BeautifulSoup 的使用

```
"""
 -*- coding : utf-8 -*-
 @Author    : quanchenliu
 @Time      : 2024/3/8
 @Function  : BeautifulSoup 的基本用法（方法选择器）
"""
```

一、BeautifulSoup 的基本认识

BeautifulSoup 是Python 的一个 HTML/XML 的解析库。它相比于正则表达式更加简单、易懂、易读，不需要程序员考虑编码问题。

BeautifulSoup 在解析时是依赖解析器的，常见的解析器有：Python标准库、LXML 解析器、html5lib。LXML 解析器是其中性能最优异的：

- 它是唯一**既支持 HTML、又支持 XML** 的解析器；
- 它的速度快、容错能力强。

因此，在本文中，我们**统一使用 LXML 解析器。**

```
from bs4 import BeautifulSoup

# 第一个参数是一个 HTML 字符串，第二个参数是解析器类型：lxml
# 定义并初始化一个 BeautifulSoup 对象
soup = BeautifulSoup(open('practice_BeautifulSoup.html'), 'lxml')
```

二、BeautifulSoup 的使用

1、基本用法——节点选择器：

prettify() 方法：可以将不标准的字符串**以标准格式输出**（自动更正格式的过程是在**初始化** BeautifulSoup 时完成）

```
print(soup.prettify())
```

节点选择器：可以通过直接调用节点的名称、指定节点的属性来选择节点。**一般在结构简单时使用。**

- 通过直接调用节点名称选择节点；

  ```
  print(soup.title, soup.head)
  print(soup.p)                    # 当有多个相同名称的节点时，仅选择第一个匹配的节点
  ```

- 通过获取节点的属性选择节点；

```
# attrs 属性：获取节点所有属性，返回的是一个字典，通过属性名访问即可
print(soup.p.attrs, soup.p.attrs['name'])
print(soup.p['name'])       # 不使用 attrs，直接传入属性名，此时需要注意返回结果的数据类型
print(soup.p.string)        # string 属性：获取节点中的文本信息
```

- 通过 find_all 函数来指定属性条件；

```
print(soup.find_all(attrs={'name': 'dromouse'}))        # 得到满足条件的一个列表
print(soup.find_all(attrs={'class': 'title'}))
```

嵌套选择：在上面的例子中，所有的返回结果都是 **Tag 类型**，而 Tag 类型的对象可以继续调用节点进行下一步选择。即：**每次返回的类型相同，则可以做嵌套选择。**

```
# 在 Tag 类型的基础上进一步进行选择，得到的仍然是 Tag 类型
print(soup.head, soup.head.title)
print(type(soup.head) == type(soup.head.title))
```

2、关联选择器：

在做选择的时候，有时不能一步到位，因此可以考虑：先选中某一节点，然后再以其为基准选择子节点、父节点、子孙节点、祖先节点。

- 直接子节点：
  - **content 属性**：`soup.p.contents`，返回 p 节点的直接子节点所组成的**列表**；
  - **children 属性**：`soup.p.children`，返回包含 p 节点的直接子节点的**生成器**；

```
# enumerate 函数是 Python 中的一个内置函数，用于将一个可迭代对象（如列表、元组、字符串等）组合为一个索引序列，同时返回索引和对应的值
print(soup.p.children)          # children 属性：返回的是一个包含直接子节点的生成器
for i, child in enumerate(soup.p.children):
    print(i, child)             # 通过循环输出相应内容，只迭代生成 直接子节点
```

- 子孙节点：
  - **descendants 属性**：`soup.p.descendants`， 返回包含 p 节点所有子孙节点的**生成器**；
- 父节点与祖先节点：
  - parent 属性：`soup.p.parent`，获取节点元素的**直接父节点**，返回的是父节点的**全部内容**；
  - parents 属性：`soup.p.parents`，获取**所有祖先节点**，返回的是一个**生成器**；
- 兄弟节点：

```
print('Next Sibling', soup.a.next_sibling)      # Next Sibling: 下一个兄弟节点
print('Prev Sibling', soup.a.previous_sibling)  # Prev Sibling: 上一个兄弟节点
print('Next Siblings',
      list(enumerate(soup.a.next_siblings)))    # Next Siblings: 后面所有兄弟节点
print('Prev Siblings',
      list(enumerate(soup.a.previous_siblings)))# Prev Siblings: 前面所有兄弟节点
```

