

定向爬虫

requests+bs4+re

<http://github.com/yihongfa/socialmining>



Requests

自动爬取HTML页面

自动网络请求提交



Beautiful Soup

解析HTML页面



Re

正则表达式

提取页面关键信息

Requests

1.1 Requests测试

```
>>> import requests  
>>> r = requests.get("http://www.baidu.com")  
>>> print(r.status_code)  
200  
>>> r.text  
'<!DOCTYPE html>\r\n<!--STATUS OK--><html> <head><meta http-equiv="text/html; charset=utf-8"><meta http-equiv=X-UA-Compatible content=t=always name=referrer><link rel=stylesheet type=text/css href=r/www/cache/bdorz/baidu.min.css><title>ç\x99%å°¡ä¸\x80ä¸\x8bi%\x8\x93</title></head> <body link=#0000cc> <div id=wrapper> <div id=
```

1.2 Requests库的7个主要方法

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

1.3 requests.get()

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

返回一个包含服务器
资源的Response对象

构造一个向服务器请求
资源的Request对象

Response

Request

```
requests.get(url, params=None, **kwargs)
```

- **url** : 拟获取页面的url链接
- **params** : url中的额外参数，字典或字节流格式，可选
- ****kwargs**: 12个控制访问的参数

Response对象

```
>>> import requests  
>>> r = requests.get("http://www.baidu.com")  
>>> print(r.status_code)  
200  
>>> type(r)  
<class 'requests.models.Response'>  
>>> r.headers  
{'Cache-Control': 'private, no-cache, no-store, proxy-revalidate,  
ection': 'Keep-Alive', 'Transfer-Encoding': 'chunked', 'Server':
```

Response对象包含服务器返回的所有信息，也包含请求的Request信息

1.4 Response对象的属性

属性	说明
r.status_code	HTTP请求的返回状态，200表示连接成功，404表示失败
r.text	HTTP响应内容的字符串形式，即，url对应的页面内容
r.encoding	从HTTP header中猜测的响应内容编码方式
r.apparent_encoding	从内容中分析出的响应内容编码方式（备选编码方式）
r.content	HTTP响应内容的二进制形式

1.5 爬取网页的通用代码框架

```
import requests

def getHTMLText(url):
    try:
        r = requests.get(url, timeout=30)
        r.raise_for_status() #如果状态不是200, 引发HTTPError异常
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return "产生异常"
```

1.6 HTTP协议

HTTP , Hypertext Transfer Protocol , 超文本传输协议

HTTP是一个基于“请求与响应”模式的、无状态的应用层协议

HTTP协议采用URL作为定位网络资源的标识，URL格式如下：

http://host[:port][path]

host: 合法的Internet主机域名或IP地址

port: 端口号，缺省端口为80

path: 请求资源的路径

HTTP协议

HTTP , Hypertext Transfer Protocol , 超文本传输协议

HTTP URL实例：

http://www.fudan.edu.cn

http://220.181.111.188/duty

HTTP URL的理解：

URL是通过HTTP协议存取资源的Internet路径，一个URL对应一个数据资源

HTTP协议对资源的操作

方法	说明
GET	请求获取URL位置的资源
HEAD	请求获取URL位置资源的响应消息报告，即获得该资源的头部信息
POST	请求向URL位置的资源后附加新的数据
PUT	请求向URL位置存储一个资源，覆盖原URL位置的资源
PATCH	请求局部更新URL位置的资源，即改变该处资源的部分内容
DELETE	请求删除URL位置存储的资源

HTTP协议与Requests库

HTTP协议方法	Requests库方法	功能一致性
GET	requests.get()	一致
HEAD	requests.head()	一致
POST	requests.post()	一致
PUT	requests.put()	一致
PATCH	requests.patch()	一致
DELETE	requests.delete()	一致

Requests库的head()方法

```
>>> r = requests.head('http://httpbin.org/get')

>>> r.headers

{'Content-Length': '238', 'Access-Control-Allow-Origin': '*', 'Access-
Control-Allow-Credentials': 'true', 'Content-Type':
'application/json', 'Server': 'nginx', 'Connection': 'keep-alive',
'Date': 'Sat, 18 Feb 2017 12:07:44 GMT'}

>>> r.text

''
```

Requests库的post()方法

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}  
>>> r = requests.post('http://httpbin.org/post', data = payload)  
>>> print(r.text)  
{ ...  
  "form": {  
    "key2": "value2",  
    "key1": "value1"  
  },  
}
```

向URL POST一个字典
自动编码为form (表单)

Requests库的put()方法

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}
```

```
>>> r = requests.put('http://httpbin.org/put', data = payload)
```

```
>>> print(r.text)
```

```
{ ...
```

```
  "form": {
```

```
    "key2": "value2",
```

```
    "key1": "value1"
```

```
  },
```

```
}
```

Requests库的7个主要方法

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

1.7 request方法

`requests.request(method, url, **kwargs)`

- **method** : 请求方式，对应get/put/post等7种
- **url** : 拟获取页面的url链接
- ****kwargs**: 控制访问的参数，共13个

requests.request(**method**, **url**, ****kwargs**)

- **method** : 请求方式

```
r = requests.request('GET', url, **kwargs)  
r = requests.request('HEAD', url, **kwargs)  
r = requests.request('POST', url, **kwargs)  
r = requests.request('PUT', url, **kwargs)  
r = requests.request('PATCH', url, **kwargs)  
r = requests.request('delete', url, **kwargs)  
r = requests.request('OPTIONS', url, **kwargs)
```

requests.request(method, url, **kwargs)

- ****kwargs**: 控制访问的参数，均为可选项

params : 字典或字节序列，作为参数增加到url中

```
>>> kv = {'key1': 'value1', 'key2': 'value2'}  
>>> r = requests.request('GET', 'http://python123.io/ws', params=kv)  
>>> print(r.url)  
http://python123.io/ws?key1=value1&key2=value2
```

requests.request(**method**, **url**, ****kwargs**)

