

目录

前言	1.1
BeautifulSoup简介	1.2
安装	1.3
使用	1.4
常用函数	1.4.1
BeautifulSoup和re详细对比	1.4.2
注意事项和心得	1.5
附录	1.6
参考资料	1.6.1

网页解析利器：BeautifulSoup

- 最新版本： v1.0
- 更新时间： 20200216

鸣谢

感谢我的老婆陈雪雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

简介

整理好用的Python的HTML网页解析器BeautifulSoup的版本选择，为何叫BeautifulSoup，如何安装，如何用其从html网页源码中提取出特定的内容，以及总结常用的函数`find`和`find_all`的具体用法，并用实例解释BeautifulSoup和正则re的效果对比和优缺点对比，以及常见的注意事项和使用期间的心得体会。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- [crifan/html_parse_tool_beautifulsoup: 网页解析利器：BeautifulSoup](#)

如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook_template: demo how to use crifan gitbook template and demo](#)

在线浏览

- [网页解析利器：BeautifulSoup book.crfan.com](#)
- [网页解析利器：BeautifulSoup crifan.github.io](#)

离线下载阅读

- [网页解析利器：BeautifulSoup PDF](#)
- [网页解析利器：BeautifulSoup ePub](#)
- [网页解析利器：BeautifulSoup Mobi](#)

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-16 19:53:58

BeautifulSoup简介

旧教程

之前已有写过一个旧版本的教程，用 docbook 发布的：[Python专题教程：BeautifulSoup详解](#)

现已把其内容整理合并到此新版教程。

对于HTML网页的解析，可以使用[Python中的正则表达式re](#)去提取所需内容。

但是前提（往往是）被解析的html不够复杂，否则正则就很难写，或者说写不出来。

而对于html网页（和xml）的解析，有个专门的库，叫做：

- `BeautifulSoup`
 - 简称： `bs`
 - 最新版本是 `v4`，简称：`bs4`
 - 核心功能：解析 `HTML` 和 `XML`
 - 特点
 - 功能强大
 - 支持语法有问题的 `HTML` 的解析

为何叫 BeautifulSoup

- `BeautifulSoup`
 - 中文直译：`美味的汤`
 - 个人推测是：
 - -》（让人）喝起来很爽（的汤）
 - -》`BeautifulSoup` 的目的就是：
 - 让你从网页中提取内容很方便
 - -》让你像喝美味的汤一样的爽

什么时候会用到BeautifulSoup

`BeautifulSoup` 这个技术所属领域：一般来说属于Python的爬虫相关的技术领域范围内

一般是在：已经用 `requests` 等库或框架，爬取得到了网页源码，然后想要从html源码中提取特定的内容时

往往才会用到这个：`BeautifulSoup`

BeautifulSoup的版本

- 之前：`BeautifulSoup 3`
 - 只支持 `Python 2`

- Python官网（在20200101之后）已不再继续维护 Python 2 了
 - 现在已经是20200216了，大家也都尽量不再用Python 2，而改用 Python 3 了
- 最后版本：3.2.2
 - 截至：2019-10-05
- 安装包：
 - Debian / Ubuntu : python-beautifulsoup
 - Fedora : python-BeautifulSoup
- 最新：BeautifulSoup 4
 - 支持：Python 2 (2.7 +)和 Python 3
 - 最新版本是：Beautiful Soup 4.8.2
 - 截至：2019-12-24
 - 基于BeautifulSoup 4有个：bs4
 - 安装包
 - Debian / Ubuntu
 - Python 2 : python-bs4
 - Python 3 : python3-bs4
 - Fedora
 - python-beautifulsoup4

官网文档

- 主入口
 - Beautiful Soup: We called him Tortoise because he taught us
 - <https://www.crummy.com/software/BeautifulSoup/>
- api文档
 - 英文
 - Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation
 - <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
 - 中文
 - Beautiful Soup 4.4.0 文档 — Beautiful Soup 4.2.0 documentation
 - <https://www.crummy.com/software/BeautifulSoup/bs4/doc.zh/>

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-16 18:50:13

安装

下面主要以，Mac中 Python 3 的 bs4 为例，解释如何安装 BeautifulSoup：

- pip3 install bs4
 - 或： pip install bs4
 - 确保你的 pip 是 Python 3 版本

举例：

```
→ reVsBeautifulSoup pip3 install bs4
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.8.2-py3-none-any.whl (106 kB)
    ██████████ 106 kB 65 kB/s
Collecting soupsieve 1.2
  Downloading soupsieve-1.9.5-py2.py3-none-any.whl (33 kB)
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Created wheel for bs4: filename bs4-0.0.1-py3-none-any.whl size 1272 sha256 603268b090d3
e1b68d5f70078c71c667ef0adab7433943d30bcbdd288161735f
  Stored in directory: /Users/crifan/Library/Caches/pip/wheels/0a/9e/ba/20e5bbc1afef3a491f
0b3bb74d508f99403aab76eda2167ca
Successfully built bs4
Installing collected packages: soupsieve, beautifulsoup4, bs4
Successfully installed beautifulsoup4-4.8.2 bs4-0.0.1 soupsieve-1.9.5
```

查看已安装的版本：

```
→ reVsBeautifulSoup pip3 show bs4
Name: bs4
Version: 0.0.1
Summary: Screen-scraping library
Home-page: https://pypi.python.org/pypi/beautifulsoup4
Author: Leonard Richardson
Author-email: leonardr@segfault.org
License: MIT
Location: /usr/local/lib/python3.7/site-packages
Requires: beautifulsoup4
Required-by:
```

此处已安装的版本是：

- bs4 : 0.0.1
 - 依赖库
 - BeautifulSoup4 : 4.8.2
 - soupsieve : 1.9.5

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-16 18:12:51

使用BeautifulSoup提取html网页内容

接着介绍，如何用 BeautifulSoup 去提取HTML网页中的特定内容。

举例：

假设有html如下：

```
<li class="clearfix"><span>2020-12-31</span>

    <a href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=" href="javasc
ript:void(0)" style="color:#ff0000;" fbfw="外">2019-2020年度全国各地选调生招录、事业单位人才引
进信息汇总——全国各地选调生信息汇总</a>

</li>
<li class="clearfix"><span>2020-12-31</span>

    <a href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=41174064&type=" href="javasc
ript:void(0)" style="color:#ff0000;" fbfw="外">学术就业相关资讯——清华大学学生职业发展指导中心</
a>

</li>
...
[1] [ ] [ ]
```

需要：提取中的 li 中的 a 的 href 值和文本内容 content

用BeautifulSoup提取html的典型步骤

先从html中解析出soup：

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(inputHtml, 'html.parser')
```

再去从soup中提取对应的元素

比如找到所有的 li 的元素

```
foundLiList = soup.find_all('li', attrs={"class": "clearfix"})
```

或：

```
foundLiList = soup.find_all('li', class_="clearfix")
```

找到了 li 后，再去找其中的 a 元素

```
foundA = eachLi.find("a", attrs={"fbfw": "外"})
```

或者是：直接找 `li` 中的 `a` 元素

```
foundAList = soup.find_all('a', attrs={"fbfw": "外"})
```

或者再加上额外限定条件：`a` 中存在属性 `ahref`，且值非空

此处搜索条件中，用上了正则的写法

```
import re
ahrefNonEmptyP = re.compile("\S+")
foundAList = soup.find_all('a', attrs={"fbfw": "外", "ahref": ahrefNonEmptyP})
```

以及也可以加上`style`值的限定：

```
ahrefNonEmptyP = re.compile("\S+") # ahref="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161
&type="
styleColorP = re.compile("color:#[a-zA-Z0-9]+;") # style="color:#ff0000;"
foundAList = soup.find_all('a', attrs={"fbfw": "外", "ahref": ahrefNonEmptyP, "style": style
ColorP})
```

然后对于 `findall` 找到的是列表，每个元素类型是`tag`：

```
class 'bs4.element.Tag'>
```

然后对于每个 `tag` 去通过字典获取属性值，通过 `string` 获取 文本值：

```
ahref = eachA["ahref"]
contentStr = eachA.string
```

即可获取到需要的值：

```
# ahref=/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=
# contentStr=2019-2020年度全国各地选调生招录、事业单位人才引进信息汇总——全国各地选调生信息汇总
```

如此，即是基本的，典型的，BeautifulSoup的soup的用法了。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-16 19:30:31

BeautifulSoup常用函数

soup.find

- 官网文档
 - 中文
 - [find\(name, attrs, recursive, string, **kwargs\)](#)
 - 英文
 - [find\(name, attrs, recursive, string, **kwargs\)](#)

更多实际用法举例：

```
# <h1 class="h1user">crifan</h1>

# method 1: no designate para name
h1userSoup = soup.find("h1", {"class": "h1user"})

# method 2: use para name
h1userSoup = soup.find(name="h1", attrs={"class": "h1user"})

h1userUnicodeStr = h1userSoup.string
```

修改其中内容：

注：只能改（Tag的）中的属性的值，不能改（Tag的）的值本身

```
soup.body.div.h1.string = changedToString

soup.body.div.h1['class'] = "newH1User"
```

soup.findall

- 官网文档
 - 中文
 - [find_all\(name, attrs, recursive, string, **kwargs\)](#)
 - 英文
 - [find_all\(name, attrs, recursive, string, limit, **kwargs\)](#)

默认findall会返回匹配的所有元素

想要限制返回个数，可以加 `limit`

```
soup.find_all('title', limit=2)
```

特殊：

```
find == limit=1 的 findall
```

即：如下是相同含义

```
soup.find_all('title', limit 1)
# [<title>The Dormouse's story</title>]

soup.find('title')
# <title>The Dormouse's story</title>
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-16 19:48:40

BeautifulSoup和re详细对比

下面就通过具体的例子，即：

以之前回复的[这个帖子](#)，来详细解释：

- 如何从HTML中提取所需内容
 - BeautifulSoup的写法
 - `find` 的用法
 - `find_all` 的用法
 - re正则的写法
 - `re.findall` 的用法
 - `re.finditer` 的用法

