

2019

Python爬虫与数据分析

——清远市中学教师Python编程培训

林世明
厦门大学信息学院

自我介绍

Introduction

林世明 厦门大学信息学院工程师

民主建国会厦门大学支部委员

厦门大学健康医疗大数据研究所所长助理

厦门市自动化学会副秘书长

厦门市书法家协会理事

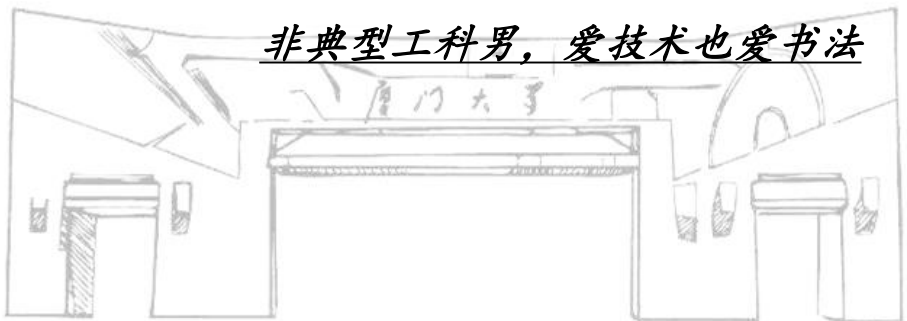
研究方向：信息化、大数据

主持参与多项横纵向项目

发表论文若干

拥有多项软件著作权

非典型工科男，爱技术也爱书法



自强不息 止于至善

目录

Content



引言



准备知识

- 1、爬虫工作原理
- 2、HTML、CSS、JavaScript
- 3、HTTP、URL



网页请求

- 1、爬虫的一般步骤
- 2、分析网页结构
- 3、requests库



网页解析

- 1、如何解析网页
- 2、beautifulsoup
- 3、xpath
- 4、re正则表达式



采集数据

- 1、循环
- 2、异常处理
- 3、数据存储



如何应对反爬



Part 1 引言



爬虫能做什么?

爬虫能做什么?

微博最近有啥热门话题

淘宝京东上某商品价格变动邮箱通知

女/男神最新动态

谁点赞最活跃

社交网络分析

抢票软件

学生成绩分析

....



爬虫能做什么?



新闻聚
合阅读
器



搜索
引擎

美女
图库

爬取数据

热门
段子

商品比
价网



房价
预测

社交
平台



Part 2

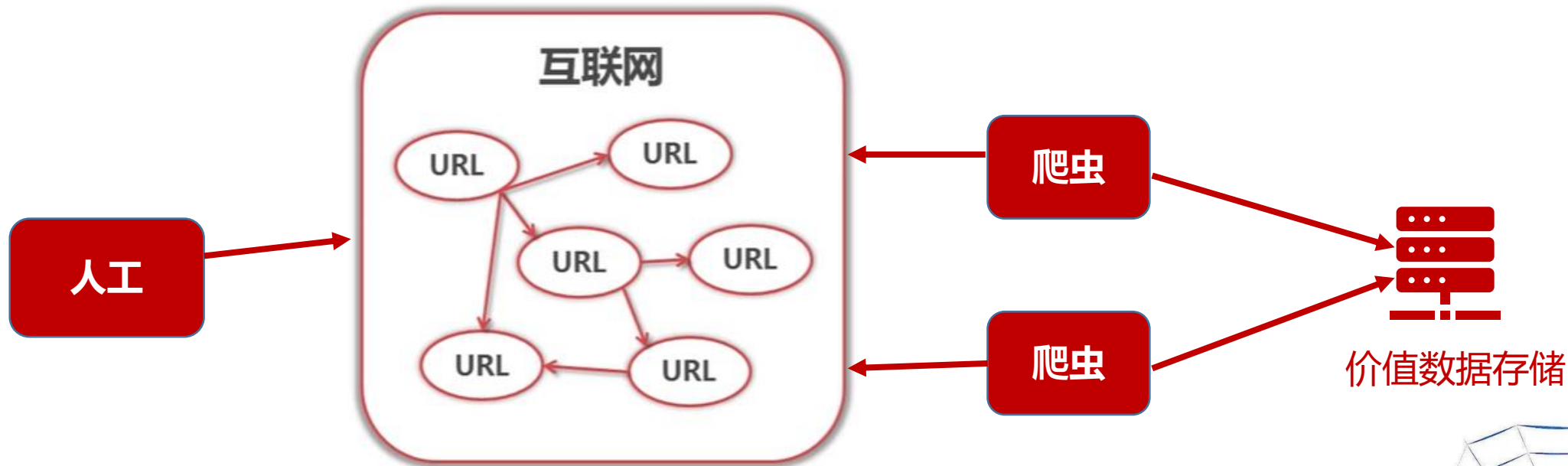
准备知识

2.1 爬虫工作原理

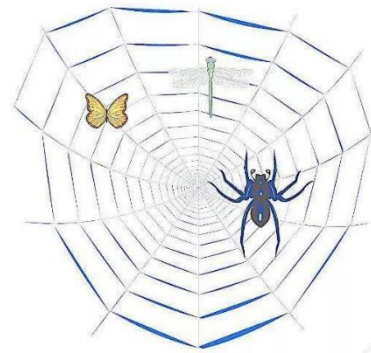
2.2 HTML、CSS、JavaScript

2.3 HTTP、URL

网络爬虫（又称网页蜘蛛、网络机器人），是一种按照一定规则，**自动抓取互联网信息**的程序或者脚本。



爬虫：我们不生产数据，我们只是数据的搬运工！



HTML (Hypertext Marked Language, 超文本标记语言)，是一种标识性的语言。它包括一系列标签。通过这些标签可以将网络上的文档格式统一，使分散的Internet资源连接为一个逻辑整体。HTML文本是由HTML命令组成的描述性文本，HTML命令可以说明文字，图形、动画、声音、表格、链接等。