- ****kwargs**: 控制访问的参数，均为可选项

params : 字典或字节序列，作为参数增加到url中

data : 字典、字节序列或文件对象，作为Request的内容

json : JSON格式的数据，作为Request的内容

headers : 字典，HTTP定制头

cookies : 字典或CookieJar，Request中的cookie

auth : 元组，支持HTTP认证功能

```
requests.request(method, url, **kwargs)
```

- ****kwargs**: 控制访问的参数(续)

files : 字典类型，传输文件

timeout : 设定超时时间，秒为单位

proxies : 字典类型，设定访问代理服务器，可以增加登录认证

allow_redirects : True/False，默认为True，重定向开关

stream : True/False，默认为True，获取内容立即下载开关

verify : True/False，默认为True，认证SSL证书开关

cert : 本地SSL证书路径

实例1：京东商品页面的爬取

实例2：亚马逊商品页面的爬取

实例3：百度/360搜索关键字提交

实例4：网络图片的爬取和存储

例1 <https://item.jd.com/2967929.html>

The screenshot shows the JD.com website interface for the Honor 8 smartphone. At the top, there is a navigation bar with the JD.com logo, a search bar containing "iPhone 7", and a shopping cart icon with a red '0' indicating no items. Below the navigation bar, there are links for "全部商品分类", "首页", "手机首页", "新机发布", "网上营业厅", "京选卖家", "配件中心", and "手机社区". The main content area displays the product details for the "Honor 8 4GB+64GB 全网通4G手机 魅海蓝". It features a large image of the phone's back and a smaller image of a man pointing his finger towards the camera on the screen. The price is listed as 2499.00. The page also includes sections for "促销" (Promotions), "配送至" (Delivery to), "重量" (Weight), and "选择颜色" (Select Color) with options for five different colors: 流光金, 珠光白, 魅海蓝 (selected), 幻夜黑, and 樱语粉.

全代码

```
import requests
url = "https://item.jd.com/2967929.html"
try:
    r = requests.get(url)
    r.raise_for_status()
    r.encoding = r.apparent_encoding
    print(r.text[:1000])
except:
    print("爬取失败")
```

例2 https://www.amazon.cn/gp/product/B01M8L5Z3Y

亚马逊
amazon.cn
免费试享Prime

图书

进口大牌男女鞋下单售

浏览 全部商品分类 我的亚马逊 Z秒杀 礼品卡 我要开店 海外购 帮助 In English 您好 登录 我的帐户 免费试享 Prime 购物车

图书 高级搜索 所有分类 新品排行榜 销售排行榜 新书店 教材教辅 少儿 文学 小说 历史 经管 励志 人文社科 生活 科普 进口图书 促销

返回结果

在线试读↓

极简:在你拥有的一切之下,发现你想要的生活 平装 - 2016年11月5日
乔舒亚·贝克尔 (Joshua Becker) (作者)
★★★★★ 47 条商品评论 | 分享 | 自营

显示所有 格式和版本

平装
¥23.80

售价: ¥ 23.80 (6.3折)
定价: ¥ 38.00

图书满¥59免运费且可货到付款
数量: 1

试享亚马逊Prime免费配送
加入购物车

登录 即可开启一键下单。

配送至: 北京东城区 现在有货

送达日期: 明天(2月19日), 请在9小时内下单并选择“快递送货上门”。
(精确送达时间请于结账页面查询)

销售配送: 由亚马逊直接销售和发货。

全新品15 售价从 ¥ 22.80起

退换承诺: 此商品支持30天免费退换 详情

加入心愿单

亚马逊的其他卖家



全代码

```
import requests
url = "https://www.amazon.cn/gp/product/B01M8L5Z3Y"
try:
    kv = {'user-agent': 'Mozilla/5.0'}
    r = requests.get(url, headers=kv)
    r.raise_for_status()
    r.encoding = r.apparent_encoding
    print(r.text[1000:2000])
except:
    print("爬取失败")
```

例3 搜索引擎

百度的关键词接口：

`http://www.baidu.com/s?wd=keyword`

360的关键词接口：

`http://www.so.com/s?q=keyword`

百度搜索全代码

```
import requests
keyword = "Python"
try:
    kv = {'wd': keyword}
    r = requests.get("http://www.baidu.com/s", params=kv)
    print(r.request.url)
    r.raise_for_status()
    print(len(r.text))
except:
    print("爬取失败")
```

360搜索全代码

```
import requests
keyword = "Python"
try:
    kv = {'q': keyword}
    r = requests.get("http://www.so.com/s", params=kv)
    print(r.request.url)
    r.raise_for_status()
    print(len(r.text))
except:
    print("爬取失败")
```

例4 网络图片的爬取

网络图片链接的格式：

`http://www.example.com/picture.jpg`

国家地理：`http://www.nationalgeographic.com.cn/`

选择一个图片Web页面：

`http://www.nationalgeographic.com.cn/photography/photo_of_the_day/3921.html`

图片爬取全代码

```
import requests
import os
url = "http://image.nationalgeographic.com.cn/2017/0211/20170211061910157.jpg"
root = "D://pics//"
path = root + url.split('/')[-1]
try:
    if not os.path.exists(root):
        os.mkdir(root)
    if not os.path.exists(path):
        r = requests.get(url)
        with open(path, 'wb') as f:
            f.write(r.content)
            f.close()
            print("文件保存成功")
    else:
        print("文件已存在")
except:
    print("爬取失败")
```

Requests 总结

```
requests.request()  
requests.get()  
requests.head()  
requests.post()  
requests.put()  
requests.patch()  
requests.delete()
```

```
try:  
    r = requests.get(url, timeout=30)  
    r.raise_for_status()  
    r.encoding = r.apparent_encoding  
    return r.text  
except:  
    return "产生异常"
```

爬取网页的通用代码框架

BeautifulSoup



Requests

自动爬取HTML页面

自动网络请求提交



Beautiful Soup

解析HTML页面



Re

正则表达式

提取页面关键信息

2.1Beautiful Soup 测试

```
>>> import requests
>>> r = requests.get("http://python123.io/ws/demo.html")
>>> r.text
'<html><head><title>This is a python demo page</title></head>\r\n<body>\r\n<p class="title"><b>The demo python introduces several python courses.</b></p>\r\n<p class="course">Python is a wonderful general-purpose programming language.  
You can learn Python from novice to professional by tracking the following cou  
rses:\r\n<a href="http://www.icourse163.org/course/BIT-268001" class="py1" id=  
"link1">Basic Python</a> and <a href="http://www.icourse163.org/course/BIT-100  
1870001" class="py2" id="link2">Advanced Python</a>.</p>\r\n</body></html>'
>>> demo = r.text
```