详细代码如下：

```
# Function: 通过对比说明如何用BeautifulSoup和正则re去提取html中的内容
# 举例所需需求来自此帖:
#     python正则表达式提取空列表-CSDN论坛
#     https://bbs.csdn.net/topics/395845984
# 后已经整理至教程:
#     网页解析利器: BeautifulSoup
#     http://book.crifan.com/books/html_parse_tool_beautifulsoup/website
# Author: Crifan Li
# Update: 20200216

import codecs
from bs4 import BeautifulSoup
import re

respHtmlFile = "responseHtml.html"

# 第一次: 初始化, 保存html到文件
import requests
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.106 Safari/537.36'}
url = 'http://career.cic.tsinghua.edu.cn/xsglxt/f/jyxt/anony/xxfb'
respHtml = requests.get(url, headers=headers).text
# save html to file for later debug
with open(respHtmlFile, "w") as htmlFp:
    htmlFp.write(respHtml)
    htmlFp.close()

# # 后续调试: 从文件中读取html代码, 方便调试
# def loadTextFromFile(fullFilename, fileEncoding="utf-8"):
#     """load file text content from file"""
#     with codecs.open(fullFilename, 'r', encoding=fileEncoding) as fp:
#         allText = fp.read()
#         # logging.debug("Complete load text from %s", fullFilename)
#     return allText
```

```
# respHtml = loadTextFromFile(respHtmlFile)
...
【要处理的html的源码】

<li class="clearfix"><span>2020-12-31</span>

    <a href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=" href="javasc
ript:void(0)" style="color:#ff0000;" fbfw="外">2019-2020年度全国各地选调生招录、事业单位人才引
进信息汇总——全国各地选调生信息汇总</a>

</li>

<li class="clearfix"><span>2020-12-31</span>

    <a href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=41174064&type=" href="javasc
ript:void(0)" style="color:#ff0000;" fbfw="外">学术就业相关资讯——清华大学学生职业发展指导中心<
/a>

</li>
...
【背景解释】
上述html元素结构是：
li
  span
  a
    中文文字

【需求说明】
假如要提取的是：
每个li中a的：
  ahref的链接地址
  中文文字
...
#####
# 用正则re提取html内容
#####
print("=* 20, "用正则re提取html内容", "=* 20)

print("=* 10, "方式1：用re.findall一次性找2个值", "=* 10)

# 方式1：匹配整个li的部分
wholeLiP = '<li\s+class="clearfix"><span>.*?</span>\s*<a\s+href="(.*?)"\s+href="javasc
ript:void(0\)"\s+style="color:.*?;"\s+fbfw="外">(.*?)</a>\s*</li>'
print("wholeLiP=%s" % wholeLiP)

findAllMatchedTupleList = re.findall(wholeLiP, respHtml, re.S)
```

```

# # 方式2: 只匹配a的部分
# onlyAP = '<a\s+ahref="(.*?)"\s+href="javascript:void\(0\)"\s+style="color:.*?;"\s+fbfw="外">(.*?)</a>'
# print("onlyAP=%s" % onlyAP)
# foundAllMatchedTupleList = re.findall(onlyAP, respHtml, re.S)

# print("foundAllMatchedTupleList=%s" % foundAllMatchedTupleList)
for curIdx, eachTuple in enumerate(foundAllMatchedTupleList):
    # 之前正则中有2个括号, 对应2个group组: ahref="(.*?)" 和 >(.*?)</a>
    # -> 此处匹配到的值是个tuple元素, 是2个元素, 分别对应着之前的2个group
    print("-" * 10, "[%d]" % curIdx, "-" * 10)
    print("type(eachTuple)=%s" % type(eachTuple))
    # type(eachTuple)=<class 'tuple'>
    (matchedFirstGroupStr, matchedSecondGroupStr) = eachTuple
    ahrefValue = matchedFirstGroupStr
    contentStrValue = matchedSecondGroupStr
    print("ahrefValue=%s, contentStrValue=%s" % (ahrefValue, contentStrValue))
    # ahrefValue=/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=, contentStrValue=2019-
2020年度全国各地选调生招录、事业单位人才引进信息汇总——全国各地选调生信息汇总

print("=" * 10, "方式2: 用re.finditer找, 支持更多可能性", "=" * 10)

foundAllMatchObjectList = re.finditer(wholeLiP, respHtml, re.S)
# foundAllMatchObjectList=<callable_iterator object at 0x10f4274e0>
print("foundAllMatchObjectList=%s" % foundAllMatchObjectList)
for curIdx, eachMatchObject in enumerate(foundAllMatchObjectList):
    print("-" * 10, "[%d]" % curIdx, "-" * 10)
    # re.finditer返回的是Match Objects的list
    print("type(eachMatchObject)=%s" % type(eachMatchObject))
    # type(eachMatchObject)=<class 're.Match'>
    group1Value = eachMatchObject.group(1)
    group2Value = eachMatchObject.group(2)
    ahrefValue = group1Value
    contentStrValue = group2Value
    print("ahrefValue=%s, contentStrValue=%s" % (ahrefValue, contentStrValue))
    # ahrefValue=/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=, contentStrValue=2019-
2020年度全国各地选调生招录、事业单位人才引进信息汇总——全国各地选调生信息汇总

# 额外说明:
# 如果你前面正则中是named group带命名的组, 比如:
# ... ahref="(P<ahref>.*)" ... >(P<contentStr>.*)</a>
# 那么也可以通过group name组名去获取值:
# ahrefValue = eachMatchObject.group("ahref")
# contentStrValue = eachMatchObject.group("contentStr")

#####
# 用BeautifulSoup提取html内容
#####
print("=" * 20, "用BeautifulSoup提取html内容", "=" * 20)

soup = BeautifulSoup(respHtml, 'html.parser')
# print("soup=%s" % soup)

```

```

print("=-*10, "方式1: 先找外层的li, 再去li中找其中的a", "-*10)

# 找li的方式1: 通过attrs指定属性
# foundLiList = soup.find_all('li', attrs={"class": "clearfix"})
# 找li的方式2: class 在Python中是保留字 -> BeautifulSoup >4.1.1后, 用class_指定CSS的类名
foundLiList = soup.find_all('li', class_="clearfix")
# print("foundLiList=%s" % foundLiList)
for curIdx, eachLi in enumerate(foundLiList):
    print("-" * 10, "[%d]" % curIdx, "-" * 10)
    print("type(eachLi)=%s" % type(eachLi))
    # type(eachLi)=<class 'bs4.element.Tag'>
    foundA = eachLi.find("a", attrs={"fbfw": "外"})
    print("foundA=%s" % foundA)
    # foundA=<a href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=" fbfw="外" hr
    ef="javascript:void(0)" style="color:#ff0000;">2019-2020年度全国各地选调生招录、事业单位人才引
    进信息汇总——全国各地选调生信息汇总</a>
    if foundA:
        href = foundA["href"]
        print("href=%s" % href)
        # href=/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=
        contentStr = foundA.string
        print("contentStr=%s" % contentStr)
    # contentStr=2019-2020年度全国各地选调生招录、事业单位人才引进信息汇总——全国各地选调生信息
    汇总

print("=-*10, "方式2: 直接找a, 加上限定条件", "-*10)

# foundAList = soup.find_all('a', attrs={"fbfw": "外"}) # 只加上一个fbfw的限定条件, 此处也是可
以的
hrefNonEmptyP = re.compile("\S+") # href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161
&type="
print("hrefNonEmptyP=%s" % hrefNonEmptyP)
# foundAList = soup.find_all('a', attrs={"fbfw": "外", "href": hrefNonEmptyP})
styleColorP = re.compile("color:#[a-zA-Z0-9]+;") # style="color:#ff0000;"
print("styleColorP=%s" % styleColorP)
foundAList = soup.find_all('a', attrs={"fbfw": "外", "href": hrefNonEmptyP, "style": style
ColorP})
# print("foundAList=%s" % foundAList)
for curIdx, eachA in enumerate(foundAList):
    print("-" * 10, "[%d]" % curIdx, "-" * 10)
    print("type(eachA)=%s" % type(eachA))
    # type(eachA)=<class 'bs4.element.Tag'>
    href = eachA["href"]
    print("href=%s" % href)
    # href=/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=
    contentStr = eachA.string
    print("contentStr=%s" % contentStr)
    # contentStr=2019-2020年度全国各地选调生招录、事业单位人才引进信息汇总——全国各地选调生信息汇总

#####
# 对比: re vs BeautifulSoup
#####

```

```
print("=="*20, "对比: re vs BeautifulSoup", "=="*20)

reVsBeautifulSoup = """
re正则的缺点:
万一html源代码改动了, 即便改动很小, 则之前已有的re正则表达式就失效了
举例:
只是a的属性的顺序变化一点点
从
<a href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=" href="javascript:void(0)
" style="color:#ff0000;" fbew="外">2019-2020年度全国各地选调生招录、事业单位人才引进信息汇总——
-全国各地选调生信息汇总</a>
改为:
<a href="javascript:void(0)" href="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161&type=
" fbew="外" style="color:#ff0000;">2019-2020年度全国各地选调生招录、事业单位人才引进信息汇总——
-全国各地选调生信息汇总</a>
之前正则:
'<a\s+ahref="(.*?)"\s+href="javascript:void\((0\)"\s+style="color:.*?;"\s+fbew="外">(.*?)<
/a>' 
就无效了, 就要再去改为:
'<a\s+href="javascript:void\((0\)"\s+ahref="(.*?)"\s+fbew="外"\s+style="color:.*?;">(.*?)<
/a>'
才可以匹配到。
"""

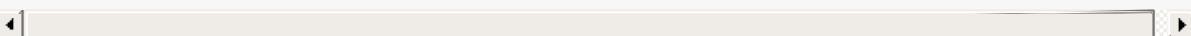
更别说, 万一html中代码有其他更大的变化
甚至是部分语法不规范的html代码, re正则根本就没法写, 因为太复杂, 复杂到写不出来
```

