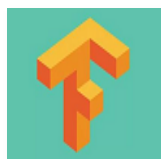


# [python知识] 爬虫知识之BeautifulSoup库安装及简单介绍

原创 Eastmount 最后发布于2015-03-25 17:50:05 阅读数 19355 ☆ 收藏

展开



## Python+TensorFlow人工智能

¥9.90

该专栏为人工智能入门专栏，采用Python3和TensorFlow实现人工智能相...



Eastmount

去订阅

## 一. 前言

在前面的几篇文章中我介绍了如何通过Python分析源代码来爬取博客、维基百科InfoBox和图片，其文章链接如下：

[\[python学习\] 简单爬取维基百科程序语言消息盒](#)

[\[Python学习\] 简单网络爬虫抓取博客文章及思想介绍](#)

[\[python学习\] 简单爬取图片网站图库中图片](#)

其中核心代码如下：

```
# coding=utf-8
import urllib
import re

# 下载静态HTML网页
url='http://www.csdn.net/'
content = urllib.urlopen(url).read()
open('csdn.html','w+').write(content)

# 获取标题
title_pat=r'(<=<title>).*?(?=</title>)'
title_ex=re.compile(title_pat,re.M|re.S)
title_obj=re.search(title_ex, content)
title=title_obj.group()
print title

# 获取超链接内容
href = r'<a href=.*?>(.*?)</a>'
m = re.findall(href,content,re.S|re.M)
for text in m:
    print unicode(text,'utf-8')
    break #只输出一个url
```

**输出结果如下:**

```
>>>
CSDN.NET - 全球最大中文IT社区，为IT专业技术人员提供最全面的信息传播和服务平台
登录
>>>
```

**图片下载的核心代码如下:**

```
import os
import urllib
class AppURLopener(urllib.FancyURLopener):
    version = "Mozilla/5.0"
urllib._urlopener = AppURLopener()
url = "http://creativ.allyes.com.cn/imedia/csdn/20150228/15_41_49_5B9C9E6A.jpg"
filename = os.path.basename(url)
urllib.urlretrieve(url , filename)
```

**但是上面这种分析HTML来爬取网站内容的方法存在很多弊端，譬如：**

**1.正则表达式被HTML源码所约束，而不是取决于更抽象的结构；网页结构中很小的改动可能会导致程序的中断。**

**2.程序需要根据实际HTML源码分析内容，可能会遇到字符实体如&amp;之类的HTML特性，需要指定处理如<span> </span>、图标超链接、下标等不同内容。**

**3.正则表达式并不是完全可读的，更复杂的HTML代码和查询表达式会显得很乱。**

**正如《Python基础教程(第2版)》采用两种解决方案：第一个是使用Tidy(Python库)的程序和XHTML解析；第二个是使用BeautifulSoup库。**

## **二. 安装及介绍Beautiful Soup库**

**Beautiful Soup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。**

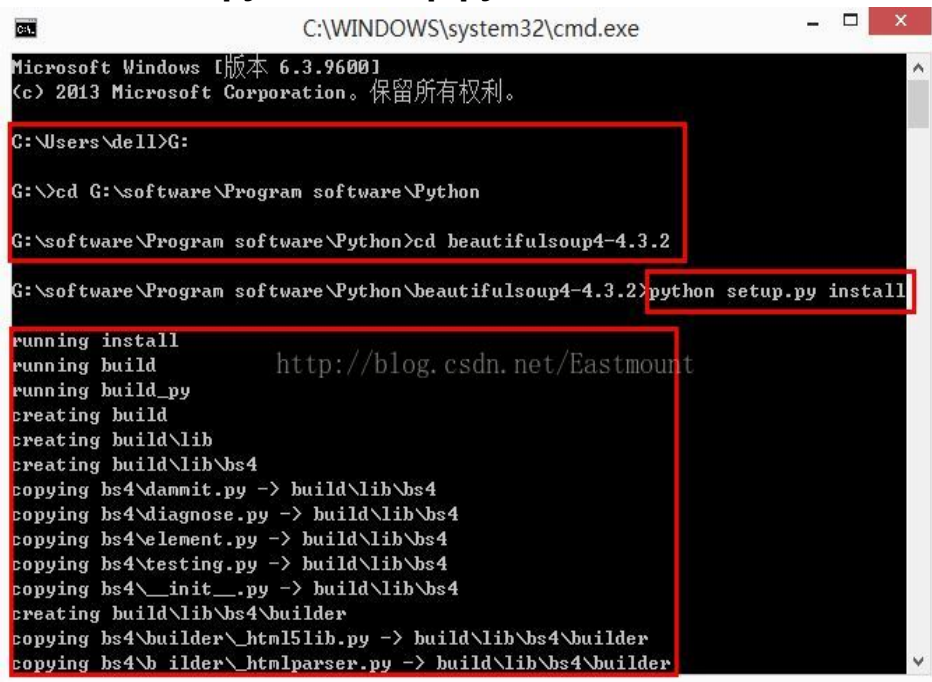
**正如书中所说“那些糟糕的网页不是你写的，你只是试图从中获得一些数据。现在你不用关心HTML是什么样子的，解析器帮你实现”。**