HTML是目前通用的互联网网页传输语言，目前版本号已经到5，大家经常听到的H5，就是指的HTML5版本。

CSS (Cascading Style Sheets, 层叠样式表)是一种用来表现HTML ([标准通用标记语言](#)的一个应用) 或XML ([标准通用标记语言](#)的一个子集) 等文件样式的计算机语言。

JavaScript，最早是在HTML ([标准通用标记语言](#)下的一个应用) 网页上使用，用来给HTML网页增加动态功能。

网页三剑客

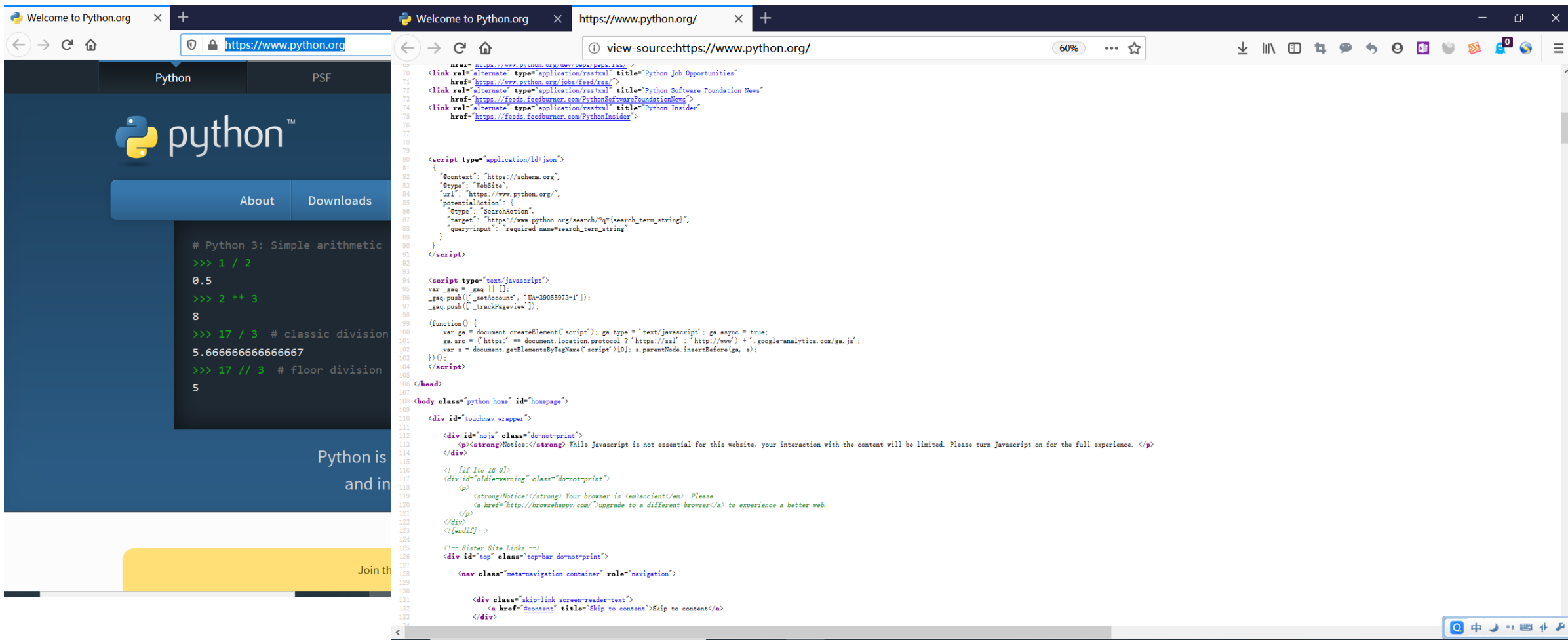


HTML

访问: <https://www.python.org/>

浏览器端显示:

HTML源代码:



The image displays a web browser window with two tabs. The left tab shows the Python.org homepage, which features the Python logo, navigation links for 'About' and 'Downloads', and a code snippet demonstrating simple arithmetic operations in Python 3. The right tab shows the source code of the same page, with the address bar indicating 'view-source:https://www.python.org/'. The source code is displayed in a monospaced font, showing the HTML structure, including links to Python job opportunities, foundation news, and the Python Insider feed, as well as a search script and a JavaScript function for Google Analytics.

```
70 <link rel="alternate" type="application/rss+xml" title="Python Job Opportunities"
71 href="https://www.python.org/jobs/feed/rss/">
72 <link rel="alternate" type="application/rss+xml" title="Python Software Foundation News"
73 href="https://feeds.feedburner.com/PythonSoftwareFoundationNews">
74 <link rel="alternate" type="application/rss+xml" title="Python Insider"
75 href="https://feeds.feedburner.com/PythonInsider">
76
77
78
79
80 <script type="application/ld+json">
81 {
82   "@context": "https://schema.org",
83   "@type": "WebSite",
84   "url": "https://www.python.org/",
85   "potentialAction": {
86     "@type": "SearchAction",
87     "target": "https://www.python.org/search/?q={search_term_string}",
88     "query-input": "required name=search_term_string"
89   }
90 }
91 </script>
92
93 <script type="text/javascript">
94 var __gaq = __gaq || [];
95 _gaq.push(['_setAccount', 'UA-39055973-1']);
96 _gaq.push(['_trackPageview']);
97
98 (function() {
99   var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
100   ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
101   var s = document.getElementsByTagName('script')[0].parentNode.insertBefore(ga, s);
102 })();
103 </script>
104
105 </head>
106
107 <body class="python home" id="homepage">
108 <div id="touchnav-wrapper">
109
110 <div id="nojs" class="do-not-print">
111 <p><strong>Notice:</strong> While Javascript is not essential for this website, your interaction with the content will be limited. Please turn Javascript on for the full experience. </p>
112 </div>
113
114 <!--[if lte IE 8]>
115 <div id="oldie-warning" class="do-not-print">
116 <p>
117 <strong>Notice:</strong> Your browser is <em>ancient</em>. Please
118 <a href="http://browsehappy.com/">upgrade to a different browser</a> to experience a better web.
119 </p>
120 </div>
121 <![endif]-->
122
123 <!-- Sister Site Links -->
124 <div id="top" class="top-bar do-not-print">
125
126 <nav class="meta-navigation container" role="navigation">
127
128
129
130
131 <div class="skip-link screen-reader-text">
132 <a href="#content" title="Skip to content">Skip to content</a>
133 </div>
```