- **enumerate 函数**是 Python 中的一个内置函数，用于将一个可迭代对象（如列表、元组、字符串等）组合为一个索引序列，同时返回**索引和对应的值**

3、方法选择器——`find_all`：

- **通过 name 属性查询**：指定标签的名称，查询所有该名称的节点，返回的是一个**列表**；

```
soup.find_all(name='ul')                        # 查询所有的 ul 节点，返回的是一个列表
print(type(soup.find_all(name='ul')[0]))        # 列表中的每个元素都是 Tag 类型

for ul in soup.find_all(name='ul'):             # 由于都是 Tag 类型，所有可以执行嵌套查询
    print(ul.find_all(name='li'))               # 查询其内部的 li 节点
    for li in ul.find_all(name='li'):           # 遍历每个 li 节点，以获取其中的文本内容
        print(li.string)
```

- **通过传递属性值查询**：

```
print(soup.find_all(attrs={'name': 'dromouse'}))        # 得到满足条件的一个列表
print(soup.find_all(attrs={'class': 'title'}))
```

- **通过匹配节点内部的文本查询**：

```
print(soup.find_all(text=re.compile('F')))
```

- `find` **方法**：返回第一个匹配的元素，**而非所有。**

```
print(soup.find(name='h4'))
```

4、CSS选择器：

详见**三、通过 BeautifulSoup 库来使用 CSS 选择器获取需要的标签/节点**

三、通过 BeautifulSoup 库来使用 CSS 选择器获取需要的标签/节点

```
# 1、直接根据标签名选择节点
items = soup.select('title')
for item in items:
    print(item.string)


# 2、根据 id 选择节点：在 id 前加 # 号，即可选择该标签
items = soup.select('#s-top-left')          # 选取 id 为 s-top-left 的节点
print(items)
items = soup.select('div#s-top-left')       # 选择 id 为 s-top-left 的 div
节点
print(items)


# 3、根据属性选择标签：在属性值前面加 . 作为 select 的参数，即可选中所有符合条件
的标签
items = soup.select('a.mnav1')          # 选择属性值为 mnav1 的 a 标签
for item in items:
    print(item)                         # 每一个 a 标签
    print(item.string)                  # 标签文本信息
    print(item.attrs)                   # 标签所有的属性
    print(item.get('class'))            # 获取指定属性值


# 4、递进式选择标签：父子、子孙关系的节点
items = soup.select('#wrapper > div > a')    # 具有直接父子关系的标签使用
‘>’，返回的是一个列表
for item in items:
    print(item)

items = soup.select('body li span')          # 不具有直接父子关系的标签用空格
表示，返回一个列表
for item in items:
    print(item)


# 5、选择具有指定属性的标签
items = soup.select('[href]')                # 选择具有 href 属性的节点，返
回的是一个列表
for item in items:
    print(item)
```

```
items = soup.select('a[href]')                  # 选择具有 href 属性的 a 标签，返回的是一个列表
for item in items:
    print(item)

items = soup.select('a[href^="https"]')         # 选择 href 属性值以 https 开头的 a 标签
for item in items:
    print(item)

items = soup.select('a[href$="hao123.com"]')  # 选择以 hao123.com 结尾的 a 标签
for item in items:
    print(item)

items = soup.select('a[href*="www"]')           # 选择 href 属性包含 www 的 a 标签
for item in items:
    print(item)

items = soup.select(
    'div#s-top-left, ul#hotsearch-content-wrapper')     # 同时选取多个标签，返回的是一个列表
for item in items:
    print(item)

items = soup.select(
    '[href="https://haokan.baidu.com/?sfrom=baidu-top"]')  # 根据具体的属性值选择标签
for item in items:
    print(item)
```