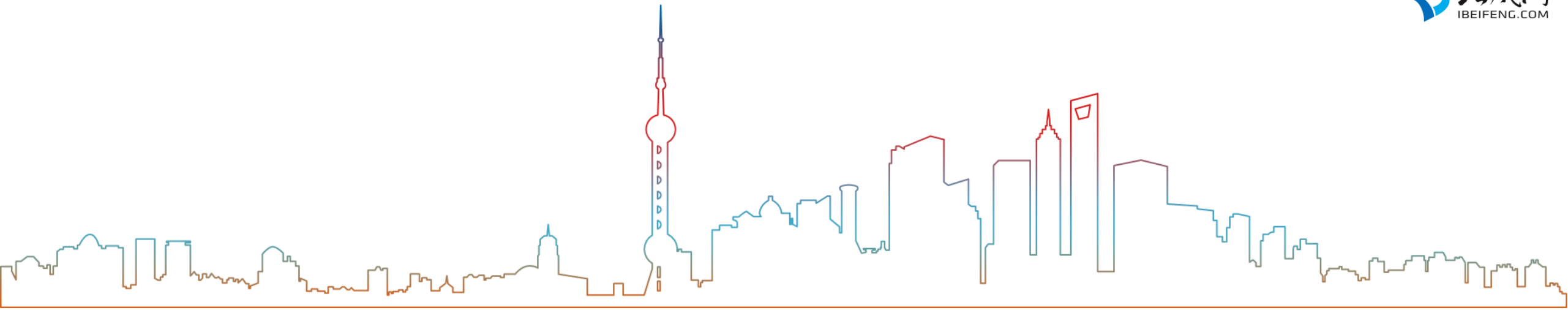
跟我学系列——走进Scrapy爬虫

爬虫利器

上海育创网络科技股份有限公司

主讲人：子沐



导 入

基础爬虫库我们已经掌握了，但是这并不能满足我们对于数据完美的抓取，那么现在我们就来说一说Python爬虫利器，教大家如何能真正的爬取到数据。

课程介绍 - 课程体系

1

Requests

之前我们用了 urllib 库，这个作为入门的工具还是不错的，对了解一些爬虫的基本理念，掌握爬虫爬取的流程有所帮助。入门之后，我们就需要学习一些更加高级的内容和工具来方便我们的爬取。

2

正则表达式

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。

3

Beautiful Soup

Beautiful Soup 提供一些简单的、python 式的函数用来处理导航、搜索、修改分析树等功能。它是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。

4

lxml

Lxml是一种使用 Python 编写的库，可以迅速、灵活地处理 XML

本章任务

01 熟练掌握Requests库安装配置及用法

02 开启Beautiful Soup之旅

03 熟练掌握Xpath语法与lxml库的安装配置及用法。

01 Requests

本章内容

我们为什么要学习Requests？

- Requests安装
- 基本请求（GET/POST）
- 会话（Cookies/超时配置/会话对象）
- 代理及SSL验证

本章内容

Requests安装

- pip和easy_install万能包管理器
  - pip install requests
  - easy_install requests

```
G:\>pip install requests
Collecting requests
  Downloading requests-2.13.0-py2.py3-none-any.whl (584kB)
    100% |████████████████████████████████| 593kB 43kB/s
Installing collected packages: requests
Successfully installed requests-2.13.0
```

本章内容

基本请求（GET/POST）

- 基本Get请求
  - requests.get(url, params=None, **kwargs)
- 基本POST请求
  - requests.post(url, data=None, json=None, **kwargs)
- kwargs
  - headers（头文件信息）
  - cookies（cookie包）
  - files（文件）
  - auth（HTTP身份验证）
  - timeout（超时设置）
  - allow_redirects（是否启用重定向）
  - proxies（协议代理）
  - verify（SSL证书验证）
  - stream（流）
  - cert（ssl证书文件路径）