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(demo, 'html.parser')
>>> print(soup.prettify())
<html>
  <head>
    <title>
      This is a python demo page
    </title>
  </head>
  <body>
    <p class="title">
      <b>
        The demo python introduces several python courses.
      </b>
    </p>
    <p class="course">
      Python is a wonderful general-purpose programming language. You can learn Python from novice to professional by tracking the following courses:
      <a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">
```

2.2 理解标签和标签树



<html> 标签树
 <body>
 <p class="title"> ... </p>
 </body>
</html>

Beautiful Soup库是解析、遍历、维护“标签树”的功能库

Beautiful Soup库的理解

< p > .. </ p > : 标签 Tag

p class="title" ... /p

名称 Name 属性 Attributes

成对出现

0个或多个

BeautifulSoup类



标签树



BeautifulSoup类

```
>>> from bs4 import BeautifulSoup  
>>> soup = BeautifulSoup("<html>data</html>" , "html.parser")  
>>> soup2 = BeautifulSoup(open("D://demo.html") , "html.parser")
```

BeautifulSoup对应一个HTML/XML文档的全部内容

2.3 Beautiful Soup库解析器

```
soup = BeautifulSoup('<html>data</html>', 'html.parser')
```

解析器	使用方法	条件
bs4的HTML解析器	BeautifulSoup(mk, 'html.parser')	安装bs4库
lxml的HTML解析器	BeautifulSoup(mk, 'lxml')	pip install lxml
lxml的XML解析器	BeautifulSoup(mk, 'xml')	pip install lxml
html5lib的解析器	BeautifulSoup(mk, 'html5lib')	pip install html5lib

2.4 BeautifulSoup类的基本元素

<p class="title"> ... </p>

基本元素	说明
Tag	标签，最基本的信息组织单元，分别用<>和</>标明开头和结尾
Name	标签的名字，<p>...</p>的名字是'p'，格式：<tag>.name
Attributes	标签的属性，字典形式组织，格式：<tag>.attrs
NavigableString	标签内非属性字符串，<>...</>中字符串，格式：<tag>.string
Comment	标签内字符串的注释部分，一种特殊的Comment类型

2.4.1 Tag 标签

基本元素	说明
Tag	标签，最基本的信息组织单元，分别用<>和</>标明开头和结尾

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(demo, "html.parser")
>>> soup.title
<title>This is a python demo page</title>
>>> tag = soup.a
>>> tag
<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Bas
ic Python</a>
```

任何存在于HTML语法中的标签都可以用soup.<tag>访问获得
当HTML文档中存在多个相同<tag>对应内容时，soup.<tag>返回第一个

2.4.2 Tag的name

基本元素	说明
Name	标签的名字，<p>...</p>的名字是 'p'，格式：<tag>.name

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(demo, "html.parser")
>>> soup.a.name
'a'
>>> soup.a.parent.name
'p'
>>> soup.a.parent.parent.name
'body'
>>>
```

每个<tag>都有自己的名字，通过<tag>.name获取，字符串类型

2.4.3 Tag的attrs（属性）

基本元素	说明
Attributes	标签的属性，字典形式组织，格式： <code><tag>.attrs</code>

```
>>> tag = soup.a
>>> tag.attrs
{'id': 'link1', 'class': ['py1'], 'href': 'http://www.icourse163.org/course/BIT-268001'}
>>> tag.attrs['class']
['py1']
>>> tag.attrs['href']
'http://www.icourse163.org/course/BIT-268001'
>>> type(tag.attrs)
<class 'dict'>
>>> type(tag)
<class 'bs4.element.Tag'>
```

一个`<tag>`可以有0或多个属性，字典类型

2.4.4 Tag的NavigableString

基本元素	说明
NavigableString	标签内非属性字符串，<>...</>中字符串，格式： <code><tag>.string</code>

```
>>> soup.a  
<a class="py1" href="http://www.icourse163.org/course/BIT-26  
8001" id="link1">Basic Python</a>  
>>> soup.a.string  
'Basic Python'  
>>> soup.p  
<p class="title"><b>The demo python introduces several pytho  
n courses.</b></p>  
>>> soup.p.string  
'The demo python introduces several python courses.'  
>>> type(soup.p.string)  
<class 'bs4.element.NavigableString'>
```

NavigableString可以跨越多个层次

BeautifulSoup类的基本元素

<p class="title"> ... </p>

基本元素	说明
Tag	标签，最基本的信息组织单元，分别用<>和</>标明开头和结尾
Name	标签的名字，<p>...</p>的名字是'p'，格式：<tag>.name
Attributes	标签的属性，字典形式组织，格式：<tag>.attrs
NavigableString	标签内非属性字符串，<>...</>中字符串，格式：<tag>.string
Comment	标签内字符串的注释部分，一种特殊的Comment类型

标签 <tag>

<p class="title"> ... </p>

名称 .name 属性 .attrs 非属性字符串/注释

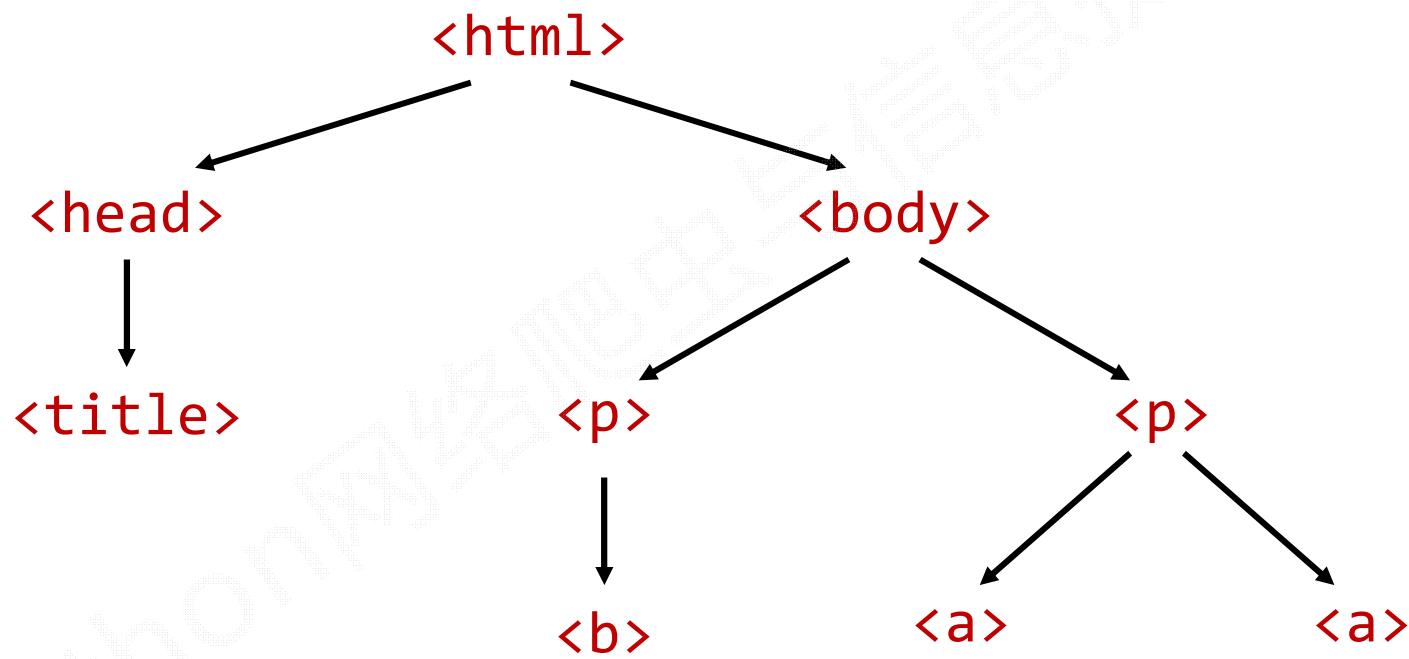
.string

2.5 HTML基本格式