**下载地址:**

<http://www.crummy.com/software/BeautifulSoup/>

<http://www.crummy.com/software/BeautifulSoup/bs4/download/4.3/>

## 安装过程如下图所示：python setup.py install



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\dell>G:

G:\>cd G:\software\Program software\Python

G:\software\Program software\Python>cd beautifulsoup4-4.3.2

G:\software\Program software\Python\beautifulsoup4-4.3.2>python setup.py install

running install
running build
running build_py
creating build
creating build\lib
creating build\lib\bs4
copying bs4\dammit.py -> build\lib\bs4
copying bs4\diagnose.py -> build\lib\bs4
copying bs4\element.py -> build\lib\bs4
copying bs4\testing.py -> build\lib\bs4
copying bs4\__init__.py -> build\lib\bs4
creating build\lib\bs4\builder
copying bs4\builder\_html5lib.py -> build\lib\bs4\builder
copying bs4\builder\_htmlparser.py -> build\lib\bs4\builder
```

具体使用方法建议参照中文：

<http://www.crummy.com/software/BeautifulSoup/bs4/doc/index.zh.html>

其中BeautifulSoup的用法简单讲解下，使用“爱丽丝梦游仙境”的官方例子：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from bs4 import BeautifulSoup

html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>
<p class="story">Once upon a time there were three little sisters; and their
names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p> <p class="story">...</p>
"""

# 获取BeautifulSoup对象并按标准缩进格式输出
soup = BeautifulSoup(html_doc)
print(soup.prettify())
```

输出内容按照标准的缩进格式的结构输出如下：

```

<html> <head>
  <title>
    The Dormouse's story
  </title>
</head>
<body>
  <p class="title">
    <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>
</body>
</html>

```

## 下面是BeautifulSoup库简单快速入门介绍: (参考: [官方文档](#))

```

''' 获取title值 '''
print soup.title
# <title>The Dormouse's story</title>
print soup.title.name
# title
print unicode(soup.title.string)
# The Dormouse's story

''' 获取<p>值 '''
print soup.p

```

```
# <p class="title"><b>The Dormouse's story</b></p>
print soup.a
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>

'''从文档中找到<a>的所有标签链接'''
print soup.find_all('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>]
for link in soup.find_all('a'):
    print(link.get('href'))
    # http://example.com/elsie
    # http://example.com/lacie
    # http://example.com/tillie
print soup.find(id='link3')
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

**如果想获取文章中所有文字内容，代码如下：**

```
'''从文档中获取所有文字内容'''
print soup.get_text()
# The Dormouse's story
#
# The Dormouse's story
#
# Once upon a time there were three little sisters; and their names were
# Elsie,
# Lacie and
# Tillie;
# and they lived at the bottom of a well.
#
# ...
```

**同时在这过程中你可能会遇到两个典型的错误提示：**

### **1.ImportError: No module named BeautifulSoup**

当你成功安装BeautifulSoup 4库后，“from BeautifulSoup import BeautifulSoup”可能会遇到该错误。

```
Traceback (most recent call last):
  File "G:\software\Program software\Python\python insert\testss.py", line 4, in
<module>
    from BeautifulSoup import BeautifulSoup
ImportError: No module named BeautifulSoup
```

其中的原因是BeautifulSoup 4库改名为bs4，需要使用“from bs4 import BeautifulSoup”导入。

## 2.TypeError: an integer is required

当你使用“`print soup.title.string`”获取title的值时，可能会遇到该错误。如下：