本章内容

基本请求（GET/POST）

查看基本返回信息

```
import requests
'''  '''
#通过URL发送get请求 ， 获取网页信息
r = requests.get('http://www.ibeifeng.com')
#查看返回类型
print(type(r))
#查看状态码
print(r.status_code)
#查看编码
print(r.encoding)
#查看返回的cookie
print(r.cookies)
#字符串方式的响应体，会自动根据响应头部的字符编码进行解码
print(r.text)
#字节方式的响应体，会自动为你解码 gzip 和 deflate 压缩
print(r.content.decode('gbk'))
```

```
requests.get('http://httpbin.org/get') #GET请求
requests.post('http://httpbin.org/post') #POST请求
requests.put('http://httpbin.org/put') #PUT请求
requests.delete('http://httpbin.org/delete') #DELETE请求
```

本章内容

基本请求（GET/POST）

使用GET传递参数

```
import requests
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.get("http://httpbin.org/get", params=payload)
print(r.url)

#http://httpbin.org/get?key1=value1&key2=value2
```

本章内容

基本请求（GET/POST）

GET请求添加头文件伪装浏览器

```
import requests

payload = {'key1': 'value1', 'key2': 'value2'}
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKi
}
r = requests.get("http://httpbin.org/get", params=payload,headers = headers)
print(r.text)
print(r.url)
```

伪装浏览器访问知乎网站

```
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) A
}
response = requests.get("https://www.zhihu.com",headers = headers)
print(response.text)
```

本章内容

基本请求（GET/POST）

GET请求，解析json数据

```
import requests

#json响应内容
#Requests 中也有一个内置的 JSON 解码器，助你处理 JSON 数据：
r = requests.get("https://github.com/timeline.json")
print(r.text)
print(r.json())

#请求本地服务器
r = requests.get("http://www.mywebs.com/re_json")
print(r.text)
print(r.json())
```

本章内容

基本请求（GET/POST）

GET请求，获取原始响应内容

```
'''
原始响应内容
在罕见的情况下，你可能想获取来自服务器的原始套接字响应，
那么你可以访问 r.raw。 如果你确实想这么干，
那请你确保在初始请求中设置了 stream=True。
'''
r = requests.get('https://github.com/timeline.json', stream=True)
print(r.raw.read(100))
```

```
#下载音乐
r = requests.get('音乐地址',stream = True)
# print(r.content)
#大的流文件可以按大小去读取
with open('4.mp3','wb') as file:
    for chunk in r.iter_content(1024*10):
        file.write(chunk)
```

```
# 下载图片
r = requests.get('图片地址',stream = True)
# print(r.content)
# 保存图片
with open('1.jpg','wb') as f:
    for chunk in r.iter_content(1024 * 10):
        f.write(chunk)
```

本章内容

基本请求（GET/POST）

GET请求，获取网页COOKIE

```
import requests
#获取网页COOKIE
url = 'http://www.ibeifeng.com'
r = requests.get(url)
print(r.cookies)
print(r.cookies['ECS[visit_times]'])
```

```
#设置cookie  1.开启session
s = requests.session()
#2.检查有没有cookie
r = s.get('http://www.mywebs.com/cookie')
print(r.text)

#3.我到服务器的setcookIE这里要了一个cookie
r = s.get('http://www.mywebs.com/setcookie')
print(r.text)
print(r.cookies)

#4.检查有没有cookie
r = s.get('http://www.mywebs.com/cookie')
print(r.text)
```

```
#请求测试网站发送cookie
r = requests.get('http://httpbin.org/cookies',cookies = {'name':'joe'})
print(r.text)
```

本章内容

基本请求（GET/POST）

Post请求，传递参数

```
#访问测试地址
#使用data参数传递post参数
datas = {'name':'joe','pwd':'123'}
r = requests.post('http://httpbin.org/post',data = datas)
print(r.text)

#访问本地服务器
r = requests.post('http://www.mywebs.com/forpost',data = datas)
print(r.text)
```

本章内容

基本请求（GET/POST）

Post请求，发送cookie

```
import requests
#post请求添加cookie
url = 'http://httpbin.org/post'

#想本地服务器发送测试
# url = 'http://www.mywebs.com/cookie'
cookies = dict(cookies_are='working')
r = requests.post(url, cookies=cookies)#设置cookie
print(r.cookies)
print(r.text)
```