```
<html>
  <head>
    <title>This is a python demo page</title>
  </head>
  <body>
    <p class="title">
      <b>The demo python introduces several python courses.</b>
    </p>
    <p class="course">Python is a wonderful general-purpose programming language.
You can learn Python from novice to professional by tracking the following courses:
      <a href="http://www.icourse163.org/course/BIT-268001" class="py1"
id="link1">Basic Python</a> and
      <a href="http://www.icourse163.org/course/BIT-1001870001" class="py2"
id="link2">Advanced Python</a>.
    </p>
  </body>
</html>
```

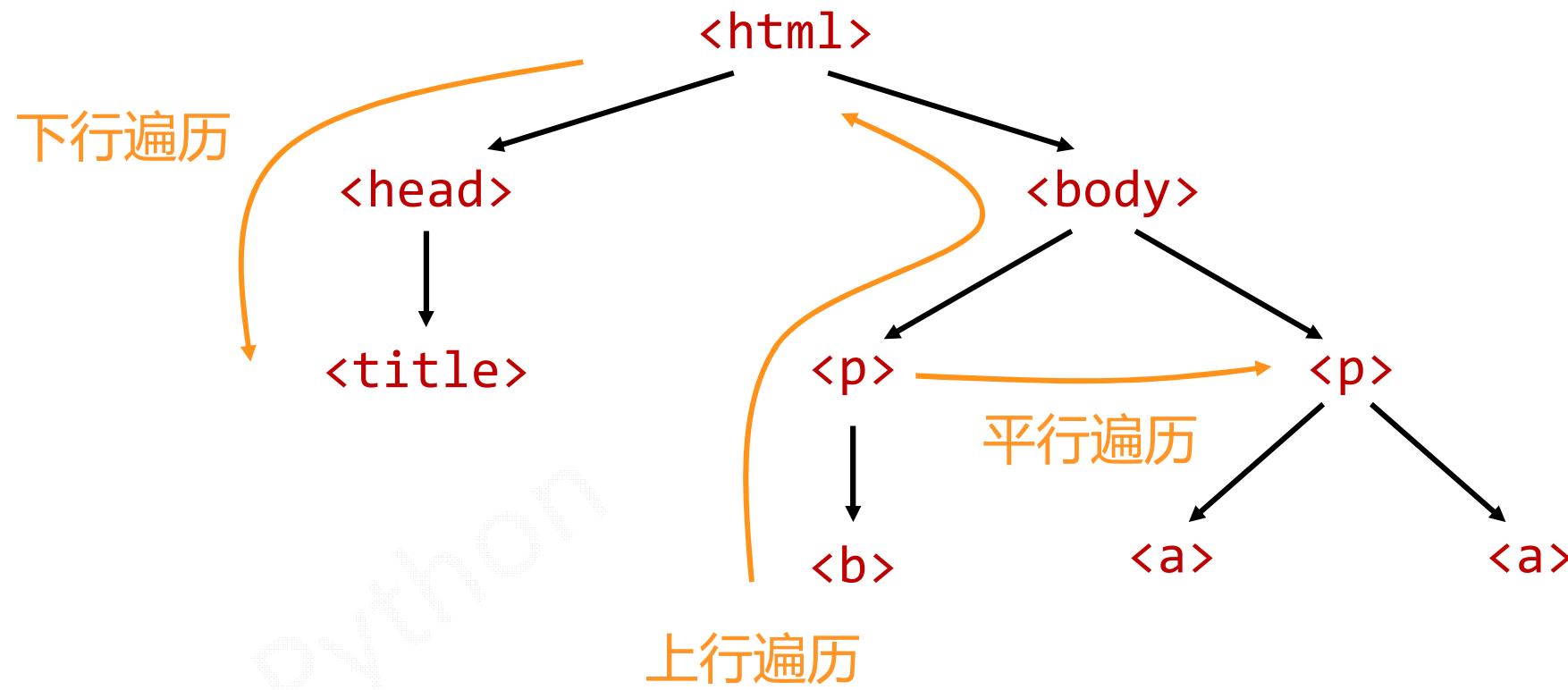
<>...</>构成了所属关系

HTML基本格式



<>...</>构成了所属关系，形成了标签的树形结构

2.6 HTML遍历



2.6.1标签树的下行遍历

属性	说明
.contents	子节点的列表，将<tag>所有儿子节点存入列表
.children	子节点的迭代类型，与.contents类似，用于循环遍历儿子节点