```
Traceback (most recent call last):
  File "G:\software\Program software\Python\python insert\testss.py", line 25, in
in <module>
    print soup.title.string
  File "G:\software\Program software\Python\python insert\lib\idlelib\PyShell.py", line 1344, in write
    s = unicode.__getslice__(s, None, None)
TypeError: an integer is required
```

它应该是IDLE的BUG，当使用命令行Command没有任何错误。参考：[stackoverflow](#)。同时可以通过下面的代码解决该问题：

```
print unicode(soup.title.string)
print str(soup.title.string)
```

## 三. BeautifulSoup常用方法介绍

Beautiful Soup将复杂HTML文档转换成一个复杂的树形结构,每个节点都是Python对象,所有对象可以归纳为4种:Tag、NavigableString、BeautifulSoup、Comment|

### 1.Tag标签

tag对象与XML或HTML文档中的tag相同，它有很多方法和属性。其中最重要的属性name和attribute。用法如下：

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from bs4 import BeautifulSoup

html = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title" id="start"><b>The Dormouse's story</b></p>
"""

soup = BeautifulSoup(html)
tag = soup.p
print tag
# <p class="title" id="start"><b>The Dormouse's story</b></p>
print type(tag)
# <class 'bs4.element.Tag'>
print tag.name
# p 标签名字
print tag['class']
# [u'title']
print tag.attrs
```

```
# {'class': [u'title'], u'id': u'start'}
```

使用BeautifulSoup每个tag都有自己的名字，可以通过.name来获取；同样一个tag可能有很多个属性，属性的操作方法与字典相同，可以直接通过“.attrs”获取属性。至于修改、删除操作请参考文档。

## 2.NavigableString

字符串常被包含在tag内，Beautiful Soup用NavigableString类来包装tag中的字符串。一个NavigableString字符串与Python中的Unicode字符串相同，并且还支持包含在**遍历文档树**和**搜索文档树**中的一些特性，通过unicode()方法可以直接将NavigableString对象转换成Unicode字符串。

```
print unicode(tag.string)
# The Dormouse's story
print type(tag.string)
# <class 'bs4.element.NavigableString'>
tag.string.replace_with("No longer bold")
print tag
# <p class="title" id="start"><b>No longer bold</b></p>
```

这是获取 “<p class="title" id="start"><b>The Dormouse's story</b></p>” 中tag = soup.p的值，其中tag中包含的字符串不能编辑，但可通过函数replace\_with()替换。

NavigableString 对象支持遍历文档树和搜索文档树 中定义的大部分属性，并非全部。尤其是一个字符串不能包含其它内容(tag能够包含字符串或是其它tag)，字符串不支持.contents 或 .string 属性或 find() 方法。

如果想在Beautiful Soup之外使用 NavigableString 对象，需要调用 unicode() 方法，将该对象转换成普通的Unicode字符串，否则就算Beautiful Soup已方法已经执行结束，该对象的输出也会带有对象的引用地址。这样会浪费内存。

## 3.Beautiful Soup对象

该对象表示的是一个文档的全部内容，大部分时候可以把它当做Tag对象，它支持遍历文档树和搜索文档树中的大部分方法。

**注意：因为BeautifulSoup对象并不是真正的HTML或XML的tag，所以它没有name和 attribute属性，但有时查看它的.name属性可以通过BeautifulSoup对象包含的一个值为[document]的特殊实行.name实现——soup.name。**

Beautiful Soup中定义的其它类型都可能会出现在XML的文档中：CData , ProcessingInstruction , Declaration , Doctype 。与 Comment 对象类似，这些类都是 NavigableString 的子类，只是添加了一些额外的方法的字符串独享。



## 4.Command注释

Tag、NavigableString、BeautifulSoup几乎覆盖了html和xml中的所有内容，但是还有些特殊对象容易让人担心——注释。Comment对象是一个特殊类型的NavigableString对象。

```
markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
soup = BeautifulSoup(markup)
comment = soup.b.string
print type(comment)
# <class 'bs4.element.Comment'>
print unicode(comment)
# Hey, buddy. Want to buy a used parser?
```