HTTP

HTTP(HyperText Transfer Protocol,超文本传输协议)是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。HTTP是一个客户端和服务端请求和应答的标准 (TCP)。

常见的HTTP请求方式:

get请求:

将请求参数追加在url后面, 不安全
url长度限制get请求方式数据的大小
没有请求体

一般的HTTP请求大多都是GET。

post请求:

请求参数在请求体处, 较安全。

请求数据大小没有显示

只有表单设置为method= "post" 才是post
请求, 其他都是get请求

此外还有HEAD请求、DELETE请求、OPTIONS请求、PUT
请求、TRACE请求、CONNECT请求

用户在浏览器输入网址:
http://www.baidu.com



URL: 统一资源定位符(Uniform Resource Locator),

它的一般格式如下(带方括号[]的为可选项):

protocol :// hostname[:port] / path / [;parameters][?query]#fragment

URL的格式由三部分组成:

(1)protocol: 第一部分就是协议, 有http、https;

(2)hostname[:port]: 第二部分就是主机名(还有端口号为可选参数), 一般网站默认的端口号为80, 例如百度的主机名就是www.baidu.com, 这个就是服务器的地址;

(3)path: 第三部分就是主机资源的具体地址, 如目录和文件名等。

网络爬虫就是根据这个URL来获取网页信息的。

Part 3

网页请求

3.1 爬虫的一般步骤

3.2 分析网页结构

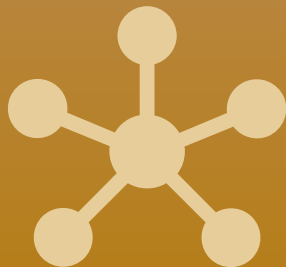
3.3 requests库

爬虫的一般步骤



分析要爬取网站的情况

- 链接跳转
- 网页代码结构



编写爬虫程序

- 使用爬虫框架
- 不使用爬虫框架
- 解析HTML网页
- 提取关键信息



数据存储

- 存储到文件
- 存储到数据库



数据分析

- 数据挖掘
- 数据聚合
- 预测
-

分析网页结构



网页生成方式

静态HTML还是JavaScript动态生成。动态生成的网页抓取比较麻烦，需要浏览器解析才能看到最终的内容，一般要用到phantomjs模块

构造、提取网页链接

要让爬虫自动爬取网页数据，爬取的内容最好是结构化的，要求要有一定的规律；而不是像搜索引擎是把所有的内容都先存储下来



Requests is an elegant and simple HTTP library for Python, built for human beings.

Requests是一个优雅简洁的Python HTTP库，给人类使用

- ✓ requests是python实现的简单易用的HTTP库，使用起来比urllib简洁很多
- ✓ 因为是第三方库，所以使用前需要自行安装
- ✓ `pip install requests`，安装完成后import一下，正常则说明可以开始使用了。

最普通的访问

```
url = 'http://www.jianshu.com/u/1562c7f164'  
r = requests.get(url)
```

伪装成浏览器的访问

```
Headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X  
10_12_3) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/56.0.2924.87 Safari/537.36'}  
r = requests.get(url, headers = Headers)
```


使用cookie访问

Cookie =

```
{ 'Cookie' : ' UM_distinctid=15ab64ecfd6592-  
0afad5b368bd69-1d3b6853-13c680-15ab64ecfd7b6;  
remember user token=W1sxMjEzMTM3XSwiJDJhJDEwJH  
hjYklYOGI2eTQ0Yi54WC5seVh2UWUiLCIxNDg5ODI2OTgw  
Ljg4ODQyODciXQ%3D%3D---  
ac835770a030c0595b2993289e39c37d82ea27e2;  
CNZZDATA1258679142=559069578-1488626597-  
https%253A%252F%252Fwww.baidu.com%252F%7C1489  
923851' }
```

```
r = requests.get(url, cookies=cookies)
```

```
>>> r.text
```

```
u'[{"repository":{"open_issues":0,"url":"https://github.com/...
```

#返回请求的状态

```
>>>r.status_code    # (4开头客户端问题, 5开头服务器问题)
```

```
200
```


方法	说明
<code>requests.request()</code>	构造一个请求, 支撑以下各方法的基础方法
<code>requests.get()</code>	获取HTML网页的主要方法, 对应于HTTP的GET
<code>requests.head()</code>	获取HTML网页头信息的方法, 对应于HTTP的HEAD
<code>requests.post()</code>	向HTML网页提交POST请求的方法, 对应于HTTP的POST
<code>requests.put()</code>	向HTML网页提交PUT请求的方法, 对应于HTTP的PUT
<code>requests.patch()</code>	向HTML网页提交局部修改请求, 对应于HTTP的PATCH
<code>requests.delete()</code>	向HTML页面提交删除请求, 对应于HTTP的DELETE