.descendants	子孙节点的迭代类型，包含所有子孙节点，用于循环遍历

BeautifulSoup类型是标签树的根节点

2.6.2 标签树的上行遍历

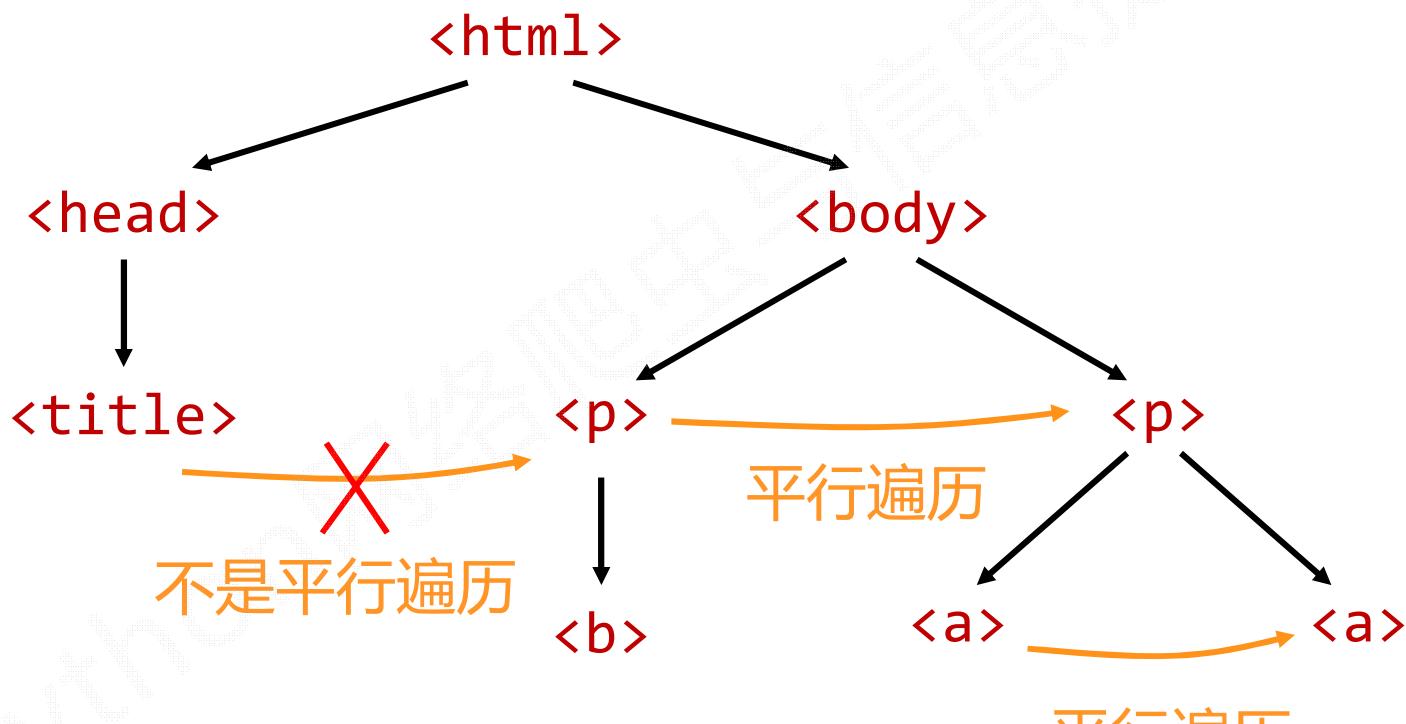
属性	说明
.parent	节点的父亲标签
.parents	节点先辈标签的迭代类型，用于循环遍历先辈节点



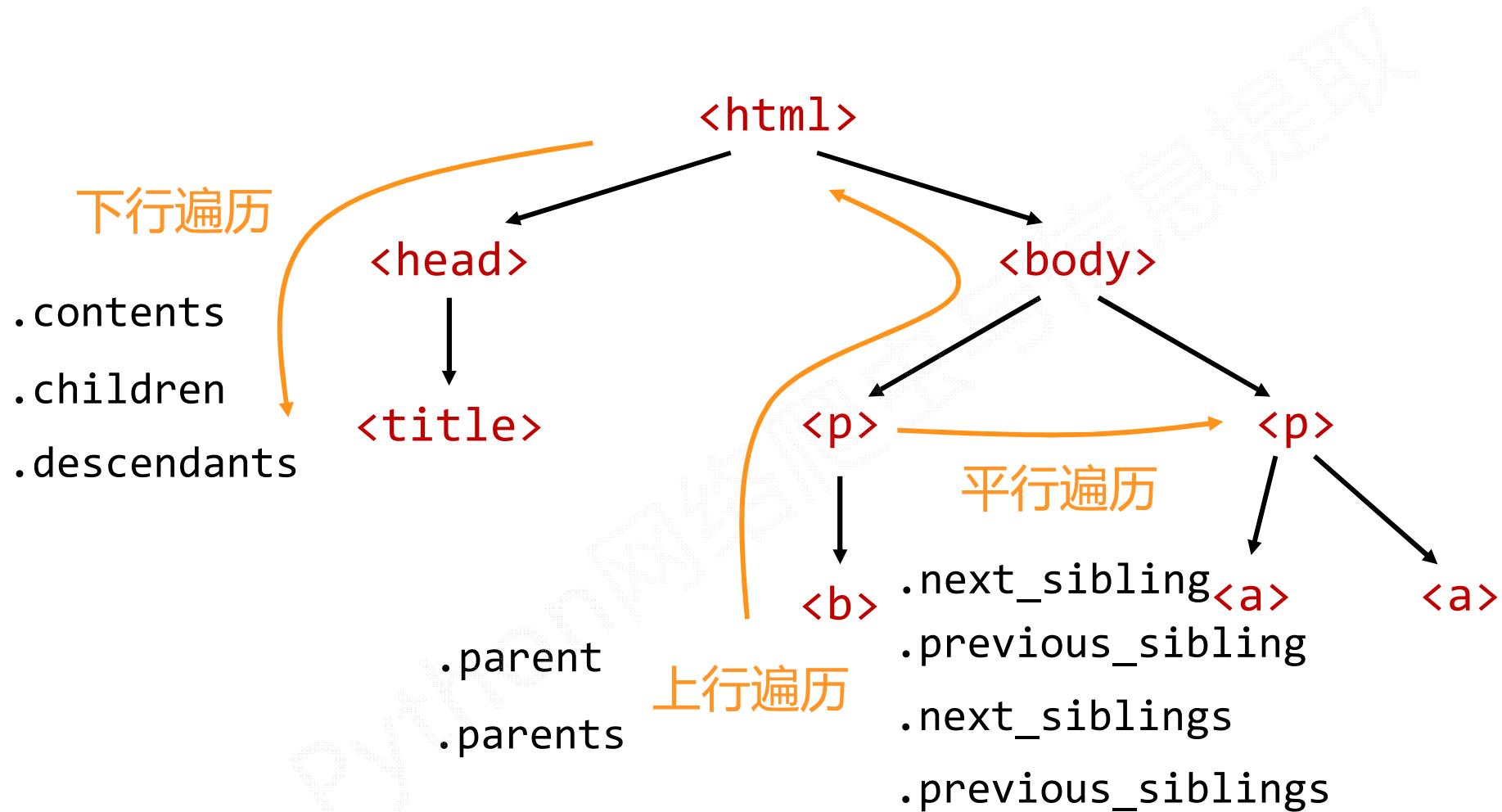
2.6.3标签树的平行遍历

属性	说明
.next_sibling	返回按照HTML文本顺序的下一个平行节点标签
.previous_sibling	返回按照HTML文本顺序的上一个平行节点标签
.next_siblings	迭代类型，返回按照HTML文本顺序的后续所有平行节点标签
.previous_siblings	迭代类型，返回按照HTML文本顺序的前续所有平行节点标签

标签树的平行遍历



平行遍历发生在同一个父节点下的各节点间



2.7 信息提取

从标记后的信息中提取所关注的内容



2.7.1 方法一

完整解析信息的标记形式，再提取关键信息

需要标记解析器，例如：bs4库的标签树遍历

优点：信息解析准确

缺点：提取过程繁琐，速度慢

2.7.2 方法二

无视标记形式，直接搜索关键信息

对信息的文本查找函数即可

优点：提取过程简洁，速度较快

缺点：提取结果准确性与信息内容相关

2.7.3 融合方法

融合方法：结合形式解析与搜索方法，提取关键信息

需要标记解析器及文本查找函数

实例

提取HTML中所有URL链接

思路： 1) 搜索到所有[标签](#)
2) 解析[标签](#)格式，提取[href](#)后的链接内容

2.8

soup.find_all() 方法