介绍完这四个对象后，下面简单介绍遍历文档树和搜索文档树及常用的函数。

## 5.遍历文档树

一个Tag可能包含多个字符串或其它的Tag，这些都是这个Tag的子节点。

BeautifulSoup提供了许多操作和遍历子节点的属性。引用官方文档中爱丽丝例子：

操作文档最简单的方法是告诉你想获取tag的name，如下：

---

```
soup.head
# <head> <title>The Dormouse's story</title> </head>

soup.title
# <title>The Dormouse's story</title>

soup.body.b
# <b>The Dormouse's story</b>
```

---

**注意：**通过点取属性的放是只能获得当前名字的第一个Tag，同时可以在文档树的tag中多次调用该方法如soup.body.b获取<body>标签中第一个<b>标签。

如果想得到所有的<a>标签，使用方法find\_all()，在前面的Python爬取维基百科等HTML中我们经常用到它+正则表达式的方法。

---

```
soup.find_all('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>]
```

---

**子节点：**在分析HTML过程中通常需要分析tag的子节点，而tag的.contents属性可以将tag的子节点以列表的方式输出。字符串没有.contents属性，因为字符串没有子节点。

---

```
head_tag = soup.head
head_tag
# <head><title>The Dormouse's story</title></head>
```

---



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

title_tag = head_tag.contents[0]
title_tag
# <title>The Dormouse's story</title>
title_tag.contents
# [u'The Dormouse's story']
```

---

**通过tag的 .children 生成器,可以对tag的子节点进行循环:**

---

```
for child in title_tag.children:
    print(child)
    # The Dormouse's story
```

---

**子孙节点: 同样 .descendants 属性可以对所有tag的子孙节点进行递归循环:**

---

```
for child in head_tag.descendants:
    print(child)
    # <title>The Dormouse's story</title>
    # The Dormouse's story
```

---

**父节点: 通过 .parent 属性来获取某个元素的父节点.在例子“爱丽丝”的文档中, <head>标签是<title>标签的父节点, 换句话说就是增加一层标签。**

**注意: 文档的顶层节点比如<html>的父节点是 BeautifulSoup 对象, BeautifulSoup 对象的 .parent 是None。**

---

```
title_tag = soup.title
title_tag
# <title>The Dormouse's story</title>

title_tag.parent
# <head><title>The Dormouse's story</title></head>

title_tag.string.parent
# <title>The Dormouse's story</title>
```

---

**兄弟节点: 因为<b>标签和<c>标签是同一层: 他们是同一个元素的子节点, 所以 <b>和<c>可以被称为兄弟节点。一段文档以标准格式输出时, 兄弟节点有相同的缩进级别.在代码中也可以使用这种关系。**

---

```
sibling_soup = BeautifulSoup("<a><b>text1</b><c>text2</c></b></a>")
print(sibling_soup.prettify())
# <html>
# <body>
# <a>
# <b>
# text1
# </b>
# <c>
# text2
# </c>
```

```
# </a>
# </body>
# </html>
```

---

在文档树中,使用 `.next_sibling` 和 `.previous_sibling` 属性来查询兄弟节点。`<b>` 标签有`.next_sibling` 属性,但是没有`.previous_sibling` 属性,因为`<b>` 标签在同级节点中是第一个。同理`<c>` 标签有`.previous_sibling` 属性,却没有`.next_sibling` 属性:

---

```
sibling_soup.b.next_sibling
# <c>text2</c>
```

```
sibling_soup.c.previous_sibling
# <b>text1</b>
```

---

介绍到这里基本就可以实现我们的BeautifulSoup库爬取网页内容,而网页修改、删除等内容建议大家阅读文档。下一篇文章就再次爬取维基百科的程序语言的内容吧!希望文章对大家有所帮助,如果有错误或不足之处,还请海涵!建议大家阅读官方文档和《Python基础教程》书。

(By: Eastmount 2015-3-25 下午6点 <http://blog.csdn.net/eastmount/>)

👍 点赞 8    ☆ 收藏    ➦ 分享    ...



Eastmount  博客专家

发布了444 篇原创文章 · 获赞 5939 · 访问量 486万+

他的留言板

关注