更多用法详见:

http://docs.python-requests.org/zh_CN/latest/user/quickstart.html

Part 4

网页解析

- 
- 4.1 如何解析网页
 - 4.2 BeautifulSoup
 - 4.3 xpath
 - 4.4 re正则表达式

使用第三方模块解析并获取标签内的内容

BeautifulSoup

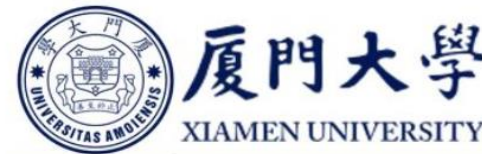
Xpath

可能还会用到正则表达式（参考：<https://docs.python.org/zh-cn/3/library/re.html>）

- BeautifulSoup 是用Python写的一个HTML/XML的解析器，它可以很好的处理不规范标记并生成剖析树(parse tree)。它提供简单又常用的导航（navigating），搜索以及修改剖析树的操作。它可以大大节省你的编程时间。
- 步骤：1、安装beautifulsoup4； 2、安装解释器

解析器	使用方法	优势	劣势
Python标准库	BeautifulSoup(markup, "html.parser")	•Python的内置标准库 •执行速度适中 •文档容错能力强	•Python 2.7.3 or 3.2.2)前的版本中文档容错能力差
lxml HTML 解析器	BeautifulSoup(markup, "lxml")	•速度快 •文档容错能力强	•需要安装C语言库
lxml XML 解析器	BeautifulSoup(markup, ["lxml", "xml"]) BeautifulSoup(markup, "xml")	•速度快 •唯一支持XML的解析器	•需要安装C语言库
html5lib	BeautifulSoup(markup, "html5lib")	•最好的容错性 •以浏览器的方式解析文档 •生成HTML5格式的文档	•速度慢 •不依赖外部扩展

BeautifulSoup--两种主要的对象: Tag、NavigableString



```
html = """
4 <html><head><title>The Dormouse's story</title></head>
5 <body>
6 <p class="title" name="dromouse"><b>The Dormouse's story</b></p>
7 <p class="story">Once upon a time there were three little sisters; and their names were
8 <a href="http://example.com/elsie" class="sister" id="link1"><!-- Elsie --></a>,
9 <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
1 <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
0 and they lived at the bottom of a well.</p>
1 <p class="story">...</p>
1 """
```

```
>>> bsObj = BeautifulSoup(html, "html.parser")
```

格式化输出 bsObj对象的内容

```
>>>bsObj.title
```

```
'<title>The Dormouse's story</title>'
```

```
>>>bsObj.head
```

```
'<head><title>The Dormouse's story</title></head>'
```

```
>>>bsObj.a
```

```
'<a class="sister" href="http://example.com/elsie" id="link1"><!--  
Elsie --></a>'
```

注意：它查找的是在所有内容中的第一个符合要求的标签，
如果要查询所有的标签，这种方法不奏效


```
>>>#直接子节点 .contents
```

```
>>>bsObj.head.contents #输出的为列表
```

```
['<title>The Dormouse's story</title>']
```

```
>>>bsObj.head.contents[0] 从列表中取出子节点
```

```
'<title>The Dormouse's story</title>'
```

BeautifulSoup –遍历文档树



```
>>>#直接子节点 .children
```

```
>>>bsObj.head.children #返回生成器, 可以迭代取出来
```

```
<listiterator object at 0x7f71457f5710>
```

```
>>>for child in soup.body.children:
```

```
>>> print child
```

```
'<p class="title" name="dromouse"><b>The Dormouse's story</b></p>'
```

```
'<p class="story">Once upon a time there were three little sisters; and their names were'
```

```
'<a class="sister" href="http://example.com/elsie" id="link1"><!-- Elsie --></a>'
```

```
'<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and'
```

```
'<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>'
```

```
'and they lived at the bottom of a well.</p><p class="story">...</p>'
```

```
>>>#Attrs    <p class="title" name="dromouse"><b>The Dormouse's  
story</b></p>
```

```
>>>bsObj.p.attrs  
{'class': 'title', 'name': 'dromouse'}
```

```
>>>bsObj.p.attrs['class']  
'title'
```

```
>>>bsObj.p.attrs.get('class')  
'title'
```

>>>#获取标签内部的文字

>>>bsObj.p.string

The Dormouse's story

find_all(name , attrs) 得到的所有符合条件的结果,返回的是列表格式

```
>>>bsObj.findall ( 'a' )
```

```
[<a class= "sister" href= "http://example.com/elsie" id= "link1" ><!-- Elsie --></a>,  
<a class= "sister" href= "http://example.com/lacie" id= "link2" >Lacie</a>,  
<a class= "sister" href= "http://example.com/tillie" id= "link3" >Tillie</a>]
```

```
>>>bsObj.findall( 'a' , { 'href' : 'http://example.com/elsie' })
```

```
[ '<a class= "sister" href= "http://example.com/elsie" id= "link1" ><!-- Elsie --  
></a>' ]
```

- **find(name , attrs)** 只返回第一个符合条件的结果，所以 soup.find()后面可以直接接.text或者get_text()来获得标签中的文本。

```
>>> bsObj.findall( 'a' )
```

```
<a class= "sister" href= "http://example.com/elsie"  
id= "link1" > <!-- Elsie --> </a>
```