```
<>.find_all(name, attrs, recursive, string, **kwargs)
```

返回一个列表类型，存储查找的结果

- **name** : 对标签名称的检索字符串

```
>>> soup.find_all('a')
[<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic Python</a>, <a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Advanced Python</a>]
>>> soup.find_all(['a','b'])
[<b>The demo python introduces several python courses.</b>, <a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic Python </a>, <a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Advanced Python</a>]
>>>
```

```
<>.find_all(name, attrs, recursive, string, **kwargs)
```

- **name** : 对标签名称的检索字符串

```
>>> for tag in soup.find_all(True):
    print(tag.name)

html
head
title
body
p
b
p
a
a
>>>
```



```
>>> import re
>>> for tag in soup.find_all(re.compile('b')):
    print(tag.name)

body
b
>>>
```

```
<>.find_all(name, attrs, recursive, string, **kwargs)
```

- **name** : 对标签名称的检索字符串
- **attrs**: 对标签属性值的检索字符串，可标注属性检索

```
>>> soup.find_all('p', 'course')
[<p class="course">Python is a wonderful general-purpose programming language. You can learn Python from novice to professional by tracking the following courses:  
<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic Python</a> and <a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Advanced Python</a>.</p>]
>>> soup.find_all(id='link1')
[<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic Python</a>]
```

```
<>.find_all(name, attrs, recursive, string, **kwargs)
```

- **name** : 对标签名称的检索字符串
- **attrs**: 对标签属性值的检索字符串，可标注属性检索

```
>>> soup.find_all(id='link')
[]
>>> import re
>>> soup.find_all(id=re.compile('link'))
[<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic Python</a>, <a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Advanced Python</a>]
>>>
```

```
<>.find_all(name, attrs, recursive, string, **kwargs)
```

- **name** : 对标签名称的检索字符串
- **attrs**: 对标签属性值的检索字符串，可标注属性检索
- **recursive**: 是否对子孙全部检索，默认True

```
>>> soup.find_all('a')
[<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic Python</a>, <a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Advanced Python</a>]
>>> soup.find_all('a', recursive=False)
[]
```

```
<>.find_all(name, attrs, recursive, string, **kwargs)
```

- **name** : 对标签名称的检索字符串
- **attrs**: 对标签属性值的检索字符串，可标注属性检索
- **recursive**: 是否对子孙全部检索，默认True
- **string**: <>...</>中字符串区域的检索字符串

```
<>.find_all(name, attrs, recursive, string, **kwargs)
```

```
>>> soup
<html><head><title>This is a python demo page</title></head>
<body>
<p class="title"><b>The demo python introduces several python courses.</b></p>
<p class="course">Python is a wonderful general-purpose programming language. You
can learn Python from novice to professional by tracking the following courses:
<a class="py1" href="http://www.icourse163.org/course/BIT-268001" id="link1">Basic Python</a> and <a class="py2" href="http://www.icourse163.org/course/BIT-1001870001" id="link2">Advanced Python</a>.</p>
</body></html>
>>> soup.find_all(string='Basic Python')
['Basic Python']
>>> import re
>>> soup.find_all(string=re.compile('python'))
['This is a python demo page', 'The demo python introduces several python courses
.']}
```

2.9 扩展方法

方法	说明
<>.find()	搜索且只返回一个结果，同.find_all()参数
<>.find_parents()	在先辈节点中搜索，返回列表类型，同.find_all()参数
<>.find_parent()	在先辈节点中返回一个结果，同.find()参数
<>.find_next_siblings()	在后续平行节点中搜索，返回列表类型，同.find_all()参数
<>.find_next_sibling()	在后续平行节点中返回一个结果，同.find()参数
<>.find_previous_siblings()	在前序平行节点中搜索，返回列表类型，同.find_all()参数
<>.find_previous_sibling()	在前序平行节点中返回一个结果，同.find()参数

实例 中国大学排名

<http://www.zuihaodaxue.cn/zuihaodaxuepaiming2016.html>

The screenshot shows a web browser displaying the 'ZUIHAODAXUE.COM' website. The title bar reads '2016中国最好大学排名' and the URL is 'zuihaodaxue.cn/zuihaodaxuepaiming2016.html'. The page features a dark blue header with the site's logo '最好大学网' and 'ZUIHAODAXUE.COM'. On the right side of the header are links for 'English | 订阅我们' (Subscribe). Below the header is a navigation menu with links to '网站首页', '中国大学排名', '世界大学排名', '原创分析', '要闻资讯', '院校信息', and '联系我们'. There are also social media icons for Weibo and WeChat.

The main content area displays the '软科中国最好大学排名2016' (2016 Chinese Best University Ranking) with a Scopus logo. A dropdown menu next to the year '2016' indicates the ranking year. Below this, a text box states: "中国最好大学排名”的排名范围是教育部公布的全国普通高等学校名单（2015年5月21日公布）中，1216所办学层次为本科的大学。这其中公办大学793所、民办大学140所、独立学院283所。" and a link '查看排名方法' (View Ranking Method).

To the right of the main content, there is a sidebar titled '相关文章' (Related Articles) which lists various university ranking topics from 2016, such as '中国最好大学排名的特点', '中国最好大学排名-排名方法', '2016中国最好大学排名', etc. At the bottom of the sidebar is a section titled '报告下载' (Report Download) with a blue button.

排名	学校名称	省市	总分	指标得分
				生源质量 (新生高考成绩得分)
1	清华大学	北京市	95.9	100.0
2	北京大学	北京市	82.6	98.9
3	浙江大学	浙江省	80	88.8
4	上海交通大学	上海市	78.7	90.6
5	复旦大学	上海市	70.9	90.4

功能描述

输入：大学排名URL链接

输出：大学排名信息的屏幕输出（排名，大学名称，总分）

技术路线：`requests-bs4`

定向爬虫：仅对输入URL进行爬取，不扩展爬取

程序的结构设计

步骤1：从网络上获取大学排名网页内容 `getHTMLText()`

步骤2：提取网页内容中信息到合适的数据结构 `fillUnivList()`

步骤3：利用数据结构保存并输出结果 `saveUnivList()`

main()

