

# 信息标记与提取方法

WS05

---



嵩天

[www.python123.org](http://www.python123.org)

# The Website is the API ...



## Requests

自动爬取HTML页面  
自动网络请求提交

## robots.txt

网络爬虫排除标准



## Beautiful Soup

解析HTML页面



## Projects

实战项目



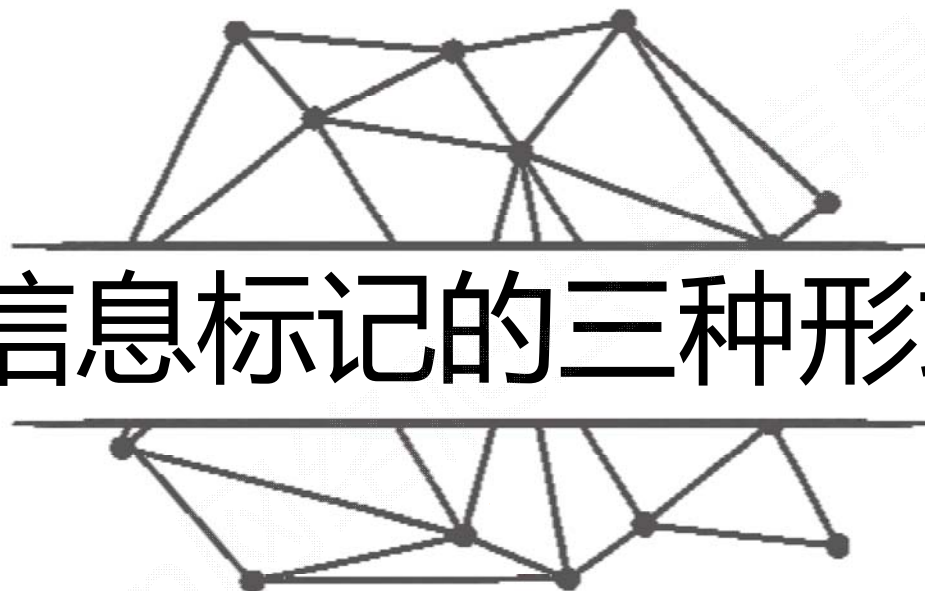
掌握定向网络数据爬取和网页解析的基本能力

# Python网络爬虫与信息提取

**python**  
弹指之间 · 享受创新

04X -Tian

# 信息标记的三种形式



# 信息的标记

'红色国防工程师的摇篮'

'首批985高校' '首批211高校'

'北京市海淀区中关村' 1940

'北京理工大学'

'中国共产党创办的第一所理工科大学'

'工业和信息化部' '第一辆轻型坦克'

'第一部低空测高雷达'

'name', '北京理工大学'

'addr', '北京市海淀区中关村'

'北京理工大学'

一个信息

一组信息

信息的标记

# 信息的标记

标记后的信息可形成信息组织结构，增加了信息维度

标记的结构与信息一样具有重要价值

标记后的信息可用于通信、存储或展示

标记后的信息更利于程序理解和运用

# HTML的信息标记



文本

声音

图像

视频

超文本

HTML是WWW(World Wide Web)的信息组织方式

# HTML的信息标记

```
<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>
```

HTML通过预定义的<>...</>标签形式组织不同类型的信息

信息标记有哪些种类呢？



# 信息标记的三种形式

**XML**

**JSON**

**YAML**

# XML

eXtensible Markup Language

标签 Tag



 ... </img>



名称 Name

属性 Attribute

# XML

eXtensible Markup Language

```

```

空元素的缩写形式

```
<!-- This is a comment, very useful -->
```

注释书写形式

# XML

eXtensible Markup Language

<name> ... </name>

<name />

<!-- -->

# JSON

JavaScript Object Notation

有类型的键值对 key:value

“” 类型 → “name” : “北京理工大学”

键 key      值 value

The diagram illustrates a JSON key-value pair: "name" : "北京理工大学". The key "name" is underlined in purple, and the value "北京理工大学" is underlined in black. An arrow points from the text "“” 类型" to the key. Another arrow points from the text "键 key" to the key. A third arrow points from the text "值 value" to the value.

# JSON

JavaScript Object Notation

“name” : [ “北京理工大学”, “延安自然科学学院” ]

多值用[, ]组织



# JSON

JavaScript Object Notation

“name” : {



“newName” : “北京理工大学” ,

键值对嵌套用{,}

“oldName” : “延安自然科学学院”



}

# JSON

JavaScript Object Notation

“key” : “value”

“key” : [“value1”, “value2”]

“key” : {“subkey” : “subvalue”}



# YAML

YAML Ain't Markup Language

无类型键值对 key:value

仅字符串 →

name : 北京理工大学

键 key

值 value

# YAML

YAML Ain't Markup Language

缩进表达所属关系

name :

newName : 北京理工大学

oldName : 延安自然科学学院

# YAML

YAML Ain't Markup Language

- 表达并列关系

name :

- 北京理工大学
- 延安自然科学学院

# YAML

YAML Ain't Markup Language

| 表达整块数据    # 表示注释

text: |            #学校介绍

北京理工大学创立于1940年，前身是延安自然科学学院，是中国共产党创办的第一所理工科大学，毛泽东同志亲自题写校名，李富春、徐特立、李强等老一辈无产阶级革命家先后担任学校主要领导。学校是新中国成立以来国家历批次重点建设的高校，首批进入国家“211工程”和“985工程”建设行列；在全球具有广泛影响力的英国QS“世界大学500强”中，位列入选的中国大陆高校第15位。学校现隶属于工业和信息化部。