- XPath 是一门在 XML 文档中查找信息的语言。XPath 可用在 XML 文档中对元素和属性进行遍历。XPath 是 W3C XSLT 标准的主要元素，并且 XQuery 和 XPointer 都构建于 XPath 表达之上。
- 标签的使用方法
 - 1) // 双斜杠 定位根节点，会对全文进行扫描，在文档中选取所有符合条件的内容，以列表的形式返回。
 - 2) / 单斜杠 寻找当前标签路径的下一层路径标签或者对当前路标签内容进行操作
 - 3) /text() 获取当前路径下的文本内容
 - 4) /@xxxx 提取当前路径下标签的属性值
 - 5) | 可选符 使用|可选取若干个路径 如//p | //div 即在当前路径下选取所有符合条件的p标签和div标签。
 - 6) . 点 用来选取当前节点
 - 7) .. 双点 选取当前节点的父节点

详细用法可参考：

https://blog.csdn.net/qq_36148847/article/details/79167267

```
import requests
from lxml import etree
target_url = 'https://www.qq717.com/html/136/136548/'
base_url = 'https://www.qq717.com'
header = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36 SE 2.X MetaSr 1.0', 'Host': 'www.biquan.com'}
resp = requests.get(target_url, {'headers': header})
html = etree.HTML(resp.text)
href_list = html.xpath('//dd/a/@href')
title_list = html.xpath('//dd/a/text()')
for i in range(len(title_list)):
    url=base_url+href_list[i]
    res=requests.get(url, {'headers':header})
    res.encoding='utf-8'
    html=etree.HTML(res.text)
    title=html.xpath("//div[@class='bookname']/h1/text()")
    line=html.xpath("//div[@id='content']/text()")
    content=''
    for l in line:
        content=content+l+'\n'
    with open('穿梭诸天.txt', 'a', encoding='utf-8') as f:
        print('正在保存第%d章...' %i)
        f.write(title[0])
        f.write(content)
        f.write('\n')
print('下载完成!')
```

etree.HTML()可以用来解析字符串格式的HTML文档对象，将传进去的字符串转变成_Element对象。作为_Element对象，可以方便的使用getparent()、remove()、xpath()等方法。

```
from lxml import etree
html='<html><head><title> 这是一个网页
</title></head><body>
<div id='hello'>Hello World!</div></body></html>'
ele=etree.HTML(html)
Print(ele.xpath('//div[@id="hello"]/text()'))
```


正则表达式（英语：Regular Expression，在代码中常简写为regex、regexp或RE）使用单个字符串来描述、匹配一系列符合某个句法规则的字符串搜索模式。
可用于文本搜索和文本替换。

像这样的火星文就是正则(此为匹配邮箱的正则) =>
`/^[0-9a-z]+@[0-9a-z]+\.(?:com.cn/(?:com/cn))/`

正则表达式在线测试 <https://regex101.com/>

正则表达式框图 <https://regexper.com/>

html = '<html><div>我叫林世明来自来自厦门大学 </div> </html>'
BeautifulSoup获取html文件中的林世明和厦门大学，需要这样

```
>>>bsObj=BeautifulSoup(html, 'html.parser')
```

```
>>>Contents = bsObj.findall('span')
```

```
>>>Contents[0]
```

```
<span>我叫林世明</span>
```

```
>>>Contents[0].string
```

```
我叫林世明
```

```
>>>Contents[0].string[2:]
```

```
林世明
```

re-正则表达式

```
html = '<html><div><span>我叫林世明</span> <span>来自厦门大学  
</span> </div> </html>'
```

```
>>>import re
```

```
>>>pattern = re.compile(r' <span>我叫(. *? )</span> <span>来自  
(. *? )</span> ')
```

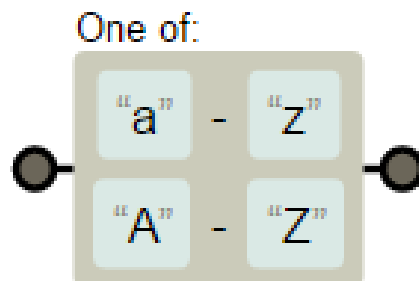
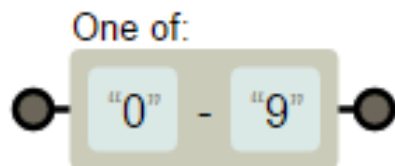
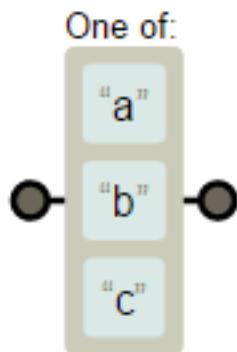
```
>>>result = re.findall(pattern, html)
```

```
>>>result
```

```
('林世明','厦门大学')
```

1、范围类

表达式	描述
[abc]	查找方括号之间的任何字符。
[0-9]	查找任何从 0 至 9 的数字。
[a-zA-z]	匹配所有字母

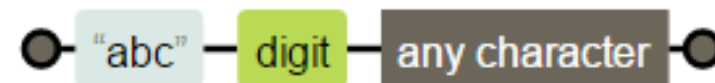


2、预定义类

字符	等价类	含义
.	[^\n\r]	除了回车符和换行符之外的所有字符
\d	[0-9]	数字字符
\D	[^0-9]	非数字字符
\s	[\t\n\x0B\f\r]	空白符
\S	[^\t\n\x0B\f\r]	非空白符
\w	[a-zA-Z_0-9]	单词字符（字母、数字、下划线）
\W	[^a-zA-Z_0-9]	非单词字符

例子：

abc\d.