BeautifulSoup的优点:
与之相对: 上述的, html代码的小改动, 比如属性值出现的顺序不同
甚至大点的变化, 多出其他属性值
甚至部分语法不规范的html代码, BeautifulSoup都可以很好的内部处理掉
而之前的代码, 比如:
`soup.find_all('a', attrs={"fbew": "外", "ahref": nonEmptyP})`
都可以很好的继续工作, 而无需改动。

汇总起来就是:

re	
性能: 好	
支持html程度: 有限	
仅限于不是很复杂的, 比较规整的html	
BeautifulSoup	
性能: 中等	
支持html程度: 很好	
不仅支持复杂的html, 还支持html内部元素和位置变化	
对于不规范的html也有很好的支持	

```
print(reVsBeautifulSoup)
```



bs 和 re 函数返回变量类型

此处调试期间，可以看到对应变量的类型：

- 正则re

- re.findall 返回的是匹配的元祖 tuple 的列表：<class 'tuple'>的 list

```

    BeautifulSoup: <class 'bs4.BeautifulSoup'>
    codecs: <module 'codecs' from '/usr/local/lib/python3.7/_codecs.py'>
    foundAllMatchedTupleList: [[{'wholeIP': '<cl><s>class="clearfix"><span>.*</span><s><a><s>href="(.*)"\><s>+<a><s>href="javascript:void(0)"><s>+<s>style="color:.*;"><s>fbfw="2">(.*)</s>', 'contentStrValue': '蒙盈投资（量化）2020年春季校园招聘——上海蒙盈投资管理有限公司', 'type': '1'}, {'wholeIP': '<cl><s>class="clearfix"><span>.*</span><s><a><s>href="(.*)"\><s>+<a><s>href="javascript:void(0)"><s>+<s>style="color:.*;"><s>fbfw="2">(.*)</s>', 'contentStrValue': '蒙盈投资（量化）2020年春季校园招聘——上海蒙盈投资管理有限公司', 'type': '1'}, {"...}], 2]
  
```

- re.finditer 返回的是匹配对象 Match Object 的列表：<class 're.Match'>的 list

```

    BeautifulSoup: <class 'bs4.BeautifulSoup'>
    dhrefValue: '<xsglx/f/jyxt/amon/showZwxz?zpxxid=152542680&type='
    codecs: <module 'codecs' from '/usr/local/lib/python3.7/_codecs.py'>
    contentStrValue: '蒙盈投资（量化）2020年春季校园招聘——上海蒙盈投资管理有限公司'
    curIdx: 19
    eachTuple: ('<xsglx/f/jyxt/amon...68&type=')
    foundAllMatchObjectList: <callable_iterator object at 0x107aa04ac>
    foundAllMatchedTupleList: [[{'wholeIP': '<cl><s>class="clearfix"><span>.*</span><s><a><s>href="(.*)"\><s>+<a><s>href="javascript:void(0)"><s>+<s>style="color:.*;"><s>fbfw="2">(.*)</s>', 'contentStrValue': '蒙盈投资（量化）2020年春季校园招聘——上海蒙盈投资管理有限公司', 'type': '1'}, {"...}], 2]
  
```

- BeautifulSoup

- BeautifulSoup 返回的 soup 变量的详情：

The screenshot shows the PyCharm IDE interface with two code editors side-by-side. The left editor contains Python code for 'reVsBeautifulSoup.py' using BeautifulSoup to parse an XML file. The right editor contains the corresponding Java code for 'reVsBeautifulSoup.java'. Both files are titled 'BeautifulSoup'. The Python code uses the BeautifulSoup module to parse an XML document and extract specific elements like 'contentStrValue'. The Java code does the same using the DOMParser class. Below the code editors is a 'Python Debug Console' window showing the execution of the code and its output.

```
reVsBeautifulSoup.py <-- reVsBeautifulSoup
reVsBeautifulSoup.py <-- Untitled-1 •
```

```
reVsBeautifulSoup.py
119 #####用BeautifulSoup提取html内容#####
120 #用BeautifulSoup提取html内容
121
122 print("=*20, "用BeautifulSoup提取html内容, "=*20)
123
124 soup = BeautifulSoup(resptml, 'html.parser')
125 # p \n<html xmlns="http://www.w3.org/1999/xhtml">
126 #   ASCII_SPACES: '\n\t\r\x0c\x1c'
127 #> DEFAULT_BUILDER_FEATURES: ['html', 'fast']
128 #> NO_PARSER_SPECIFIED_WARNING: 'No parser was <
129 #> ROOT_TAG_NAME: ['document']
130 #>     :> clearfix">
131 #>     > attrs: {}
132 #>     > builder: <bs4.builder._HTMLParser>
133 #>     > can_be_empty_element: False
134 #>     > cdata_list_attributes: {<'class', 'access
135 #>     > children: <list_iterator object at 0x10e0ed
136 #>     > contains_replacement_characters: False
137 #>     > contents: ['\n', '<html xmlns="http://...dy/>
138 #>     > currentTag: <ndhtml xmlns="http://www.w3.org/
139 #>     > currentData: []
140 #>     > declared_html_encoding: None
141 #>     > descendants: <generator object Tag.descendant
142 #>     > element_classes: []
143 #>     > hidden: 1
144 #>     > isSelfClosing: False
145
146 type(eachMatchObject)<=>class 're.Match'
147 hrefValue='/xslt/f/jyxt/anony/showZwxz?zxpid=152542675&type=, contentStrValue=OPPO 2020届春季校园招聘——OPPO广东
148 移动通信有限公司
149 [18] -----
150 type(eachMatchObject)<=>class 're.Match'
151 hrefValue='/xslt/f/jyxt/anony/showZwxz?zxpid=152542680&type=, contentStrValue=蒙玺投资（量化）2020年春季校园招聘—
152 上海蒙玺投资管理有限公司
153 ====== 用BeautifulSoup提取html内容 ======
```

```
2: Python Debug Console
```

```
断点
    ■ Raised Exceptions
    ☐ Uncaught Exceptions
    ● reVsBeautifulSoup
    reVsBeautifulSoup.py 127/1
```

- 而 BeautifulSoup 的 `find_all` 返回的是标签元素的列表：`<class 'bs4.element.Tag'>` 的 `list`

The screenshot shows the PyCharm IDE interface with the following details:

- Top Bar:** Shows the project name "reVsBeautifulSoup.py" and the current file "reVsBeautifulSoup.py".
- Left Sidebar:** Contains sections for "运行和调试" (Run and Debug), "变量" (Variables), "监视" (Watch), "调用堆栈" (Call Stack), and "断点" (Breakpoints). The "断点" section shows a list of breakpoints, with one for "ahrefValue/xs" highlighted.
- Code Editor:** Displays the "reVsBeautifulSoup.py" file. The cursor is at line 110, which contains the code `# curlid: 19`. A vertical red bar highlights the line where the breakpoint is set.
- Search Results:** A floating search panel titled "BREKPOINT 已暂停" shows results for "curlid: 19". It lists several matches from the code, including lines 110, 121, 131, 141, 151, 161, 171, 181, 191, 201, 211, 221, 231, 241, 251, 261, 271, 281, 291, 301, 311, 321, 331, 341, 351, 361, 371, 381, 391, 401, 411, 421, 431, 441, 451, 461, 471, 481, 491, 501, 511, 521, 531, 541, 551, 561, 571, 581, 591, 601, 611, 621, 631, 641, 651, 661, 671, 681, 691, 701, 711, 721, 731, 741, 751, 761, 771, 781, 791, 801, 811, 821, 831, 841, 851, 861, 871, 881, 891, 901, 911, 921, 931, 941, 951, 961, 971, 981, 991, 1001, 1011, 1021, 1031, 1041, 1051, 1061, 1071, 1081, 1091, 1101, 1111, 1121, 1131, 1141, 1151, 1161, 1171, 1181, 1191, 1201, 1211, 1221, 1231, 1241, 1251, 1261, 1271, 1281, 1291, 1301, 1311, 1321, 1331, 1341, 1351, 1361, 1371, 1381, 1391, 1401, 1411, 1421, 1431, 1441, 1451, 1461, 1471, 1481, 1491, 1501, 1511, 1521, 1531, 1541, 1551, 1561, 1571, 1581, 1591, 1601, 1611, 1621, 1631, 1641, 1651, 1661, 1671, 1681, 1691, 1701, 1711, 1721, 1731, 1741, 1751, 1761, 1771, 1781, 1791, 1801, 1811, 1821, 1831, 1841, 1851, 1861, 1871, 1881, 1891, 1901, 1911, 1921, 1931, 1941, 1951, 1961, 1971, 1981, 1991, 2001, 2011, 2021, 2031, 2041, 2051, 2061, 2071, 2081, 2091, 2101, 2111, 2121, 2131, 2141, 2151, 2161, 2171, 2181, 2191, 2201, 2211, 2221, 2231, 2241, 2251, 2261, 2271, 2281, 2291, 2301, 2311, 2321, 2331, 2341, 2351, 2361, 2371, 2381, 2391, 2401, 2411, 2421, 2431, 2441, 2451, 2461, 2471, 2481, 2491, 2501, 2511, 2521, 2531, 2541, 2551, 2561, 2571, 2581, 2591, 2601, 2611, 2621, 2631, 2641, 2651, 2661, 2671, 2681, 2691, 2701, 2711, 2721, 2731, 2741, 2751, 2761, 2771, 2781, 2791, 2801, 2811, 2821, 2831, 2841, 2851, 2861, 2871, 2881, 2891, 2901, 2911, 2921, 2931, 2941, 2951, 2961, 2971, 2981, 2991, 3001, 3011, 3021, 3031, 3041, 3051, 3061, 3071, 3081, 3091, 3101, 3111, 3121, 3131, 3141, 3151, 3161, 3171, 3181, 3191, 3201, 3211, 3221, 3231, 3241, 3251, 3261, 3271, 3281, 3291, 3301, 3311, 3321, 3331, 3341, 3351, 3361, 3371, 3381, 3391, 3401, 3411, 3421, 3431, 3441, 3451, 3461, 3471, 3481, 3491, 3501, 3511, 3521, 3531, 3541, 3551, 3561, 3571, 3581, 3591, 3601, 3611, 3621, 3631, 3641, 3651, 3661, 3671, 3681, 3691, 3701, 3711, 3721, 3731, 3741, 3751, 3761, 3771, 3781, 3791, 3801, 3811, 3821, 3831, 3841, 3851, 3861, 3871, 3881, 3891, 3901, 3911, 3921, 3931, 3941, 3951, 3961, 3971, 3981, 3991, 4001, 4011, 4021, 4031, 4041, 4051, 4061, 4071, 4081, 4091, 4101, 4111, 4121, 4131, 4141, 4151, 4161, 4171, 4181, 4191, 4201, 4211, 4221, 4231, 4241, 4251, 4261, 4271, 4281, 4291, 4301, 4311, 4321, 4331, 4341, 4351, 4361, 4371, 4381, 4391, 4401, 4411, 4421, 4431, 4441, 4451, 4461, 4471, 4481, 4491, 4501, 4511, 4521, 4531, 4541, 4551, 4561, 4571, 4581, 4591, 4601, 4611, 4621, 4631, 4641, 4651, 4661, 4671, 4681, 4691, 4701, 4711, 4721, 4731, 4741, 4751, 4761, 4771, 4781, 4791, 4801, 4811, 4821, 4831, 4841, 4851, 4861, 4871, 4881, 4891, 4901, 4911, 4921, 4931, 4941, 4951, 4961, 4971, 4981, 4991, 5001, 5011, 5021, 5031, 5041, 5051, 5061, 5071, 5081, 5091, 5011, 5021, 5031, 5041, 5051, 5061, 5071, 5081, 5091, 5101, 5111, 5121, 5131, 5141, 5151, 5161, 5171, 5181, 5191, 5111, 5121, 5131, 5141, 5151, 5161, 5171, 5181, 5191, 5201, 5211, 5221, 5231, 5241, 5251, 5261, 5271, 5281, 5291, 5211, 5221, 5231, 5241, 5251, 5261, 5271, 5281, 5291, 5301, 5311, 5321, 5331, 5341, 5351, 5361, 5371, 5381, 5391, 5311, 5321, 5331, 5341, 5351, 5361, 5371, 5381, 5391, 5401, 5411, 5421, 5431, 5441, 5451, 5461, 5471, 5481, 5491, 5411, 5421, 5431, 5441, 5451, 5461, 5471, 5481, 5491, 5501, 5511, 5521, 5531, 5541, 5551, 5561, 5571, 5581, 5591, 5511, 5521, 5531, 5541, 5551, 5561, 5571, 5581, 5591, 5601, 5611, 5621, 5631, 5641, 5651, 5661, 5671, 5681, 5691, 5611, 5621, 5631, 5641, 5651, 5661, 5671, 5681, 5691, 5701, 5711, 5721, 5731, 5741, 5751, 5761, 5771, 5781, 5791, 5711, 5721, 5731, 5741, 5751, 5761, 5771, 5781, 5791, 5801, 5811, 5821, 5831, 5841, 5851, 5861, 5871, 5881, 5891, 5811, 5821, 5831, 5841, 5851, 5861, 5871, 5881, 5891, 5901, 5911, 5921, 5931, 5941, 5951, 5961, 5971, 5981, 5991, 5911, 5921, 5931, 5941, 5951, 5961, 5971, 5981, 5991, 6001, 6011, 6021, 6031, 6041, 6051, 6061, 6071, 6081, 6091, 6011, 6021, 6031, 6041, 6051, 6061, 6071, 6081, 6091, 6101, 6111, 6121, 6131, 6141, 6151, 6161, 6171, 6181, 6191, 6111, 6121, 6131, 6141, 6151, 6161, 6171, 6181, 6191, 6201, 6211, 6221, 6231, 6241, 6251, 6261, 6271, 6281, 6291, 6211, 6221, 6231, 6241, 6251, 6261, 6271, 6281, 6291, 6301, 6311, 6321, 6331, 6341, 6351, 6361, 6371, 6381, 6391, 6311, 6321, 6331, 6341, 6351, 6361, 6371, 6381, 6391, 6401, 6411, 6421, 6431, 6441, 6451, 6461, 6471, 6481, 6491, 6411, 6421, 6431, 6441, 6451, 6461, 6471, 6481, 6491, 6501, 6511, 6521, 6531, 6541, 6551, 6561, 6571, 6581, 6591, 6511, 6521, 6531, 6541, 6551, 6561, 6571, 6581, 6591, 6601, 6611, 6621, 6631, 6641, 6651, 6661, 6671, 6681, 6691, 6611, 6621, 6631, 6641, 6651, 6661, 6671, 6681, 6691, 6701, 6711, 6721, 6731, 6741, 6751, 6761, 6771, 6781, 6791, 6711, 6721, 6731, 6741, 6751, 6761, 6771, 6781, 6791, 6801, 6811, 6821, 6831, 6841, 6851, 6861, 6871, 6881, 6891, 6811, 6821, 6831, 6841, 6851, 6861, 6871, 6881, 6891, 6901, 6911, 6921, 6931, 6941, 6951, 6961, 6971, 6981, 6991, 6911, 6921, 6931, 6941, 6951, 6961, 6971, 6981, 6991, 7001, 7011, 7021, 7031, 7041, 7051, 7061, 7071, 7081, 7091, 7011, 7021, 7031, 7041, 7051, 7061, 7071, 7081, 7091, 7101, 7111, 7121, 7131, 7141, 7151, 7161, 7171, 7181, 7191, 7111, 7121, 7131, 7141, 7151, 7161, 7171, 7181, 7191, 7201, 7211, 7221, 7231, 7241, 7251, 7261, 7271, 7281, 7291, 7211, 7221, 7231, 7241, 7251, 7261, 7271, 7281, 7291, 7301, 7311, 7321, 7331, 7341, 7351, 7361, 7371, 7381, 7391, 7311, 7321, 7331, 7341, 7351, 7361, 7371, 7381, 7391, 7401, 7411, 7421, 7431, 7441, 7451, 7461, 7471, 7481, 7491, 7411, 7421, 7431, 7441, 7451, 7461, 7471, 7481, 7491, 7501, 7511, 7521, 7531, 7541, 7551, 7561, 7571, 7581, 7591, 7511, 7521, 7531, 7541, 7551, 7561, 7571, 7581, 7591, 7601, 7611, 7621, 7631, 7641, 7651, 7661, 7671, 7681, 7691, 7611, 7621, 7631, 7641, 7651, 7661, 7671, 7681, 7691, 7701, 7711, 7721, 7731, 7741, 7751, 7761, 7771, 7781, 7791, 7711, 7721, 7731, 7741, 7751, 7761, 7771, 7781, 7791, 7801, 7811, 7821, 7831, 7841, 7851, 7861, 7871, 7881, 7891, 7811, 7821, 7831, 7841, 7851, 7861, 7871, 7881, 7891, 7901, 7911, 7921, 7931, 7941, 7951, 7961, 7971, 7981, 7991, 7911, 7921, 7931, 7941, 7951, 7961, 7971, 7981, 7991, 8001, 8011, 8021, 8031, 8041, 8051, 8061, 8071, 8081, 8091, 8011, 8021, 8031, 8041, 8051, 8061, 8071, 8081, 8091, 8101, 8111, 8121, 8131, 8141, 8151, 8161, 8171, 8181, 8191, 8111, 8121, 8131, 8141, 8151, 8161, 8171, 8181, 8191, 8201, 8211, 8221, 8231, 8241, 8251, 8261, 8271, 8281, 8291, 8211, 8221, 8231, 8241, 8251, 8261, 8271, 8281, 8291, 8301, 8311, 8321, 8331, 8341, 8351, 8361, 8371, 8381, 8391, 8311, 8321, 8331, 8341, 8351, 8361, 8371, 8381, 8391, 8401, 8411, 8421, 8431, 8441, 8451, 8461, 8471, 8481, 8491, 8411, 8421, 8431, 8441, 8451, 8461, 8471, 8481, 8491, 8501, 8511, 8521, 8531, 8541, 8551, 8561, 8571, 8581, 8591, 8511, 8521, 8531, 8541, 8551, 8561, 8571, 8581, 8591, 8601, 8611, 8621, 8631, 8641, 8651, 8661, 8671, 8681, 8691, 8611, 8621, 8631, 8641, 8651, 8661, 8671, 8681, 8691, 8701, 8711, 8721, 8731, 8741, 8751, 8761, 8771, 8781, 8791, 8711, 8721, 8731, 8741, 8751, 8761, 8771, 8781, 8791, 8801, 8811, 8821, 8831, 8841, 8851, 8861, 8871, 8881, 8891, 8811, 8821, 8831, 8841, 8851, 8861, 8871, 8881, 8891, 8901, 8911, 8921, 8931, 8941, 8951, 8961, 8971, 8981, 8991, 8911, 8921, 8931, 8941, 8951, 8961, 8971, 8981, 8991, 9001, 9011, 9021, 9031, 9041, 9051, 9061, 9071, 9081, 9091, 9011, 9021, 9031, 9041, 9051, 9061, 9071, 9081, 9091, 9101, 9111, 9121, 9131, 9141, 9151, 9161, 9171, 9181, 9191, 9111, 9121, 9131, 9141, 9151, 9161, 9171, 9181, 9191, 9201, 9211, 9221, 9231, 9241, 9251, 9261, 9271, 9281, 9291, 9211, 9221, 9231, 9241, 9251, 9261, 9271, 9281, 9291, 9301, 9311, 9321, 9331, 9341, 9351, 9361, 9371, 9381, 9391, 9311, 9321, 9331, 9341, 9351, 9361, 9371, 9381, 9391, 9401, 9411, 9421, 9431, 9441, 9451, 9461, 9471, 9481, 9491, 9411, 9421, 9431, 9441, 9451, 9461, 9471, 9481, 9491, 9501, 9511, 9521, 9531, 9541, 9551, 9561, 9571, 9581, 9591, 9511, 9521, 9531, 9541, 9551, 9561, 9571, 9581, 9591, 9601, 9611, 9621, 9631, 9641, 9651, 9661, 9671, 9681, 9691, 9611, 9621, 9631, 9641, 9651, 9661, 9671, 9681, 9691, 9701, 9711, 9721, 9731, 9741, 9751, 9761, 9771, 9781, 9791, 9711, 9721, 9731, 9741, 9751, 9761, 9771, 9781, 9791, 9801, 9811, 9821, 9831, 9841, 9851, 9861, 9871, 9881, 9891, 9811, 9821, 9831, 9841, 9851, 9861, 9871, 9881, 9891, 9901, 9911, 9921, 9931, 9941, 9951, 9961, 9971, 9981, 9991, 9911, 9921, 9931, 9941, 9951, 9961, 9971, 9981, 9991, 10001, 10011, 10021, 10031, 10041, 10051, 10061, 10071, 10081, 10091, 10011, 10021, 10031, 10041, 10051, 10061, 10071, 10081, 10091, 10101, 10111, 10121, 10131, 10141, 10151, 10161, 10171, 10181, 10191, 10111, 10121, 10131, 10141, 10151, 10161, 10171, 10181, 10191, 10201, 10211, 10221, 10231, 10241, 10251, 10261, 10271, 10281, 10291, 10211, 10221, 10231, 10241, 10251, 10261, 10271, 10281, 10291, 10301, 10311, 10321, 10331, 10341, 10351, 10361, 10371, 10381, 10391, 10311, 10321, 10331, 10341, 10351, 10361, 10371, 10381, 10391, 10401, 10411, 10421, 10431, 10441, 10451, 10461, 10471, 10481, 10491, 10411, 10421, 10431, 10441, 10451, 10461, 10471, 10481, 10491, 10501, 10511, 10521, 10531, 10541, 10551, 10561, 10571, 10581, 10591, 10511, 10521, 10531, 10541, 10551, 10561, 10571, 10581, 10591, 10601, 10611, 10621, 10631, 10641, 10651, 10661, 10671, 10681, 10691, 10611, 10621, 10631, 10641, 10651, 10661, 10671, 10681, 10691, 10701, 10711, 10721, 10731, 10741, 10751, 10761, 10771, 10781, 10791, 10711, 10721, 10731, 10741, 10751, 10761, 10771, 10781, 10791, 10801, 10811, 10821, 10831, 10841, 10851, 10861, 10871, 10881, 10891, 10811, 10821, 10831, 10841, 10851, 10861, 10871, 10881, 10891, 10901, 10911, 10921, 10931, 10941, 10951, 10961, 10971, 10981, 10991, 10911, 10921, 10931, 10941, 10951, 10961, 10971, 10981, 10991, 11001, 11011, 11021, 11031, 11041, 11051, 11061, 11071, 11081, 11091, 11011, 11021, 11031, 11041, 11051, 11061, 11071, 11081, 11091, 11101, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11201, 11211, 11221, 11231, 11241, 11251, 11261, 11271, 11281, 11291, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11201, 11211, 11221, 11231, 11241, 11251, 11261, 11271, 11281, 11291, 11301, 11311, 11321, 11331, 11341, 11351, 11361, 11371, 11381, 11391, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11201, 11211, 11221, 11231, 11241, 11251, 11261, 11271, 11281, 11291, 11301, 11311, 11321, 11331, 11341, 11351, 11361, 11371, 11381, 11391, 11401, 11411, 11421, 11431, 11441, 11451, 11461, 11471, 11481, 11491, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11201, 11211, 11221, 11231, 11241, 11251, 11261, 11271, 11281, 11291, 11301, 11311, 11321, 11331, 11341, 11351, 11361, 11371, 11381, 11391, 11401, 11411, 11421, 11431, 11441, 11451, 11461, 11471, 11481, 11491, 11501, 11511, 11521, 11531, 11541, 11551, 11561, 11571, 11581, 11591, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11201, 11211, 11221, 11231, 11241, 11251, 11261, 11271, 11281, 11291, 11301, 11311, 11321, 11331, 11341, 11351, 11361, 11371, 11381, 11391, 11401, 11411, 11421, 11431, 11441, 11451, 11461, 11471, 11481, 11491, 11501, 11511, 11521, 11531, 11541, 11551, 11561, 11571, 11581, 11591, 11601, 11611, 11621, 11631, 11641, 11651, 11661, 11671, 11681, 11691, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11201, 11211, 11221, 11231, 11241, 11251, 11261, 11271, 11281, 11291, 11301, 11311, 11321, 11331, 11341, 11351, 11361, 11371, 11381, 11391, 11401, 11411, 11421, 11431, 11441, 11451, 11461, 11471, 11481, 11491, 11501, 11511, 11521, 11531, 11541, 11551, 11561, 11571, 11581, 11591, 11601, 11611, 11621, 11631, 11641, 11651, 11661, 11671, 11681, 11691, 11701, 11711, 11721, 11731, 11741, 11751, 11761, 11771, 11781, 11791, 11111, 11121, 11131, 11141, 11151, 11161, 11171, 11181, 11191, 11201, 11211, 11221, 11231, 11241, 11251, 11261, 11271, 11281, 11291, 11301, 11311, 11321, 11331, 11341, 11351, 113