本章内容

基本请求（GET/POST）

Post请求，发送文件

```
import requests
#POST发送文件
url = 'http://httpbin.org/post'
files = {'file': open('post_file.txt','rb')}
r = requests.post(url, files=files)
print(r.text)
```

Post请求，发送json数据

```
import json
import requests

#向httpbin 发送json数据
# url = 'http://httpbin.org/post'
url = 'http://www.mywebs.com/re_json'
payload = {'some':'data'}
r = requests.post(url,data=json.dumps(payload))
print(r.text)
```

本章内容

基本请求（GET/POST）

Session会话

```
#session会话
s = requests.session()  #开启了session
#
header = {
    'cookie':'one',
    "User-Agent":'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
}
#
r = s.get('http://httpbin.org/get',headers = header )
print(r.text)
#
s.headers.update(header)
r = s.get('http://httpbin.org/get',headers ={'cookes':'two'})
print(r.text)
```

本章内容

基本请求（GET/POST）

Session会话

```
#session会话  03访问本地模拟环境
r = requests.get("http://www.mywebs.com/setcookie")
print(r.cookies)
r = requests.get("http://www.mywebs.com/cookie")
print(r.text)

s = requests.Session()
r = s.get("http://www.mywebs.com/setcookie")
print(r.cookies)
r = s.get("http://www.mywebs.com/cookie")
print(r.text)
```

本章内容

基本请求（GET/POST）

session模拟登陆

```
import json
s = requests.session()
r = s.post('http://123.207.11.209:8080/index.php/Home/Index/checkLogin', data =
{'admin_name':'root_python','admin_password':'pythonroot'} ,headers={'x-test2': 'true'})
print(json.loads(r.content))

r = s.post('http://123.207.11.209:8080/index.php/Home/System/aboutme', headers={'x-
test2': 'true'})
print(r.text)
```

本章内容

基本请求（GET/POST）

import requests

超时

你可以告诉 requests 在经过以 timeout 参数设定的秒数时间之后停止等待响应。

基本上所有的生产代码都应该使用这一参数。如果不使用，你的程序可能会永远失去响应：

r = requests.get('http://github.com', timeout=0.1)

print(r.text)

```
from requests.exceptions import ConnectTimeout,ConnectionError,RequestException
try:
    r = requests.get('http://github.com', timeout=0.1)
    print(r.status_code)

except ConnectTimeout:
    print('Time out')
except ConnectionError:
    print('Connect error')
except RequestException as e:
    print(e)
```

本章内容

基本请求（GET/POST）

根据协议类型，选择不同的代理

```
proxies = {
   "http": "http://139.196.122.166:8080",
   # "https": "http://118.193.19.158:8080",
}
headers = {
   'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:56.0) Gecko/20100101 Firefox/56.0'
}
# response = requests.get("http://httpbin.org/get", proxies = proxies)
response = requests.get("http://www.baidu.com/", proxies = proxies,headers = headers)
print(response.text)

response = requests.get("https://www.taobao.com", proxies=proxies)
print(response.status_code)
print(response.text)
```

本章内容

基本请求（GET/POST）

SSL验证

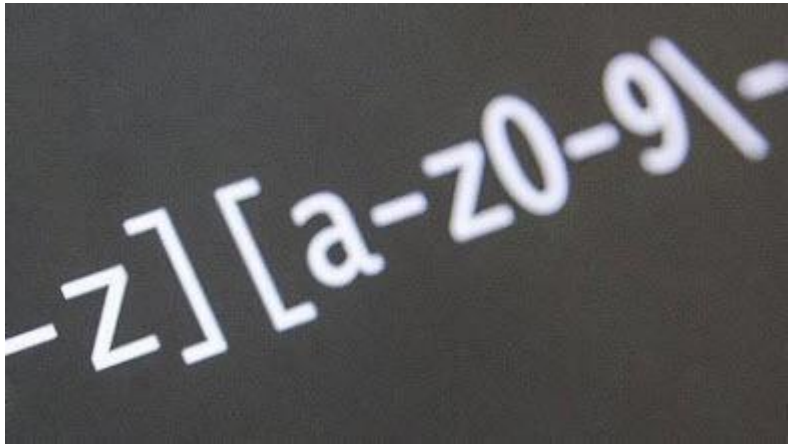
```
#SSL验证
import requests

response = requests.get('https://www.12306.cn', verify=False)
print(response.status_code)

print(response.content.decode('UTF-8'))
```