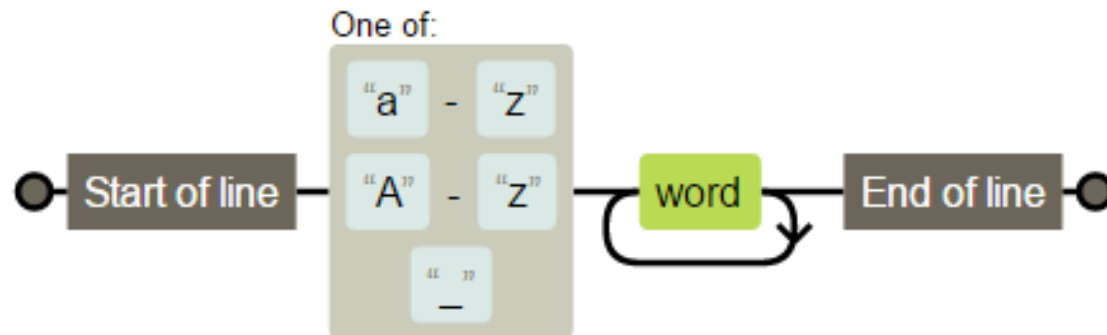


3、边界

字符	含义
^	以xx开头
\$	以xx结尾
\b	单词边界，指[a-zA-Z_0-9]之外的字符
\B	非单词边界

例子：

`^[a-zA-Z_]\w+$`

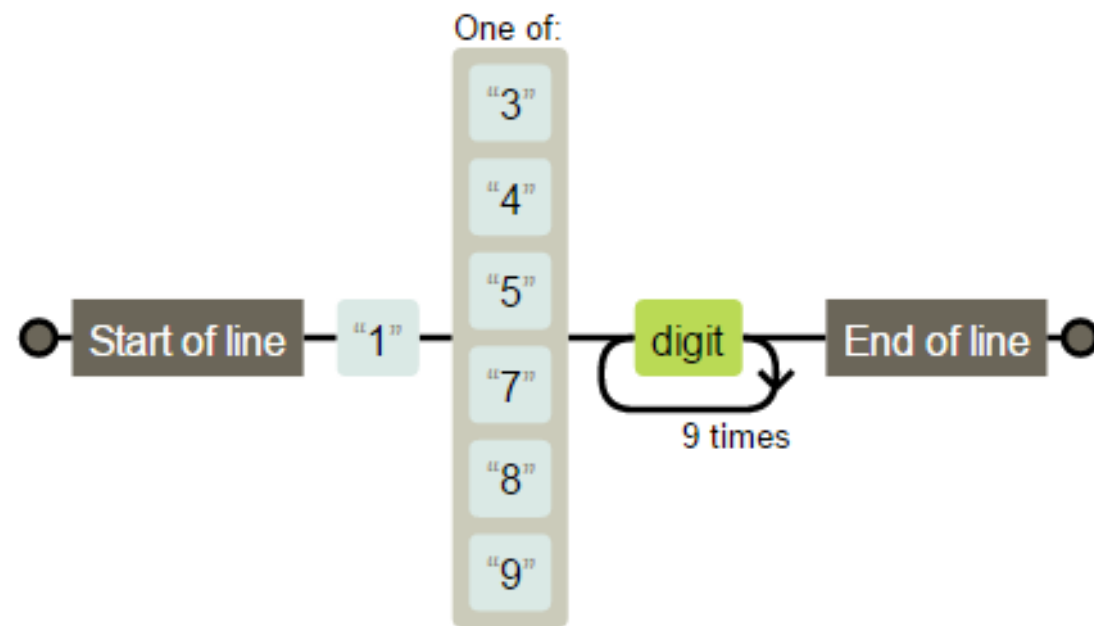


4、量词

正则表达式	描述	举例
*	匹配 ≥ 0 个，是 $\{0,\}$ 的简写	x^* 表示匹配零个或多个字母 x ， $.*$ 表示匹配任何字符串
+	匹配 ≥ 1 个，是 $\{1,\}$ 的简写	x^+ 表示匹配一个或多个字母 x
?	匹配 1 个或 0 个，是 $\{0,1\}$ 的简写	$x^?$ 表示匹配 0 个或 1 个字母 x
$\{x\}$	只匹配 x 个字符	$\{d\{3\}$ 表示匹配 3 个数字， $\{.10\}$ 表示匹配任何长度是 10 的字符串
$\{X,Y\}$	匹配 $\geq X$ 且 $\leq Y$ 个	$\{d\{1,4\}$ 表示匹配至少 1 个最多 4 个数字
$*?$	如果 ? 是限定符 * 或 + 或 ? 或 $\{ \}$ 后面的第一个字符，那么表示非贪婪模式（尽可能少的匹配字符），而不是默认的贪婪模式	

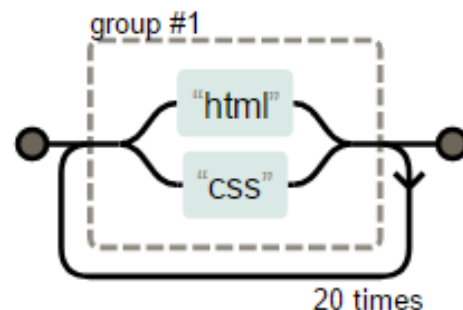
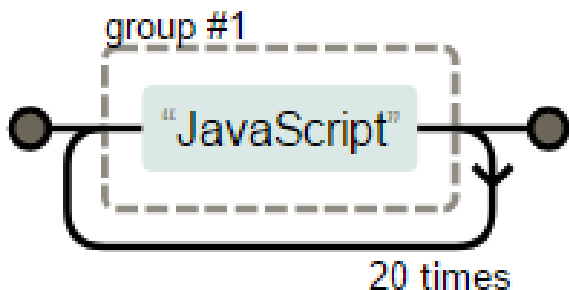
例子：

