能找到这个私密日记本。

第三话：心事

王尼玛回到家又打开了电脑，仔细看了看尼美的主页，果然  发现一个登录界面，用admin和尼美的生日+手机+邮箱+地址+最喜欢的数字+最喜欢的食物作为密码的组合成功进入私密日记。 

得到了私密日记的地址序列后，王尼玛用之前曹尼玛写的代码开始抓，发现抓下来都是错误页面  ，没办法，只能又去找曹尼玛了…… 

曹尼玛的小黑屋时间#3

对于带有身份认证的页面的处理方法，根据对HTTP协议的分析，一般网站登录后在客户端留有cookies当作下次请求的验证信息，只需要在请求的头（header）附带cookies信息即可。

```
19  # Story3
20  import urllib2
21  import BeautifulSoup
22  s1_opener = urllib2.build_opener()
23  s1_opener.addheaders = [("Cookies","SESSIONID=ECCBC87E4B5CE2FE28308FD9F2A7BAF3")]
24  try:
25      s1_response = s1_opener.open("http://nimei.xxx/index.html")
26      # Read the Content
27      # Deal with BeautifulSoup, Extract An UrlList
28      urllist = []
29      for i in urllist:
30          s1_response = s1_opener.open(i)
31  except Exception,e:
32      # Do Something
```

当然，服务器还可能对请求头作更多的限制，常见的有：

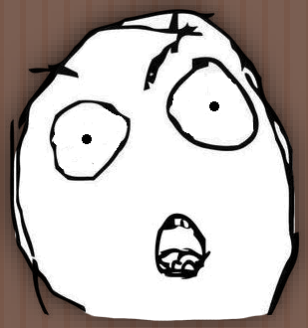
Host/Referer/User-Agent

可以通过浏览器Network捕获并构造“合法”的请求，然后在Python中重现。

第四话：情敌

经过曹尼玛的又一次帮助，知道真相的王尼玛想好了安慰尼美的计划，

正当王尼玛想约尼美出来的时候，发现尼美跟另外一个男生在一起……



……

后来才知道，不只王尼玛一个人对王尼美一见钟情，王尼美是很多人的

大众女神，王尼玛决定要击败这些情敌！

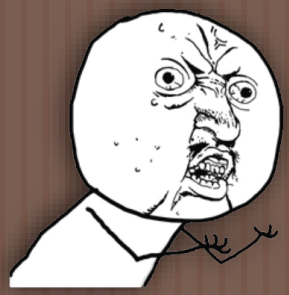
第四话：情敌

王尼玛回到家又打开了电脑，想从王尼美网站的私密日记再找到线索，

看看别人都做了些什么，好让自己想到比他们更好的计划。 

可是，因为王尼美的网站访问量太大，服务器加入了单个ip请求时间

间隔的限制，请求太快还会有验证码，大量请求还会被封ip一天……



……然后王尼玛又只能屁颠屁颠地去找曹尼玛了。 

曹尼玛的小黑屋时间#4

有不少访问流量大的网站会对访问者的请求作出时间或空间上的限制，即使访问者已经登录，网站还是会执行一些机器人的检测手段，如：验证码等。

对于时间限制，只需要在爬虫中加入休眠代码即可；而对于想要提升速度的话，可以考虑执行多个爬虫并发连接（多线程/多进程）也可以（留意网站对于多线程的限制）。

```
35  # Story4
36  import urllib2
37  import BeautifulSoup
38  s1_opener = urllib2.build_opener()
39  s1_opener.addheaders = [("Cookies","SESSIONID=ECCBC87E4B5CE2FE28308FD9F2A7BAF3")]
40  try:
41      s1_response = s1_opener.open("http://nimei.xxx/index.html")
42      # Read the Content
43      # Deal with BeautifulSoup, Extract An UrlList
44      urlList = []
45      for i in urlList:
46          s1_response = s1_opener.open(i)
47          time.sleep(13)
48  except Exception,e:
49      # Do Something
```

不过，对于ip黑名单或者验证码这些硬限制，能够绕开的方法有限，通常需要暴力解决（寻找代理及手工输入验证码）。

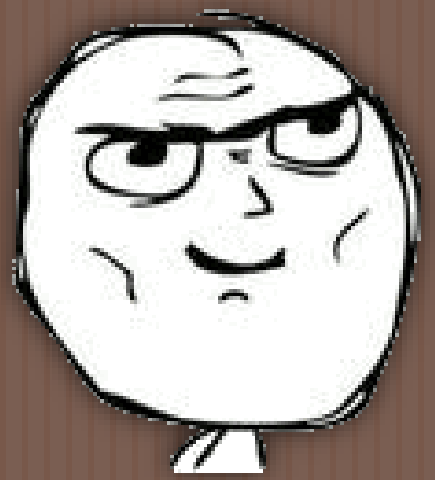
第四话：情敌

在曹尼玛的帮助下，王尼玛夜以继日地对着电脑打……验证码，每天实时掌握情敌信息，同时作出高人一筹的追尼美计划。

就这样日复一日，终于有一天，王尼玛追到了王尼美。

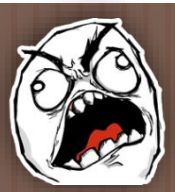
两个人最后幸福地生活在一起。

（全剧终）



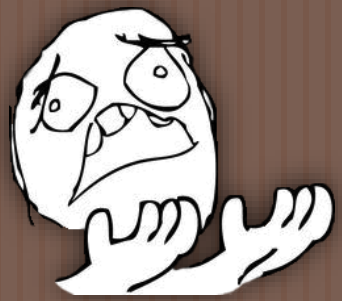

你TM在逗我

第五话：结局

第二天王尼玛醒来，发现自己原来做了个梦。 

在连续输入了4294967296个验证码后，王尼玛发现自己的电脑崩溃了，重新开机的时候发现曹尼玛写给他的爬虫已经吐不出任何验证码了。

手动打开女神尼美的网站，王尼玛发现了403Forbidden错误，无论怎么尝试，都没办法获得任何内容了……



这回万能的曹尼玛也帮不了王尼玛了  ，无论做什么都没办法破解王尼美网站的这个403反爬虫技术。

所以最后王尼美就跟别人走了……（全剧终）


注定孤独一生

曹尼玛的小黑屋时间#5

其实前文所谓“不能破解的403反爬虫技术”只是服务器临时出了点问题，禁止了所有人访问。但是是否存在更好的反爬虫技术呢?

多数爬虫类似于httpClient类型，比如Python、wget等发出请求，实际上不会构造html结构树，更不会解析页面的javascript（动作层），基于此，可以构造一些特殊反爬虫页面，比如:

- js设置cookies
- 动态生成临时链接
- 表单隐藏域（hidden field）

不过，一般来说，验证码就够呛了……

曹尼玛的爬虫/反爬虫攻略

服务器反爬虫策略	爬虫对应策略	曹尼玛怎么看
散列化地址	获得地址列表后递归发出请求	“这简直侮辱我的智商！”
请求头限制	录制正常请求并用脚本重现	“太简单没什么意思！”
请求时空限制	增加时间间隔，ip分布式多线程请求，验证码手动。	“你敢玩阴的我就敢人肉暴力破解。”
动作层方式限制	附加爬虫到浏览器，增加表现层解析能力，分析静态js逻辑。	“比较麻烦，有些可解，有些无解。”
其它策略	其它对应策略	“兵来将挡，水来土掩，最不行就社会工程学。”