02 正则表达式

本章内容

正则表达式

- 了解正则表达式
- 正则表达式的语法规则
- 数量词的贪婪模式与非贪婪模式
- 反斜杠问题
- 使用Python Re 模块操作正则表达式



本章内容

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。 正则表达式是用来匹配字符串非常强大的工具，在其他编程语言中同样有正则表达式的概念，Python同样不例外，利用了正则表达式，我们想要从返回的页面内容提取出我们想要的内容就易如反掌了。

语法	说明	表达式实例	完整匹配的字符串
字符			
一般字符	匹配自身	abc	abc
.	匹配任意除换行符"\n"外的字符。 在DOTALL模式中也能匹配换行符。	a.c	abc
\	转义字符，使后一个字符改变原来的意思。 如果字符串中有字符*需要匹配，可以使用*或者字符集[*]。	a\.c a\\c	a.c a\c
[...]	字符集（字符类）。对应的位置可以是字符集中任意字符。 字符集中的字符可以逐个列出，也可以给出范围，如[abc]或[a-c]。第一个字符如果是^则表示取反，如[^abc]表示不是abc的其他字符。 所有的特殊字符在字符集中都失去其原有的特殊含义。在字符集中如果要使用]、-或^，可以在前面加上反斜杠，或把]、-放在第一个字符，把^放在非第一个字符。	a[bcd]e	abe ace ade

本章内容

预定义字符集（可以写在字符集[...]中）			
\d	数字：[0-9]	a\dc	a1c
\D	非数字：[^\d]	a\Dc	abc
\s	空白字符：[<空格>\t\r\n\f\v]	a\sc	a c
\S	非空白字符：[^\s]	a\Sc	abc
\w	单词字符：[A-Za-z0-9_]	a\wc	abc
\W	非单词字符：[^\w]	a\Wc	a c
数量词（用在字符或(...)之后）			
*	匹配前一个字符0或无限次。	abc*	ab abccc
+	匹配前一个字符1次或无限次。	abc+	abc abccc
?	匹配前一个字符0次或1次。	abc?	ab abc
{m}	匹配前一个字符m次。	ab{2}c	abbc
{m,n}	匹配前一个字符m至n次。 m和n可以省略：若省略m，则匹配0至n次；若省略n，则匹配m至无限次。	ab{1,2}c	abc abbc
*? +? ?? {m,n}?	使 * + ? {m,n}变成非贪婪模式。	示例将在下文中介绍。	

本章内容

边界匹配（不消耗待匹配字符串中的字符）			
^	匹配字符串开头。 在多行模式中匹配每一行的开头。	^abc	abc
$	匹配字符串末尾。 在多行模式中匹配每一行的末尾。	abc$	abc
\A	仅匹配字符串开头。	\Aabc	abc
\Z	仅匹配字符串末尾。	abc\Z	abc
\b	匹配\w和\W之间。	a\b!bc	a!bc
\B	[^\b]	a\Bbc	abc
逻辑、分组			
|	|代表左右表达式任意匹配一个。 它总是先尝试匹配左边的表达式，一旦成功匹配则跳过匹配右边的表达式。 如果|没有被包括在()中，则它的范围是整个正则表达式。	abc|def	abc def
(...)	被括起来的表达式将作为分组，从表达式左边开始每遇到一个分组的左括号'('，编号+1。 另外，分组表达式作为一个整体，可以后接数量词。表达式中的|仅在该组中有效。	(abc){2} a(123|456)c	abcabc a456c
(?P<name>...)	分组，除了原有的编号外再指定一个额外的别名。	(?P<id>abc){2}	abcabc
\<number>	引用编号为<number>的分组匹配到的字符串。	(\d)abc\1	1abc1 5abc5
(?P=name)	引用别名为<name>的分组匹配到的字符串。	(?P<id>\d)abc(?P=id)	1abc1 5abc5