- 而 BeautifulSoup 的 find 返回的是单个标签元素: <class 'bs4.element.Tag'>

```

reVsBeautifulSoup.py — reVsBeautifulSoup
124     soup = BeautifulSoup(open('2019-2020年度全国各地选调生招录.html'), 'lxml')
125     # print("soup")
126
127     print("=*10, "方式1: 先找外层的li, 再去li中找其中的a", "*=10)
128
129     # 找li的方式1 通过attrs指定属性
130     # foundLiList = soup.findall('li', attrs={"class": "clearfix"})
131     # 找li的方式2 class 在Python中是保留字 -> BeautifulSoup >4.1.1后, 用class_指定
132     foundLiList = soup.findall('li', class_="clearfix")
133     # print("foundLiList=%s" % foundLiList)
134     for curIdx, eachLi in enumerate(foundLiList):
135         print("-*10, [%d]" % curIdx, "*=10)
136         if type(eachLi) == class_:
137             print("type(eachLi)=%s" % type(eachLi))
138             # type(eachLi)==class_>bs4.element.Tag>
139             foundA = eachLi.find("a", attrs={"fbfw": "外"})
140             print("a href=%s" % foundA)
141             if f:
142                 com_be_empty_element: False
143                 ah: cdata_list_attributes: {'*': ['class', 'accesskey', 'dropzone'], 'a':...}
144                 pr: children: <list_iterator object at 0x107ce1390>
145                 co: contents: ['2019-2020年度全国各地选调生招录...各地选调生信息汇总']
146                 pr: descendants: <generator object Tag.descendants at 0x107ce2138>
147                 hidden: False
148                 isSelfClosing: False
149                 is_empty_element: False
150                 known_xml: False
151                 name: 'a'
152                 namespace: None
153                 next: '2019-2020年度全国各地选调生招录、事业单位人才引进信息汇总——全国各地选调生信息汇总——全国'
154                 nextSibling: '\n'
155             > 监视
156             > 调用堆栈
157             <module>
158             reVsBeautifulSoup.py [39:1]
159             print("a href=%s" % foundA)
160             if f:
161                 com_be_empty_element: False
162                 ah: cdata_list_attributes: {'*': ['class', 'accesskey', 'dropzone'], 'a':...}
163                 pr: children: <list_iterator object at 0x107cd516>
164                 co: contents: ['2019-2020年度全国各地选调生招录...各地选调生信息汇总']
165                 pr: descendants: <generator object Tag.descendant at 0x107cd516>
166                 hidden: False
167                 isSelfClosing: False
168                 is_empty_element: False
169                 known_xml: False
170                 name: 'a'
171                 type: eachLi: type=<class 'bs4.element.Tag'>
172                 namespace: None
173                 next: '2019-2020年度全国各地选调生招录、事业单位人才引进信息汇总——全国各地选调生信息汇总——全国'
174                 nextSibling: '\n'
175             =====
176             type(eachMat):
177             ahrefValue=/
178             师——天津新
179             =====
180             type(eachMat):
181             ahrefValue=/
182             (量化) 2020
183             =====
184             type(eachLi):
185             ahrefValue=/
186             next_sibling: '\n'
187             =====
188             type(eachLi):
189             ahrefValue=/
190             next_siblings: <generator object PageElement.next_siblings at 0x107cd516>
191             type(eachLi)<class 'bs4.element.Tag'>
192             =====
193             type(eachLi):
194             ahrefValue=/
195             next_sibling: '\n'
196             =====
197             type(eachLi):
198             ahrefValue=/
199             next_sibling: '\n'
200             =====
201             type(eachLi):
202             ahrefValue=/
203             next_sibling: '\n'
204             =====
205             type(eachLi):
206             ahrefValue=/
207             next_sibling: '\n'
208             =====
209             type(eachLi):
210             ahrefValue=/
211             next_sibling: '\n'
212             =====
213             type(eachLi):
214             ahrefValue=/
215             next_sibling: '\n'
216             =====
217             type(eachLi):
218             ahrefValue=/
219             next_sibling: '\n'
220             =====
221             type(eachLi):
222             ahrefValue=/
223             next_sibling: '\n'
224             =====
225             type(eachLi):
226             ahrefValue=/
227             next_sibling: '\n'
228             =====
229             type(eachLi):
230             ahrefValue=/
231             next_sibling: '\n'
232             =====
233             type(eachLi):
234             ahrefValue=/
235             next_sibling: '\n'
236             =====
237             type(eachLi):
238             ahrefValue=/
239             next_sibling: '\n'
240             =====
241             type(eachLi):
242             ahrefValue=/
243             next_sibling: '\n'
244             =====
245             type(eachLi):
246             ahrefValue=/
247             next_sibling: '\n'
248             =====
249             type(eachLi):
250             ahrefValue=/
251             next_sibling: '\n'
252             =====
253             type(eachLi):
254             ahrefValue=/
255             next_sibling: '\n'
256             =====
257             type(eachLi):
258             ahrefValue=/
259             next_sibling: '\n'
260             =====
261             type(eachLi):
262             ahrefValue=/
263             next_sibling: '\n'
264             =====
265             type(eachLi):
266             ahrefValue=/
267             next_sibling: '\n'
268             =====
269             type(eachLi):
270             ahrefValue=/
271             next_sibling: '\n'
272             =====
273             type(eachLi):
274             ahrefValue=/
275             next_sibling: '\n'
276             =====
277             type(eachLi):
278             ahrefValue=/
279             next_sibling: '\n'
280             =====
281             type(eachLi):
282             ahrefValue=/
283             next_sibling: '\n'
284             =====
285             type(eachLi):
286             ahrefValue=/
287             next_sibling: '\n'
288             =====
289             type(eachLi):
290             ahrefValue=/
291             next_sibling: '\n'
292             =====
293             type(eachLi):
294             ahrefValue=/
295             next_sibling: '\n'
296             =====
297             type(eachLi):
298             ahrefValue=/
299             next_sibling: '\n'
299             =====
300             type(eachLi):
301             ahrefValue=/
302             next_sibling: '\n'
303             =====
304             type(eachLi):
305             ahrefValue=/
306             next_sibling: '\n'
307             =====
308             type(eachLi):
309             ahrefValue=/
310             next_sibling: '\n'
311             =====
312             type(eachLi):
313             ahrefValue=/
314             next_sibling: '\n'
315             =====
316             type(eachLi):
317             ahrefValue=/
318             next_sibling: '\n'
319             =====
320             type(eachLi):
321             ahrefValue=/
322             next_sibling: '\n'
323             =====
324             type(eachLi):
325             ahrefValue=/
326             next_sibling: '\n'
327             =====
328             type(eachLi):
329             ahrefValue=/
330             next_sibling: '\n'
331             =====
332             type(eachLi):
333             ahrefValue=/
334             next_sibling: '\n'
335             =====
336             type(eachLi):
337             ahrefValue=/
338             next_sibling: '\n'
339             =====
340             type(eachLi):
341             ahrefValue=/
342             next_sibling: '\n'
343             =====
344             type(eachLi):
345             ahrefValue=/
346             next_sibling: '\n'
347             =====
348             type(eachLi):
349             ahrefValue=/
350             next_sibling: '\n'
351             =====
352             type(eachLi):
353             ahrefValue=/
354             next_sibling: '\n'
355             =====
356             type(eachLi):
357             ahrefValue=/
358             next_sibling: '\n'
359             =====
360             type(eachLi):
361             ahrefValue=/
362             next_sibling: '\n'
363             =====
364             type(eachLi):
365             ahrefValue=/
366             next_sibling: '\n'
367             =====
368             type(eachLi):
369             ahrefValue=/
370             next_sibling: '\n'
371             =====
372             type(eachLi):
373             ahrefValue=/
374             next_sibling: '\n'
375             =====
376             type(eachLi):
377             ahrefValue=/
378             next_sibling: '\n'
379             =====
380             type(eachLi):
381             ahrefValue=/
382             next_sibling: '\n'
383             =====
384             type(eachLi):
385             ahrefValue=/
386             next_sibling: '\n'
387             =====
388             type(eachLi):
389             ahrefValue=/
390             next_sibling: '\n'
391             =====
392             type(eachLi):
393             ahrefValue=/
394             next_sibling: '\n'
395             =====
396             type(eachLi):
397             ahrefValue=/
398             next_sibling: '\n'
399             =====
399             type(eachLi):
400             ahrefValue=/
401             next_sibling: '\n'
402             =====
403             type(eachLi):
404             ahrefValue=/
405             next_sibling: '\n'
406             =====
407             type(eachLi):
408             ahrefValue=/
409             next_sibling: '\n'
410             =====
411             type(eachLi):
412             ahrefValue=/
413             next_sibling: '\n'
414             =====
415             type(eachLi):
416             ahrefValue=/
417             next_sibling: '\n'
418             =====
419             type(eachLi):
420             ahrefValue=/
421             next_sibling: '\n'
422             =====
423             type(eachLi):
424             ahrefValue=/
425             next_sibling: '\n'
426             =====
427             type(eachLi):
428             ahrefValue=/
429             next_sibling: '\n'
430             =====
431             type(eachLi):
432             ahrefValue=/
433             next_sibling: '\n'
434             =====
435             type(eachLi):
436             ahrefValue=/
437             next_sibling: '\n'
438             =====
439             type(eachLi):
440             ahrefValue=/
441             next_sibling: '\n'
442             =====
443             type(eachLi):
444             ahrefValue=/
445             next_sibling: '\n'
446             =====
447             type(eachLi):
448             ahrefValue=/
449             next_sibling: '\n'
450             =====
451             type(eachLi):
452             ahrefValue=/
453             next_sibling: '\n'
454             =====
455             type(eachLi):
456             ahrefValue=/
457             next_sibling: '\n'
458             =====
459             type(eachLi):
460             ahrefValue=/
461             next_sibling: '\n'
462             =====
463             type(eachLi):
464             ahrefValue=/
465             next_sibling: '\n'
466             =====
467             type(eachLi):
468             ahrefValue=/
469             next_sibling: '\n'
470             =====
471             type(eachLi):
472             ahrefValue=/
473             next_sibling: '\n'
474             =====
475             type(eachLi):
476             ahrefValue=/
477             next_sibling: '\n'
478             =====
479             type(eachLi):
480             ahrefValue=/
481             next_sibling: '\n'
482             =====
483             type(eachLi):
484             ahrefValue=/
485             next_sibling: '\n'
486             =====
487             type(eachLi):
488             ahrefValue=/
489             next_sibling: '\n'
490             =====
491             type(eachLi):
492             ahrefValue=/
493             next_sibling: '\n'
494             =====
495             type(eachLi):
496             ahrefValue=/
497             next_sibling: '\n'
498             =====
499             type(eachLi):
500             ahrefValue=/
501             next_sibling: '\n'
502             =====
503             type(eachLi):
504             ahrefValue=/
505             next_sibling: '\n'
506             =====
507             type(eachLi):
508             ahrefValue=/
509             next_sibling: '\n'
510             =====