```
#导入库
import requests
from bs4 import BeautifulSoup

#获取网页内容
def getHTMLText(url):
    return html

#提取信息
def fillUnivList(html):
    return ulist

#保存输出数据
def saveUnivList(ulist):
    return udat

def main():
    url = "http://www.zuihaodaxue.cn/zuihaodaxuepaiming2016.html"
    html = getHTMLText(url)
    ulist = fillUnivList(html)
    udat = saveUnivList(ulist)
    return udat[:20]

main()
```

getHTMLText()

```
def getHTMLText(url):
    try:
        r = requests.get(url, timeout=30)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return ""
```

fillUnivList()

```
import bs4
def fillUnivList(html):
    ulist = []
    soup = BeautifulSoup(html, "html.parser")
    for tr in soup.find('tbody').children:
        if isinstance(tr, bs4.element.Tag):
            tds = tr('td')
            ulist.append([tds[0].string, tds[1].string, tds[3].string])
    return ulist
```

saveUnivList()

```
import pandas as pd
def saveUnivList(ulist):
    udat = pd.DataFrame(ulist)
    udat.columns = ['排名', '学校', '得分']
    udat.to_csv('univlist.csv')
    return udat
```

全代码

```
#导入库
import requests
from bs4 import BeautifulSoup

#获取网页内容
def getHTMLText(url):
    try:
        r = requests.get(url, timeout=30)
        r.raise_for_status
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return ""

#提取信息
import bs4
def fillUnivList(html):
    ulist = []
    soup = BeautifulSoup(html, "html.parser")
    for tr in soup.find('tbody').children:
        if isinstance(tr, bs4.element.Tag):
            tds = tr('td')
            ulist.append([tds[0].string, tds[1].string, tds[3].string])
    return ulist

#保存输出数据
import pandas as pd
def saveUnivList(ulist):
    udat = pd.DataFrame(ulist)
    udat.columns = ['排名', '学校', '得分']
    udat.to_csv('univlist.csv')
    return udat

def main():
    url = "http://www.zuihaodaxue.cn/zuihaodaxuepaiming2016.html"
    html = getHTMLText(url)
    ulist = fillUnivList(html)
    udat = saveUnivList(ulist)
    return udat[:20]

main()
```

Re



Requests

自动爬取HTML页面

自动网络请求提交



Beautiful Soup

解析HTML页面



Re

正则表达式

提取页面关键信息

正则表达式

regular expression, regex, RE

正则表达式是用来简洁表达一组字符串的表达式

- 'PN'
- 'PYN'
- 'PYTN'
- 'PYTHN'
- 'PYTHON'



正则表达式：

$P(Y|YT|YTH|YTHO)?N$

一组字符串

简洁表达

正则表达式

regular expression, regex, RE

'PY'

'PYY'

'PYYY'

'PYYYY'

.....

'PYYYYY.....'



正则表达式：
PY+

一组字符串（无穷个）

无穷字符串组的简洁表达

正则表达式

regular expression, regex, RE

'PY'开头

后续存在不多于10个字符

后续字符不能是'P'或'Y'

'PYABC' ✓

'PYKXYZ' X

一组具有某些特点的字符串
(可以枚举，但很繁琐)

正则表达式：

PY[^PY]{0,10}

某种特征字符串组的简洁表达

3.1 正则表达式的语法

P (Y | YT | YTH | YTHO)? N

正则表达式语法由字符和操作符构成

3.2 正则表达式的常用操作符

操作符	说明	实例
.	表示任何单个字符	
[]	字符集，对单个字符给出取值范围	[abc] 表示a、b、c，[a-z]表示a到z单个字符
[^]	非字符集，对单个字符给出排除范围	[^abc] 表示非a或b或c的单个字符
*	前一个字符0次或无限次扩展	abc* 表示 ab、abc、abcc、abccc等
+	前一个字符1次或无限次扩展	abc+ 表示 abc、abcc、abccc等
?	前一个字符0次或1次扩展	abc? 表示 ab、abc
	左右表达式任意一个	abc def 表示 abc、def

正则表达式的常用操作符

操作符	说明	实例
{m}	扩展前一个字符m次	ab{2}c表示abbc
{m,n}	扩展前一个字符m至n次（含n）	ab{1,2}c表示abc、abbc
^	匹配字符串开头	^abc表示abc且在一个字符串的开头
\$	匹配字符串结尾	abc\$表示abc且在一个字符串的结尾
()	分组标记，内部只能使用 操作符	(abc)表示abc，(abc def)表示abc、def
\d	数字，等价于[0-9]	
\w	单词字符，等价于[A-Za-z0-9]	

正则表达式语法实例

P(Y|YT|YTH|YTH0)?N

PYTHON+

PY[TH]ON

PY[^TH]?ON

PY{,:3}N

正则表达式

对应字符串

正则表达式语法实例

P(Y|YT|YTH|YTHO)?N 'PN'、'PYN'、'PYTN'、'PYTHN'、'PYTHON'

PYTHON+ 'PYTHON'、'PYTHONN'、'PYTHONNN' ...

PY[TH]ON 'PYTON'、'PYHON'

PY[^TH]?ON 'PYON'、'PYaON'、'PYbON'、'PYcON' ...

PY{3}N 'PN'、'PYN'、'PYYN'、'PYYYN' ...

正则表达式

对应字符串

经典正则表达式实例

`^[A-Za-z]+$`

由26个字母组成的字符串

`^[A-Za-z0-9]+$`

由26个字母和数字组成的字符串

`^-?\d+$`

整数形式的字符串

`^[0-9]*[1-9][0-9]*$`

正整数形式的字符串

`[1-9]\d{5}`

中国境内邮政编码，6位

`[\u4e00-\u9fa5]`

匹配中文字符

`\d{3}-\d{8}|\d{4}-\d{7}`

国内电话号码，010-68913536

3.3 正则表达式的表示类型

raw string类型（原生字符串类型）

re库采用raw string类型表示正则表达式，表示为：

r'text'

例如： r'[1-9]\d{5}'

r'\d{3}-\d{8}|\d{4}-\d{7}'

raw string是不包含对转义符再次转义的字符串

正则表达式的表示类型

re库也可以采用string类型表示正则表达式，但更繁琐
例如：