本章内容

◆ compile()

编译正则表达式模式，返回一个对象的模式。

```
re = re.compile('\w{1}\w')
print(type(re))
```

◆ match()

从字符串的开头进行匹配， 匹配成功就返回一个匹配对象，匹配失败就返回None

```
#2.match匹配开头
print(re.match('hello','hello haha').group()) #返回匹配的结果
```

```
pattern = re.compile('(\w+) (\w+)')
str1 = 'hello worald,hahahaahha efsfsd dsfdsfd'
result = pattern.match(str1)
print(result)  #返回匹配结果match对象
print(result.group(0))  #返回匹配成功的完整的子串
print(result.group(1))
print(result.group(2))
# print(result.group(3))
```

本章内容

- serach()

函数会在字符串内查找模式匹配,只要找到第一个匹配然后返回，如果字符串没有匹配，则返回None

```
res = re.compile('[a-zA-Z]{1}')
strs = '123abc456'
print(re.search(res,strs).group())
```

search 方法与 match 方法极其类似，区别在于 match() 函数只检测 re 是不是在 string的开始位置匹配，search() 会扫描整个 string 查找匹配，match（ ）只有在0位置匹配成功的话才有返回，如果不是开始位置匹配成功的话，match() 就返回 None。同样，search 方法的返回对象同样 match() 返回对象的方法和属性

```
pattern = re.compile('\d{2}')  #
str1 = 'hellod12worald, 11'
result = pattern.search(str1)  #只匹配一次
print(result)#返回match对象
print(result.group())  #使用group查看返回结果
'''
    search 与 match 的区别:
    match: 只匹配开头部分，查看字符串的开头是否符合条件
    search:会在整个字符串中匹配，但是，只匹配一次，匹配成功之后就不继续往下匹配了
'''
```

本章内容

- findall()遍历匹配，可以获取字符串中所有匹配的字符串，返回一个列表

```
pa = re.compile('[a-zA-Z]{1}')
strs = '123abc456'
print(re.findall(pa,strs))
```

Ps:遇上分组的时候只返回分组匹配的结果

- finditer()返回一个顺序访问每一个匹配结果（Match对象）的迭代器。找到 RE 匹配的所有子串，并把它们作为一个迭代器返回。

```
pa = re.compile('[a-zA-Z]{1}')
strs = '123abc456'
print(re.finditer(pa,strs))
```

```
pattern = re.compile('([a-z])[a-z]([a-z])')  #任意长度匹配英文字母
str1 = '123abc456def789'    #会匹配整个字符串，将符合规则的保存在列表中
result = pattern.finditer(str1)   #返回一个迭代器
# print(result)
for i in result:
    print(i.group(0))  #返回一个match,使用group返回完整的匹配结果
    print(i.group(1))   #返回第一个分组匹配的结果
    print(i.group(2))   #返回第二个分组匹配的结果
```

- Split()按照能够匹配的子串将string分割后返回列表。

```
print(re.split('\d+','one11two2three3four4five5'))
print(re.split('\W+','one,2two,three,four,five,'))
```

本章内容

- sub()使用re替换string中每一个匹配的子串后返回替换后的字符串

```
#sub使用re替换string中每一个匹配的子串后返回替换后的字符串
print(re.sub('\d+','-','one11two2three3four4five5'))
```

- subn ()返回替换后的记过和替换次数

```
#subn 返回替换次数
print(re.subn('\d+','-','one11two2three3four4five5'))
```

- 引用分组　\1表示第一个分组　\2表示第二个分组

```
strs = 'hello 123,world 321'
pattern = re.compile('(\w+) (\d+)')
for i in pattern.finditer(strs):
    print(i.group(0))
    print(i.group(1))
    print(i.group(2))
print(pattern.sub(r'\2 \1',strs))  #取得分组  \1  \2
```

本章内容

贪婪与非贪婪

贪婪：再整个表达式匹配成功的前提下，尽可能多的匹配

非贪婪：在整个表达式匹配成功的前提下，尽可能少的匹配

```
str1 = 'aaa<p>hello</p>bbb<p>world</p>ccc'
pattern = re.compile('<p>.*</p>')  #.匹配任意字符*0-无限次
print(pattern.findall(str1))       #贪婪模式

pattern = re.compile('<p>.*?</p>')  #.匹配任意字符*0-无限次
print(pattern.findall(str1))       #非贪婪 尽可能少的匹配，遇到结束的标签则结束
```