511             type(eachLi):
512             ahrefValue=/
513             next_sibling: '\n'
514             =====
515             type(eachLi):
516             ahrefValue=/
517             next_sibling: '\n'
518             =====
519             type(eachLi):
520             ahrefValue=/
521             next_sibling: '\n'
522             =====
523             type(eachLi):
524             ahrefValue=/
525             next_sibling: '\n'
526             =====
527             type(eachLi):
528             ahrefValue=/
529             next_sibling: '\n'
530             =====
531             type(eachLi):
532             ahrefValue=/
533             next_sibling: '\n'
534             =====
535             type(eachLi):
536             ahrefValue=/
537             next_sibling: '\n'
538             =====
539             type(eachLi):
540             ahrefValue=/
541             next_sibling: '\n'
542             =====
543             type(eachLi):
544             ahrefValue=/
545             next_sibling: '\n'
546             =====
547             type(eachLi):
548             ahrefValue=/
549             next_sibling: '\n'
550             =====
551             type(eachLi):
552             ahrefValue=/
553             next_sibling: '\n'
554             =====
555             type(eachLi):
556             ahrefValue=/
557             next_sibling: '\n'
558             =====
559             type(eachLi):
560             ahrefValue=/
561             next_sibling: '\n'
562             =====
563             type(eachLi):
564             ahrefValue=/
565             next_sibling: '\n'
566             =====
567             type(eachLi):
568             ahrefValue=/
569             next_sibling: '\n'
570             =====
571             type(eachLi):
572             ahrefValue=/
573             next_sibling: '\n'
574             =====
575             type(eachLi):
576             ahrefValue=/
577             next_sibling: '\n'
578             =====
579             type(eachLi):
580             ahrefValue=/
581             next_sibling: '\n'
582             =====
583             type(eachLi):
584             ahrefValue=/
585             next_sibling: '\n'
586             =====
587             type(eachLi):
588             ahrefValue=/
589             next_sibling: '\n'
590             =====
591             type(eachLi):
592             ahrefValue=/
593             next_sibling: '\n'
594             =====
595             type(eachLi):
596             ahrefValue=/
597             next_sibling: '\n'
598             =====
599             type(eachLi):
600             ahrefValue=/
601             next_sibling: '\n'
602             =====
603             type(eachLi):
604             ahrefValue=/
605             next_sibling: '\n'
606             =====
607             type(eachLi):
608             ahrefValue=/
609             next_sibling: '\n'
610             =====
611             type(eachLi):
612             ahrefValue=/
613             next_sibling: '\n'
614             =====
615             type(eachLi):
616             ahrefValue=/
617             next_sibling: '\n'
618             =====
619             type(eachLi):
620             ahrefValue=/
621             next_sibling: '\n'
622             =====
623             type(eachLi):
624             ahrefValue=/
625             next_sibling: '\n'
626             =====
627             type(eachLi):
628             ahrefValue=/
629             next_sibling: '\n'
630             =====
631             type(eachLi):
632             ahrefValue=/
633             next_sibling: '\n'
634             =====
635             type(eachLi):
636             ahrefValue=/
637             next_sibling: '\n'
638             =====
639             type(eachLi):
640             ahrefValue=/
641             next_sibling: '\n'
642             =====
643             type(eachLi):
644             ahrefValue=/
645             next_sibling: '\n'
646             =====
647             type(eachLi):
648             ahrefValue=/
649             next_sibling: '\n'
650             =====
651             type(eachLi):
652             ahrefValue=/
653             next_sibling: '\n'
654             =====
655             type(eachLi):
656             ahrefValue=/
657             next_sibling: '\n'
658             =====
659             type(eachLi):
660             ahrefValue=/
661             next_sibling: '\n'
662             =====
663             type(eachLi):
664             ahrefValue=/
665             next_sibling: '\n'
666             =====
667             type(eachLi):
668             ahrefValue=/
669             next_sibling: '\n'
670             =====
671             type(eachLi):
672             ahrefValue=/
673             next_sibling: '\n'
674             =====
675             type(eachLi):
676             ahrefValue=/
677             next_sibling: '\n'
678             =====
679             type(eachLi):
680             ahrefValue=/
681             next_sibling: '\n'
682             =====
683             type(eachLi):
684             ahrefValue=/
685             next_sibling: '\n'
686             =====
687             type(eachLi):
688             ahrefValue=/
689             next_sibling: '\n'
690             =====
691             type(eachLi):
692             ahrefValue=/
693             next_sibling: '\n'
694             =====
695             type(eachLi):
696             ahrefValue=/
697             next_sibling: '\n'
698             =====
699             type(eachLi):
700             ahrefValue=/
701             next_sibling: '\n'
702             =====
703             type(eachLi):
704             ahrefValue=/
705             next_sibling: '\n'
706             =====
707             type(eachLi):
708             ahrefValue=/
709             next_sibling: '\n'
710             =====
711             type(eachLi):
712             ahrefValue=/
713             next_sibling: '\n'
714             =====
715             type(eachLi):
716             ahrefValue=/
717             next_sibling: '\n'
718             =====
719             type(eachLi):
720             ahrefValue=/
721             next_sibling: '\n'
722             =====
723             type(eachLi):
724             ahrefValue=/
725             next_sibling: '\n'
726             =====
727             type(eachLi):
728             ahrefValue=/
729             next_sibling: '\n'
730             =====
731             type(eachLi):
732             ahrefValue=/
733             next_sibling: '\n'
734             =====
735             type(eachLi):
736             ahrefValue=/
737             next_sibling: '\n'
738             =====
739             type(eachLi):
740             ahrefValue=/
741             next_sibling: '\n'
742             =====
743             type(eachLi):
744             ahrefValue=/
745             next_sibling: '\n'
746             =====
747             type(eachLi):
748             ahrefValue=/
749             next_sibling: '\n'
750             =====
751             type(eachLi):
752             ahrefValue=/
753             next_sibling: '\n'
754             =====
755             type(eachLi):
756             ahrefValue=/
757             next_sibling: '\n'
758             =====
759             type(eachLi):
760             ahrefValue=/
761             next_sibling: '\n'
762             =====
763             type(eachLi):
764             ahrefValue=/
765             next_sibling: '\n'
766             =====
767             type(eachLi):
768             ahrefValue=/
769             next_sibling: '\n'
770             =====
771             type(eachLi):
772             ahrefValue=/
773             next_sibling: '\n'
774             =====
775             type(eachLi):
776             ahrefValue=/
777             next_sibling: '\n'
778             =====
779             type(eachLi):
780             ahrefValue=/
781             next_sibling: '\n'
782             =====
783             type(eachLi):
784             ahrefValue=/
785             next_sibling: '\n'
786             =====
787             type(eachLi):
788             ahrefValue=/
789             next_sibling: '\n'
790             =====
791             type(eachLi):
792             ahrefValue=/
793             next_sibling: '\n'
794             =====
795             type(eachLi):
796             ahrefValue=/
797             next_sibling: '\n'
798             =====
799             type(eachLi):
800             ahrefValue=/
801             next_sibling: '\n'
802             =====
803             type(eachLi):
804             ahrefValue=/
805             next_sibling: '\n'
806             =====
807             type(eachLi):
808             ahrefValue=/
809             next_sibling: '\n'
810             =====
811             type(eachLi):
812             ahrefValue=/
813             next_sibling: '\n'
814             =====
815             type(eachLi):
816             ahrefValue=/
817             next_sibling: '\n'
818             =====
819             type(eachLi):
820             ahrefValue=/
821             next_sibling: '\n'
822             =====
823             type(eachLi):
824             ahrefValue=/
825             next_sibling: '\n'
826             =====
827             type(eachLi):
828             ahrefValue=/
829             next_sibling: '\n'
830             =====
831             type(eachLi):
832             ahrefValue=/
833             next_sibling: '\n'
834             =====
835             type(eachLi):
836             ahrefValue=/
837             next_sibling: '\n'
838             =====
839             type(eachLi):
840             ahrefValue=/
841             next_sibling: '\n'
842             =====
843             type(eachLi):
844             ahrefValue=/
845             next_sibling: '\n'
846             =====
847             type(eachLi):
848             ahrefValue=/
849             next_sibling: '\n'
850             =====
851             type(eachLi):
852             ahrefValue=/
853             next_sibling: '\n'
854             =====
855             type(eachLi):
856             ahrefValue=/
857             next_sibling: '\n'
858             =====
859             type(eachLi):
860             ahrefValue=/
861             next_sibling: '\n'
862             =====
863             type(eachLi):
864             ahrefValue=/
865             next_sibling: '\n'
866             =====
867             type(eachLi):
868             ahrefValue=/
869             next_sibling: '\n'
870             =====
871             type(eachLi):
872             ahrefValue=/
873             next_sibling: '\n'
874             =====
875             type(eachLi):
876             ahrefValue=/
877             next_sibling: '\n'
878             =====
879             type(eachLi):
880             ahrefValue=/
881             next_sibling: '\n'
882             =====
883             type(eachLi):
884             ahrefValue=/
885             next_sibling: '\n'
886             =====
887             type(eachLi):
888             ahrefValue=/
889             next_sibling: '\n'
890             =====
891             type(eachLi):
892             ahrefValue=/
893             next_sibling: '\n'
894             =====
895             type(eachLi):
896             ahrefValue=/
897             next_sibling: '\n'
898             =====
899             type(eachLi):
900             ahrefValue=/
901             next_sibling: '\n'
902             =====
903             type(eachLi):
904             ahrefValue=/
905             next_sibling: '\n'
906             =====
907             type(eachLi):
908             ahrefValue=/
909             next_sibling: '\n'
910             =====
911             type(eachLi):
912             ahrefValue=/
913             next_sibling: '\n'
914             =====
915             type(eachLi):
916             ahrefValue=/
917             next_sibling: '\n'
918             =====
919             type(eachLi):
920             ahrefValue=/
921             next_sibling: '\n'
922             =====
923             type(eachLi):
924             ahrefValue=/
925             next_sibling: '\n'
926             =====
927             type(eachLi):
928             ahrefValue=/
929             next_sibling: '\n'
930             =====
931             type(eachLi):
932             ahrefValue=/
933             next_sibling: '\n'
934             =====
935             type(eachLi):
936             ahrefValue=/
937             next_sibling: '\n'
938             =====
939             type(eachLi):