```
'[1-9]\d{5}'
```

```
'\d{3}-\d{8}|\d{4}-\d{7}'
```

建议：当正则表达式包含转义符时，使用raw string

3.4 Re库主要功能函数

函数	说明
re.search()	在一个字符串中搜索匹配正则表达式的第一位置，返回match对象
re.match()	从一个字符串的开始位置起匹配正则表达式，返回match对象
re.findall()	搜索字符串，以列表类型返回全部能匹配的子串
re.split()	将一个字符串按照正则表达式匹配结果进行分割，返回列表类型
re.finditer()	搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象
re.sub()	在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

`re.search(pattern, string, flags=0)`

在一个字符串中搜索匹配正则表达式的第一个位置
返回match对象

- **pattern** : 正则表达式的字符串或原生字符串表示
- **string** : 待匹配字符串
- **flags** : 正则表达式使用时的控制标记

re.search(pattern, string, flags=0)

- **flags** : 正则表达式使用时的控制标记

常用标记	说明
re.I re.IGNORECASE	忽略正则表达式的大小写，[A-Z]能够匹配小写字符
re.M re.MULTILINE	正则表达式中的^操作符能够将给定字符串的每行当作匹配开始
re.S re.DOTALL	正则表达式中的.操作符能够匹配所有字符，默认匹配除换行外的所有字符

`re.match(pattern, string, flags=0)`

从一个字符串的开始位置起匹配正则表达式
返回match对象

- **pattern** : 正则表达式的字符串或原生字符串表示
- **string** : 待匹配字符串
- **flags** : 正则表达式使用时的控制标记

```
re.findall(pattern, string, flags=0)
```

搜索字符串，以列表类型返回全部能匹配的子串

- **pattern** : 正则表达式的字符串或原生字符串表示
- **string** : 待匹配字符串
- **flags** : 正则表达式使用时的控制标记

```
re.split(pattern, string, maxsplit=0, flags=0)
```

将一个字符串按照正则表达式匹配结果进行分割
返回列表类型

- **pattern** : 正则表达式的字符串或原生字符串表示
- **string** : 待匹配字符串
- **maxsplit**: 最大分割数，剩余部分作为最后一个元素输出
- **flags** : 正则表达式使用时的控制标记

```
re.finditer(pattern, string, flags=0)
```

搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象

- **pattern** : 正则表达式的字符串或原生字符串表示
- **string** : 待匹配字符串
- **flags** : 正则表达式使用时的控制标记

```
re.sub(pattern, repl, string, count=0, flags=0)
```

在一个字符串中替换所有匹配正则表达式的子串
返回替换后的字符串

- **pattern** : 正则表达式的字符串或原生字符串表示
- **repl** : 替换匹配字符串的字符串
- **string** : 待匹配字符串
- **count** : 匹配的最大替换次数
- **flags** : 正则表达式使用时的控制标记

3.5 Match 对象

Match对象是一次匹配的结果，包含匹配的很多信息

Match对象的属性

属性	说明
.string	待匹配的文本
.re	匹配时使用的patter对象（正则表达式）
.pos	正则表达式搜索文本的开始位置
.endpos	正则表达式搜索文本的结束位置

Match对象的方法

方法	说明
.group(0)	获得匹配后的字符串
.start()	匹配字符串在原始字符串的开始位置
.end()	匹配字符串在原始字符串的结束位置
.span()	返回(.start(), .end())

实例2：淘宝商品比价定向爬虫

功能描述

目标：获取淘宝搜索页面的信息，提取其中的商品名称和价格
理解：

淘宝的搜索接口

翻页的处理



技术路线：`requests-bs4-re`

“书包”

https://s.taobao.com/search?q=书包&js=1&stats_click=search_radio_all%3A1&initiative_id=staobaoz_20170105&ie=utf8 起始页

https://s.taobao.com/search?q=书包&js=1&stats_click=search_radio_all%3A1&initiative_id=staobaoz_20170105&ie=utf8&bcoffset=0&ntoffset=0&p4ppushleft=1%2C48&s=44 第2页

https://s.taobao.com/search?q=书包&js=1&stats_click=search_radio_all%3A1&initiative_id=staobaoz_20170105&ie=utf8&bcoffset=-3&ntoffset=-3&p4ppushleft=1%2C48&s=88 每页44个商品 第3页

搜索接口和翻页的URL对应属性

定向爬虫可行性

<https://s.taobao.com/robots.txt>

User-agent: *

Disallow: /

请注意：这个例子仅探讨技术实现，请不要不加限制的爬取该网站

程序的结构设计

步骤1：提交商品搜索请求，循环获取页面

步骤2：对于每个页面，提取商品名称和价格信息

步骤3：将信息保存并输出