◆ 匹配中文字符

```
str1 = '你好,hello,帅哥'
pattern = re.compile('\w+')
print(pattern.findall(str1))

pattern = re.compile('[\u4e00-\u9fa5]+')  #匹配中文字符
print(pattern.findall(str1))
```

本章内容

练习

1.将以下字符串中所有的url匹配出来

```
str = '''313156566@qq.com

    hjasd23@163.com

    http://www.abc.com.cn

    https://www.sae.com

    ftp://www.nnn.org

    ftps://www.jksad.net'''
```

2. 判断以下字符是否全是中文(中文正则模式[\u4E00-\u9FA5])

```
str='广东省广州市';
```

本章内容

练习

3.写出一个正则表达式，过虑网页上的所有JS脚本(即把script标记及其内容都去掉)

script="以下内容不显示：<script language='javascript'>alert('cc');</script>

fdgdfgdgsdg

<script>alert('dd');</script> “

4. 通过正则表达式把img标签中的src路径匹配出来

str='''

'''

5.将电话号码13811119999变成138****9999

03 开启Beautiful Soup 之旅

本章内容

Beautiful Soup是什么？

- Beautiful Soup提供一些简单的、python式的函数用来处理导航、搜索、修改分析树等功能。
- Beautiful Soup自动将输入文档转换为Unicode编码，输出文档转换为utf-8编码。
- Beautiful Soup已成为和lxml、html6lib一样出色的python解释器，为用户灵活地提供不同的解析策略或强劲的速度。

本章内容

怎么创建Beautiful Soup？

- 怎么创建Beautiful Soup？
- 创建BeautifulSoup对象

```
from bs4 import BeautifulSoup
import requests
html=requests.get('http://www.baidu.com').content.decode('utf-8')
#创建bs对象
soup=BeautifulSoup(html,'html.parser')
html = soup.prettify()#格式
#保存文件
with open('baidus.html','a+',encoding='UTF-8') as file:
    file.write(html)
print(html)
print(type(soup))
```

```
from bs4 import BeautifulSoup
import requests
soup=BeautifulSoup(open('baidu.html',encoding='utf-8'),'html.parser')
```

04 Beautiful Soup节点遍历

本章内容
四大对象种类？

- Tag：通俗点讲就是HTML中的标签 如(title div)对于Tag有两个重要的属性

  Name:返回标签名称

  Attrs:返回标签属性

- NavigableString

  通过string获取标签里的内容。

  Strings：获取多个内容，不过需要遍历获取

- BeautifulSoup

  BeautifulSoup 对象表示的是一个文档的内容。大部分时候,可以把它当作 Tag 对象，是一个特殊的 Tag，我们可以分别获取它的类型，名称，以及属性

- Comment

  Comment 对象是一个特殊类型的 NavigableString 对象，其输出的内容不包括注释符号。

本章内容

如何遍历文档树？

- 直接子节点

  .contents：标签的 .content 属性可以将tag的子节点以列表的方式输出

  .children：返回一个可迭代对象

- 所有子孙节点

  .descendants 属性可以对所有tag的子孙节点进行递归循环，和 children类似，我们也需要遍历获取其中的内容。

- 节点内容

  .string：返回标签里面的内容

  .text:返回标签的文本

本章内容

如何遍历文档树？

- 父节点

  Parent:获取当前节点的父节点

- 全部父节点

  Parents:获取当前节点的所有父节点

本章内容

如何遍历文档树？

◆ 兄弟节点

兄弟节点可以理解为和本节点处在统一级的节点，.next_sibling 属性获取了该节点的下一个兄弟节点，.previous_sibling 则与之相反，如果节点不存在，则返回 None