`^1[345789]\d{9}$`



5、分组

- 有时候我们希望使用量词的时候匹配多个字符，而不是像上面例子只是匹配一个，比如希望匹配JavaScript出现20次的字符串，我们如果写成 `JavaScript{20}` 的话匹配的是JavaScript出现20次，怎么把JavaScript作为一个整体呢？使用`()`就可以达到次目的，我们称为分组。
- 如果希望匹配html或css出现20次该怎么办呢？可以使用字符 `|` 达到或的功效。 `(html|css){20}`



注意区别： `[jpg|png]` 代表匹配 `j` 或 `p` 或 `g` 或 `p` 或 `n` 或 `g` 中的任意一个字符。
`(jpg|png)` 代表匹配 `jpg` 或 `png`。

6、常用方法

pattern = re.compile(string)

#以下为匹配所用函数

re.findall(pattern, **string**) 返回列表

re.match(pattern, **string**) 从string的开头匹配，匹配成功立即返回pattern的内容，
不再匹配string剩余部分

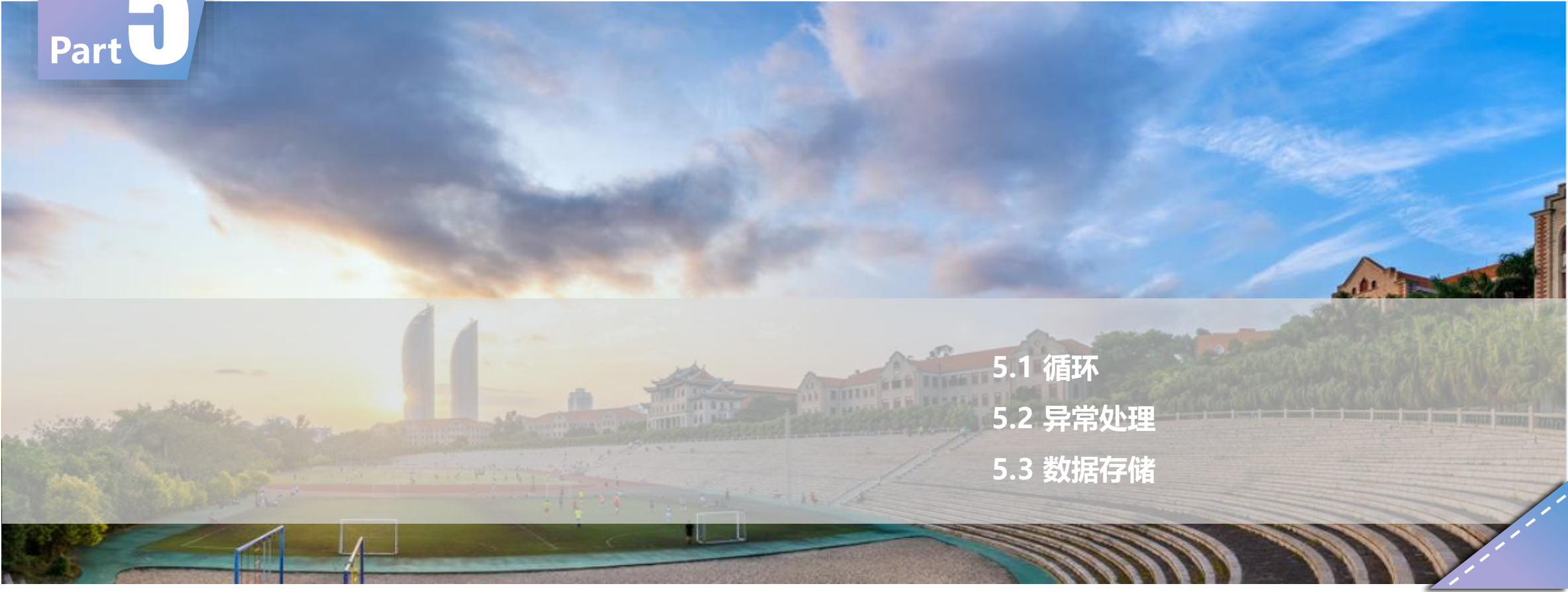
re.search(pattern, **string**) 从string全部匹配，如果匹配不成功返回none，
匹配成功返回一个或多个匹配内容

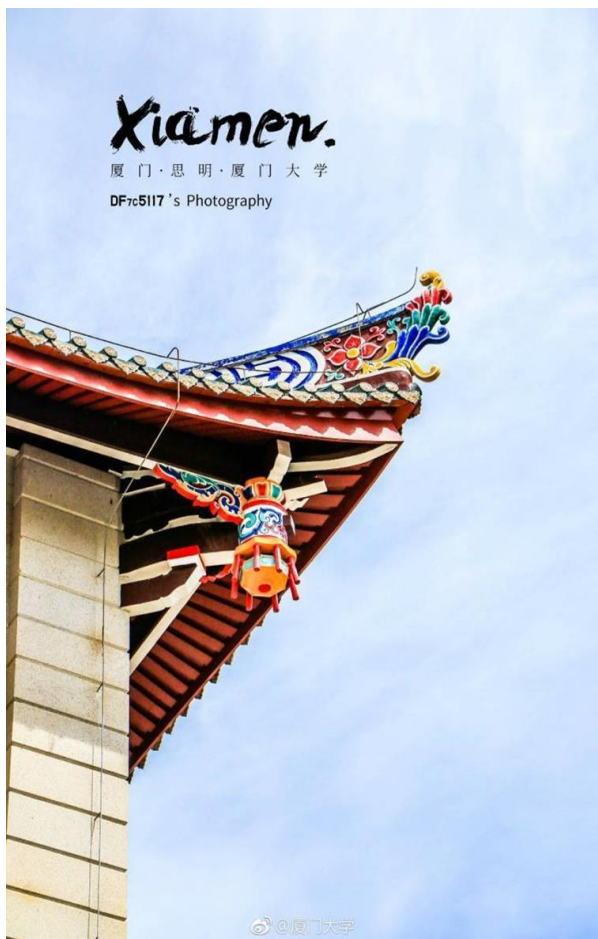
re.split(pattern, **string**)

re.sub(pattern, repl, **string**)