940             ahrefValue=/
941             next_sibling: '\n'
942             =====
943             type(eachLi):
944             ahrefValue=/
945             next_sibling: '\n'
946             =====
947             type(eachLi):
948             ahrefValue=/
949             next_sibling: '\n'
950             =====
951             type(eachLi):
952             ahrefValue=/
953             next_sibling: '\n'
954             =====
955             type(eachLi):
956             ahrefValue=/
957             next_sibling: '\n'
958             =====
959             type(eachLi):
960             ahrefValue=/
961             next_sibling: '\n'
962             =====
963             type(eachLi):
964             ahrefValue=/
965             next_sibling: '\n'
966             =====
967             type(eachLi):
968             ahrefValue=/
969             next_sibling: '\n'
970             =====
971             type(eachLi):
972             ahrefValue=/
973             next_sibling: '\n'
974             =====
975             type(eachLi):
976             ahrefValue=/
977             next_sibling: '\n'
978             =====
979             type(eachLi):
980             ahrefValue=/
981             next_sibling: '\n'
982             =====
983             type(eachLi):
984             ahrefValue=/
985             next_sibling: '\n'
986             =====
987             type(eachLi):
988             ahrefValue=/
989             next_sibling: '\n'
990             =====
991             type(eachLi):
992             ahrefValue=/
993             next_sibling: '\n'
994             =====
995             type(eachLi):
996             ahrefValue=/
997             next_sibling: '\n'
998             =====
999             type(eachLi):
1000             ahrefValue=/
1001             next_sibling: '\n'
1002             =====
1003             type(eachLi):
1004             ahrefValue=/
1005             next_sibling: '\n'
1006             =====
1007             type(eachLi):
1008             ahrefValue=/
1009             next_sibling: '\n'
1010             =====
1011             type(eachLi):
1012             ahrefValue=/
1013             next_sibling: '\n'
1014             =====
1015             type(eachLi):
1016             ahrefValue=/
1017             next_sibling: '\n'
1018             =====
1019             type(eachLi):
1020             ahrefValue=/
1021             next_sibling: '\n'
1022             =====
1023             type(eachLi):
1024             ahrefValue=/
1025             next_sibling: '\n'
1026             =====
1027             type(eachLi):
1028             ahrefValue=/
1029             next_sibling: '\n'
1030             =====
1031             type(eachLi):
1032             ahrefValue=/
1033             next_sibling: '\n'
1034             =====
1035             type(eachLi):
1036             ahrefValue=/
1037             next_sibling: '\n'
1038             =====
1039             type(eachLi):
1040             ahrefValue=/
1041             next_sibling: '\n'
1042             =====
1043             type(eachLi):
1044             ahrefValue=/
1045             next_sibling: '\n'
1046             =====
1047             type(eachLi):
1048             ahrefValue=/
1049             next_sibling: '\n'
1050             =====
1051             type(eachLi):
1052             ahrefValue=/
1053             next_sibling: '\n'
1054             =====
1055             type(eachLi):
1056             ahrefValue=/
1057             next_sibling: '\n'
1058             =====
1059             type(eachLi):
1060             ahrefValue=/
1061             next_sibling: '\n'
1062             =====
1063             type(eachLi):
1064             ahrefValue=/
1065             next_sibling: '\n'
1066             =====
1067             type(eachLi):
1068             ahrefValue=/
1069             next_sibling: '\n'
1070             =====
1071             type(eachLi):
1072             ahrefValue=/
1073             next_sibling: '\n'
1074             =====
1075             type(eachLi):
1076             ahrefValue=/
1077             next_sibling: '\n'
1078             =====
1079             type(eachLi):
1080             ahrefValue=/
1081             next_sibling: '\n'
1082             =====
1083             type(eachLi):
1084             ahrefValue=/
1085             next_sibling: '\n'
1086             =====
1087             type(eachLi):
1088             ahrefValue=/
1089             next_sibling: '\n'
1090             =====
1091             type(eachLi):
1092             ahrefValue=/
1093             next_sibling: '\n'
1094             =====
1095             type(eachLi):
1096             ahrefValue=/
1097             next_sibling: '\n'
1098             =====
1099             type(eachLi):
1100             ahrefValue=/
1101             next_sibling: '\n'
1102             =====
1103             type(eachLi):
1104             ahrefValue=/
1105             next_sibling: '\n'
1106             =====
1107             type(eachLi):
1108             ahrefValue=/
1109             next_sibling: '\n'
1110             =====
1111             type(eachLi):
1112             ahrefValue=/
1113             next_sibling: '\n'
1114             =====
1115             type(eachLi):
1116             ahrefValue=/
1117             next_sibling: '\n'
1118             =====
1119             type(eachLi):
1120             ahrefValue=/
1121             next_sibling: '\n'
1122             =====
1123             type(eachLi):
1124             ahrefValue=/
1125             next_sibling: '\n'
1126             =====
1127             type(eachLi):
1128             ahrefValue=/
1129             next_sibling: '\n'
1130             =====
1131             type(eachLi):
1132             ahrefValue=/
1133             next_sibling: '\n'
1134             =====
1135             type(eachLi):
1136             ahrefValue=/
1137             next_sibling: '\n'
1138             =====
1139             type(eachLi):
1140             ahrefValue=/
1141             next_sibling: '\n'
1142             =====
1143             type(eachLi):
1144             ahrefValue=/
1145             next_sibling: '\n'
1146             =====
1147             type(eachLi):
1148             ahrefValue=/
1149             next_sibling: '\n'
1150             =====
1151             type(eachLi):
1152             ahrefValue=/
1153             next_sibling: '\n'
1154             =====
1155             type(eachLi):
1156             ahrefValue=/
1157             next_sibling: '\n'
1158             =====
1159             type(eachLi):
1160             ahrefValue=/
1161             next_sibling: '\n'
1162             =====
1163             type(eachLi):
1164             ahrefValue=/
1165             next_sibling: '\n'
1166             =====
1167             type(eachLi):
1168             ahrefValue=/
1169             next_sibling: '\n'
1170             =====
1171             type(eachLi):
1172             ahrefValue=/
1173             next_sibling: '\n'
1174             =====
1175             type(eachLi):
1176             ahrefValue=/
1177             next_sibling: '\n'
1178             =====
1179             type(eachLi):
1180             ahrefValue=/
1181             next_sibling: '\n'
1182             =====
1183             type(eachLi):
1184             ahrefValue=/
1185             next_sibling: '\n'
1186             =====
1187             type(eachLi):
1188             ahrefValue=/
1189             next_sibling: '\n'
1190             =====
1191             type(eachLi):
1192             ahrefValue=/
1193             next_sibling: '\n'
1194             =====
1195             type(eachLi):
1196             ahrefValue=/
1197             next_sibling: '\n'
1198             =====
1199             type(eachLi):
1200             ahrefValue=/
1201             next_sibling: '\n'
1202             =====
1203             type(eachLi):
1204             ahrefValue=/
1205             next_sibling: '\n'
1206             =====
1207             type(eachLi):
1208             ahrefValue=/
1209             next_sibling: '\n'
1210             =====
1211             type(eachLi):
1212             ahrefValue=/
1213             next_sibling: '\n'
1214             =====
1215             type(eachLi):
1216             ahrefValue=/
1217             next_sibling: '\n'
1218             =====
1219             type(eachLi):
1220             ahrefValue=/
1221             next_sibling: '\n'
1222             =====
1223             type(eachLi):
1224             ahrefValue=/
1225             next_sibling: '\n'
1226             =====
1227             type(eachLi):
1228             ahrefValue=/
1229             next_sibling: '\n'
1230             =====
1231             type(eachLi):
1232             ahrefValue=/
1233             next_sibling: '\n'
1234             =====
1235             type(eachLi):
1236             ahrefValue=/
1237             next_sibling: '\n'
1238             =====
1239             type(eachLi):
1240             ahrefValue=/
1241             next_sibling: '\n'
1242             =====
1243             type(eachLi):
1244             ahrefValue=/
1245             next_sibling: '\n'
1246             =====
1247             type(eachLi):
1248             ahrefValue=/
1249             next_sibling: '\n'
1250             =====
1251             type(eachLi):
1252             ahrefValue=/
1253             next_sibling: '\n'
1254             =====
1255             type(eachLi):
1256             ahrefValue=/
1257             next_sibling: '\n'
1258             =====
1259             type(eachLi):
1260             ahrefValue=/
1261             next_sibling: '\n'
1262             =====
1263             type(eachLi):
1264             ahrefValue=/
1265             next_sibling: '\n'
1266             =====
1267             type(eachLi):
1268             ahrefValue=/
1269             next_sibling: '\n'
1270             =====
1271             type(eachLi):
1272             ahrefValue=/
1273             next_sibling: '\n'
1274             =====
1275             type(eachLi):
1276             ahrefValue=/
1277             next_sibling: '\n'
1278             =====
1279             type(eachLi):
1280             ahrefValue=/
1281             next_sibling: '\n'
1282             =====
1283             type(eachLi):
1284             ahrefValue=/
1285             next_sibling: '\n'
1286             =====
1287             type(eachLi):
1288             ahrefValue=/
1289             next_sibling: '\n'
1290             =====
1291             type(eachLi):
1292             ahrefValue=/
1293             next_sibling: '\n'
1294             =====
1295             type(eachLi):
1296             ahrefValue=/
1297             next_sibling: '\n'
1298             =====
1299             type(eachLi):
1300             ahrefValue=/
1301             next_sibling: '\n'
1302             =====
1303             type(eachLi):
1304             ahrefValue=/
1305             next_sibling: '\n'
1306             =====
1307             type(eachLi):
1308             ahrefValue=/
1309             next_sibling: '\n'
1310             =====
1311             type(eachLi):
1312             ahrefValue=/
1313             next_sibling: '\n'
1314             =====
1315             type(eachLi):
1316             ahrefValue=/
1317             next_sibling: '\n'
1318             =====
1319             type(eachLi):
1320             ahrefValue=/
1321             next_sibling: '\n'
1322             =====
1323             type(eachLi):
1324             ahrefValue=/
1325             next_sibling: '\n'
1326             =====
1327             type(eachLi):
1328             ahrefValue=/
1329             next_sibling: '\n'
1330             =====
1331             type(eachLi):
1332             ahrefValue=/
1333             next_sibling: '\n'
1334             =====
1335             type(eachLi):
1336             ahrefValue=/
1337             next_sibling: '\n'
1338             =====
1339             type(eachLi):
1340             ahrefValue=/
1341             next_sibling: '\n'
1342             =====
1343             type(eachLi):
1344             ahrefValue=/
1345             next_sibling: '\n'
1346             =====
1347             type(eachLi):
1348             ahrefValue=/
1349             next_sibling: '\n'
1350             =====
1351             type(eachLi):
1352             ahrefValue=/
1353             next_sibling: '\n'
1354             =====
1355             type(eachLi):
1356             ahrefValue
```