◆ 前后节点

next_element：与 .next_sibling .previous_sibling 不同，它并不是针对于兄弟节点，而是在所有节点，不分层次

本章内容

如何搜索文档树？

- find_all(name , attrs , recursive , text , **kwargs)
- 1.传字符串

```
print(soup.find_all('p'))

print(soup.find_all(name = 'p'))
```

- 2.传正则

```
print(soup.find_all(href=re.compile('www.*')))

print(soup.find_all(re.compile('^a')))
```

- 3.如果传入列表参数,Beautiful Soup会将与列表中任一元素匹配的内容返回

```
print(soup.find_all(['a','p']))
```

本章内容

如何搜索文档树？

- 4. keyword 参数 (name,attrs)

```
print(soup.find_all(id='p1'))
print(soup.find_all(name = 'p'))
print(soup.find_all(name='p',attrs={'name':2}))
```

- 5.通过 text 参数可以搜搜文档中的字符串内容

```
print(soup.find_all(text = '百度'))
```

- 6. 限定查找个数

```
print(soup.find_all(href=re.compile('www.*'),limit=1))
```

rescurive 参数，默认find_all会搜索所有的子孙节点，
rescurive设置为false，得到就是直接子节点

本章内容

如何搜索文档树？

- find(name , attrs , recursive , text , **kwargs)

结合节点操作

```
print(soup.find(id='head').div.div.next_sibling.next_sibling)
```

- find_parents() find_parent()
- find_next_siblings() find_next_sibling()
- find_previous_siblings() find_previous_sibling()
- find_all_next() find_next()
- find_all_previous() 和 find_previous()

本章内容

CSS选择器

这就是另一种与 find_all 方法有异曲同工之妙的查找方法.

写 CSS 时，标签名不加任何修饰，类名前加.，id名前加#

在这里我们也可以利用类似的方法来筛选元素，用到的方法是 soup.select()，返回类型是 list。http://www.w3school.com.cn/cssref/css_selectors.asp

- 通过标签名查找:select('p')
- 通过类名查找:select('.menu')
- 通过 id 名查找:select('#link')
- 属性查找:select('p[name=2]')
- 组合查找:select('div #p2')
- 获取内容：get_text()

```
#组合查找
print(soup.select("div[id='menu'] + .clearn + #content a"))

# 选择所有p标签中的第三个标签
print(soup.select("p:nth-of-type(2)"))

# 选择body标签下的所有a标签
print(soup.select("#content a"))

# 选择body标签下的直接a子标签
print(soup.select("#content > a"))

# 选择id=link1后的所有兄弟节点标签
soup.select("#link1 ~ .mysis")

# 选择id=link1后的下一个兄弟节点标签
soup.select("#link1 + .mysis")
```

05 lxml用法

本章内容

如何安装lxml

- xml的详细介绍，官网链接：http://lxml.de/，是一种使用 Python 编写的库，可以迅速、灵活地处理 XML

- 安装

```
>pip install lxml
```

本章内容

lxml用法

- 初步使用
- 文件读取

```
from lxml import etree
text = '''
<div>
    <ul>
         <li class="item-0"><a href="link1.html">first item</a></li>
         <li class="item-1"><a href="link2.html">second item</a></li>
         <li class="item-inactive"><a href="link3.html">third item</a></li>
         <li class="item-1"><a href="link4.html">fourth item</a></li>
         <li class="item-0"><a href="link5.html">fifth item</a>
    </ul>
 </div>
'''
html = etree.HTML(text)
result = etree.tostring(html)
print(result)
```