# YAML

YAML Ain't Markup Language

```
key : value
```

```
key : #Comment
```

```
- value1
```

```
- value2
```

```
key :
```

```
  subkey : subvalue
```



# 三种信息标记形式的比较

# XML

eXtensible Markup Language

<name> ... </name>

<name />

<!-- -->

# JSON

JavaScript Object Notation

“key” : “value”

“key” : [“value1”, “value2”]

“key” : {“subkey” : “subvalue”}



# YAML

YAML Ain't Markup Language

```
key : value
```

```
key : #Comment
```

```
- value1
```

```
- value2
```

```
key :
```

```
  subkey : subvalue
```

# XML实例

```
<person>
  <firstName>Tian</firstName>
  <lastName>Song</lastName>
  <address>
    <streetAddr>中关村南大街5号</streetAddr>
    <city>北京市</city>
    <zipcode>100081</zipcode>
  </address>
  <prof>Computer System</prof><prof>Security</prof>
</person>
```

# JSON实例

```
{  
  "firstName" : "Tian" ,  
  "lastName"  : "Song" ,  
  "address"   : {  
    "streetAddr" : "中关村南大街5号" ,  
    "city"       : "北京市" ,  
    "zipcode"    : "100081"  
  } ,  
  "prof"      : [ "Computer System" , "Security" ]  
}
```

# YAML实例

```
firstName : Tian
lastName  : Song
address   :
    streetAddr : 中关村南大街5号
    city       : 北京市
    zipcode    : 100081
prof       :
-Computer System
-Security
```

# 三种信息标记形式的比较

**XML** 最早的通用信息标记语言，可扩展性好，但繁琐

**JSON** 信息有类型，适合程序处理(js)，较XML简洁

**YAML** 信息无类型，文本信息比例最高，可读性好

# 三种信息标记形式的比较

**XML**

Internet上的信息交互与传递

**JSON**

移动应用云端和节点的信息通信，无注释

**YAML**

各类系统的配置文件，有注释易读

# 信息提取的一般方法



# 信息提取

从标记后的信息中提取所关注的内容

**XML JSON YAML**

(标记, 信息)



# 信息提取的一般方法

方法一：完整解析信息的标记形式，再提取关键信息

XML JSON YAML

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

# 信息提取的一般方法

方法二：无视标记形式，直接搜索关键信息

搜索

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

# 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

XML JSON YAML 搜索

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<a>标签

2) 解析<a>标签格式，提取href后的链接内容

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(demo, "html.parser")
>>> for link in soup.find_all('a'):
    print(link.get('href'))

http://www.icourse163.org/course/BIT-268001
http://www.icourse163.org/course/BIT-1001870001
>>>
```

# 基于bs4库的HTML内容查找方法

# 回顾demo.html

```
>>> import requests
>>> r = requests.get("http://python123.io/ws/demo.html")
>>> demo = r.text
>>> demo
'<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>'
>>>
```

`<>.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">Basi  
c Python</a> and <a class="py2" href="http://www.icourse163.org/course/BIT-100187  
0001" id="link2">Advanced Python</a>.</p>
```

```
</body></html>
```

```
>>> soup.find_all(string='Basic Python')
```

```
['Basic Python']
```

```
>>> import re
```

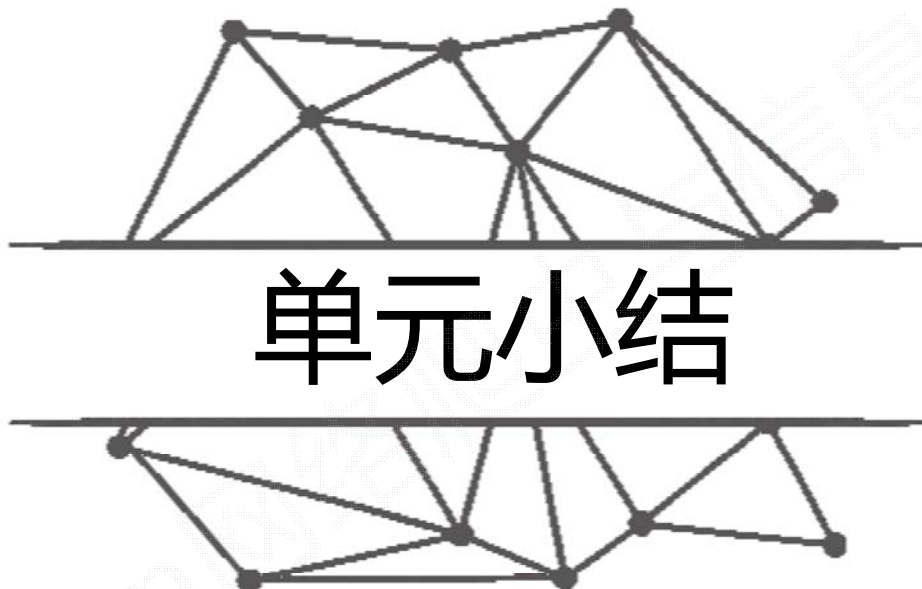
```
>>> soup.find_all(string=re.compile('python'))
```

```
['This is a python demo page', 'The demo python introduces several python courses  
.']
```

`<tag>(..)` 等价于 `<tag>.find_all(..)`  
`soup(..)` 等价于 `soup.find_all(..)`

# 扩展方法

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



# 单元小结



# 信息标记与提取方法

**XML**

`<>..</>`

信息提取的一般方法

**JSON**

有类型 `key:value`

`<tag>(..)` 等价于 `<tag>.find_all(..)`

**YAML**

无类型 `key:value`

`soup(..)` 等价于 `soup.find_all(..)`

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