Part 5

采集数据

- 
- 5.1 循环
 - 5.2 异常处理
 - 5.3 数据存储



for 语句

用来循环，重复爬虫动作



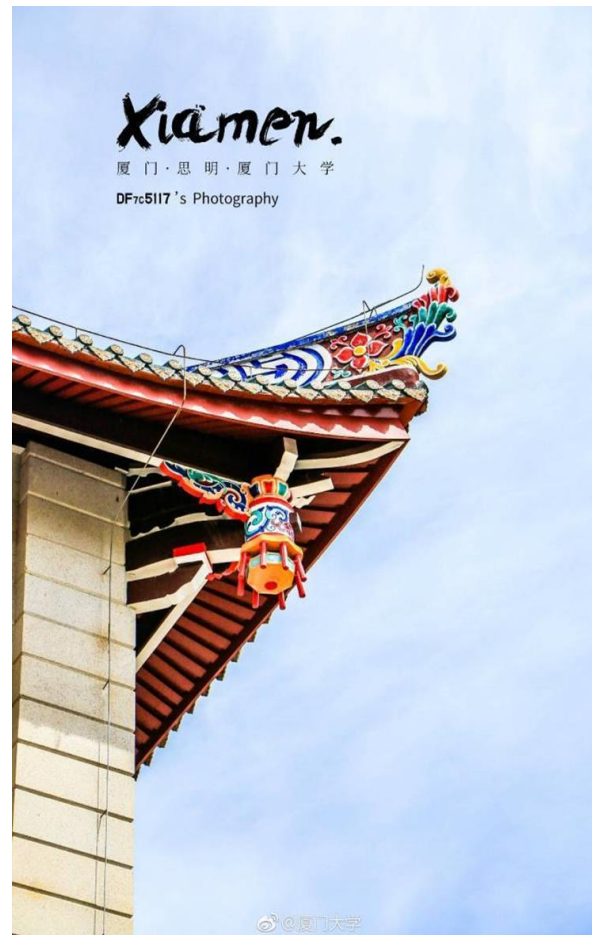
遍历网址，依次对网址发送请求



解析网页时，返回的是列表或生成器的对象，通过for遍历对象中的元素并对每个元素进行操作

if 条件判断语句

用来解决爬虫过程中哪些要，哪些不要的问题
哪些执行，哪些不执行的问题



情景：

你设计的程序在完美情况下当然不会出错，
但现实是经常有意外出现，一遇到问题就程序就退出运行。

解决办法：

try except异常处理，
增强你爬虫的**健壮性**，
解决遇到问题程序停止





利用pandas存储

Pandas, to_csv保存成csv文件



保存成txt文件

可以使用python的文件读写操作,
保存txt文件



保存图像文件

读取文件的二进制流, 保存图像
文件



保存excel文件

可以利用第三方模块xlwd, 将文
件保存成excel格式

Part 6

如何应对反爬



道高一尺魔高一丈！

如何应对反爬



1、伪装成浏览器

```
'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/77.0.3865.120 Safari/537.36',
```


2、控制访问频率

短时间的大流量访问网站，会给服务器带来非常大的负担。网络中最常见的一种攻击方式ddos攻击就是利用这个原理。

因此我们在爬取网站数据的时候，出于人道主义精神，可以把访问的频率降低一些。

可以使用time模块，设置一定的延时

例如：

```
import time
```

```
time.sleep(1) #延时1秒
```

3、使用代理IP

免费代理IP网站: <https://www.xicidaili.com/>

```
import requests
import ssl
from lxml import etree
ssl._create_default_https_context = ssl._create_unverified_context
```

```
#获取当前访问使用的IP地址网站
url="https://www.ipip.net/"
```

```
#设置代理, 从西刺免费代理网站上找出一个可用的代理IP
proxies={'https': '101.236.54.97:8866'} #此处也可以通过列表形式, 设置多个代理IP, 后面通过random.choice()随机选取一个
进行使用
```

```
#使用代理IP进行访问
res=requests.get(url,proxies=proxies,timeout=10)
status=res.status_code # 状态码
print(status)
content=res.text
html=etree.HTML(content)
ip=html.xpath('//ul[@class="inner"]/li[1]/text())[0]
print("当前请求IP地址为: "+ip)
```

- 找一个感兴趣的网站爬取有意思的内容并保存下来;
- 也可以任意爬取<https://www.qq717.com/>上的一本小说。

Homework

- 爬取安居客房源信息;
- 链接地址: <https://qy.lianjia.com/ershoufang/>

Tips: 需要用到的模块urllib / requests、BeautifulSoup、lxml、pandas(写csv)、time (设置定时, 避免速度太快被服务器拒绝) 等。

要求: 1、命名要规范, 采用驼峰式命名, 或var_word这种命名方式;
2、必要的地方需要添加注释, 比如类或者函数需要添加参数的相关注释。

结束语

网络给我们工作、学习、生活提供了极大的便利，可是也正是因为网络的飞速发展，相关法律法规已严重滞后。有人说，网络时代人人都是在“裸奔”，你已毫无隐私可言。这几年爆发出不少搜集用户隐私的事件，例如著名的“棱镜门”、facebook售卖用户隐私等。网络时代亟需底线意识，和法律、道德约束。



本教程仅供交流、学习、研究之用，不承担任何法律责任。请同学们自觉遵守互联网有关法律法规，坚守道德底线，做一个文明的“爬手”。

最重要的一件事，如果被抓了，千万别说是我教的。😁



Thank You !