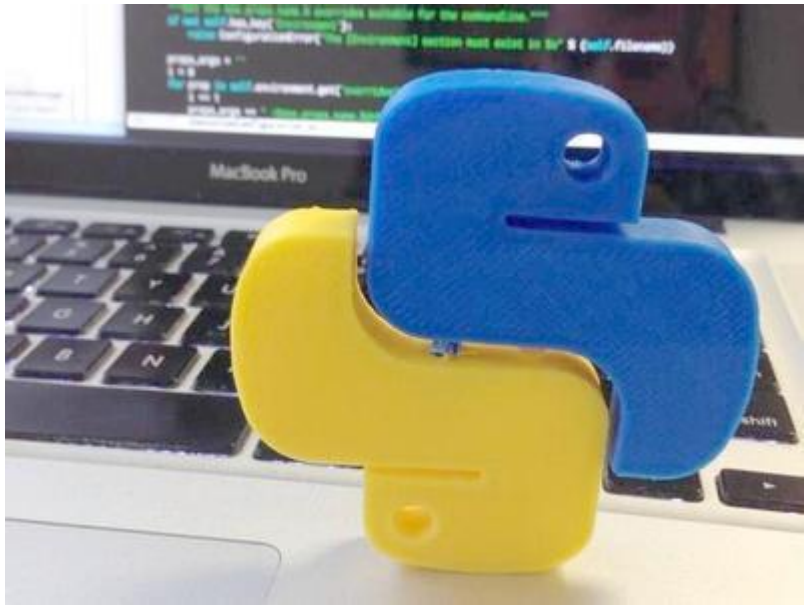
06 Xpath语法

本章内容

Xpath是什么鬼

XPath (XML Path Language) 是一门在 XML 文档中查找信息的语言，可用来在 XML 文档中对元素和属性进行遍历。

- 节点关系
- 选取节点
- 谓语
- 选取未知节点
- Xpath运算符



本章内容

创建lxml对象

```
from lxml import etree

#1.打开文件解析为html文档
html = etree.HTML(open('web.html',encoding='UTF-8').read())  #会对标签自动补全
# html = etree.parse(open('web.html',encoding='UTF-8'))   #严格遵守w3c规范
# html = etree.fromstring(open('web.html',encoding='UTF-8').read())  #严格遵守w3c规范
# print(html)

#2.将html文档字符串序列化
result = etree.tostring(html,pretty_print=True, encoding='utf-8').decode('utf-8')
print(result)
```

本章内容

- 选取节点

表达式	描述
nodename	选取此节点的所有子节点。
/	从根节点选取。
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
.	选取当前节点。
..	选取当前节点的父节点。
@	选取属性。

本章内容

◆ 谓语

谓语用来查找某个特定的节点或者包含某个指定的值的节点。
谓语被嵌在方括号中。

路径表达式	结果
/bookstore/book[1]	选取属于 bookstore 子元素的第一个 book 元素。
/bookstore/book[last()]	选取属于 bookstore 子元素的最后一个 book 元素。
/bookstore/book[last()-1]	选取属于 bookstore 子元素的倒数第二个 book 元素。
/bookstore/book[position()<3]	选取最前面的两个属于 bookstore 元素的子元素的 book 元素。
//title[@lang]	选取所有拥有名为 lang 的属性的 title 元素。
//title[@lang='eng']	选取所有 title 元素，且这些元素拥有值为 eng 的 lang 属性。
/bookstore/book[price>35.00]	选取 bookstore 元素的所有 book 元素，且其中的 price 元素的值须大于 35.00。
/bookstore/book[price>35.00]/title	选取 bookstore 元素中的 book 元素的所有 title 元素，且其中的 price 元素的值须大于 35.00。

本章内容

- 选取未知节点

XPath 通配符可用来选取未知的 XML 元素。

通配符	描述
*	匹配任何元素节点。
@*	匹配任何属性节点。
node()	匹配任何类型的节点。

本章内容

◆ Xpath运算符

|	计算两个节点集	//book | //cd	返回所有拥有 book 和 cd 元素的节点集
+	加法	6 + 4	10
-	减法	6 - 4	2
*	乘法	6 * 4	24
div	除法	8 div 4	2
=	等于	price=9.80	如果 price 是 9.80，则返回 true。 如果 price 是 9.90，则返回 false。
!=	不等于	price!=9.80	如果 price 是 9.90，则返回 true。 如果 price 是 9.80，则返回 false。
<	小于	price<9.80	如果 price 是 9.00，则返回 true。 如果 price 是 9.90，则返回 false。
<=	小于或等于	price<=9.80	如果 price 是 9.00，则返回 true。

本章总结

- 了解了如何使用Request爬取网页
- 掌握了Beautiful Soup的基本语法
- 掌握了lxml用法及Xpath基本语法

THANKS !

上海育创网络科技股份有限公司

主讲人：子沐老师