注意事项和心得

注意事项

BeautifulSoup的Tag的属性

BeautifulSoup处理这种html源码：

```
<a href="http://creativecommons.org/licenses/by/3.0/deed.zh" target="_blank">版权声明</a>
```

后，是可以通过 `soup.a['href']` 去获得对应的href属性值的。

但是，想要去获得当前的某个未知的BeautifulSoup.Tag中，一共存在多少个属性，以及每个属性的值的时候，却不知道如何下手了。

比如对于如下html源码：

```
<p class="cc-lisence" style="line-height:180%;">.....</p>
```

想要得知，一共有两个属性，分别是class和style，然后就可以通过上面的方法，去获得对应的属性值了。

对此问题，虽然看到了官网的解释：[Tags的属性](#)中的 你可以将Tag看成字典来访问标签的属性

但是还是无法通过：

```
if("class" in soup)
```

的方式去判断soup中是否存在class属性，因为此时soup是Beautifulsoup.Tag类型变量，而不是dict变量。

并且，如果去强制将soup转换成为dict变量：

```
soupDict = dict(soup)
```

会报错的。

最后，还是无意间发现，原来 BeautifulSoup.Tag 是有个 `attrs` 属性的，其可以获得对应的元组列表，每一个元组是对应属性名和属性值：

```
attrsList = soup.attrs
print("attrsList=%s" % attrsList())

attrsList= [('class', 'cc-lisence'), ('style', 'line-height:180%;')]
```

这样自己就可以从元组列表中去转换，获得属性的列表或字典变量了，就可以接着按照自己意愿去处理了。

另外，此处也通过 `soup.name` 获得了该tag的名字

而想要获得整个soup变量所有的属性和方法的话，可以用经典的dir去打印出来：

```
print("dir(soup)=%s" % dir(soup))
```

此处的打印输出为：

```
dir(soup)= ['BARE_AMPERSAND_OR_BRACKET', 'XML_ENTITIES_TO_SPECIAL_CHARS', 'XML_SPECIAL_CHARS_TO_ENTITIES', '__call__', '__contains__', '__delitem__', '__doc__', '__eq__', '__getatt r__', '__getitem__', '__init__', '__iter__', '__len__', '__module__', '__ne__', '__nonzero__', '__repr__', '__setitem__', '__str__', '__unicode__', '_convertEntities', '_findAll', '_findOne', '_getAttrMap', '_invert', '_lastRecursiveChild', '_sub_entity', 'append', 'attrMap', 'attrs', 'childGenerator', 'containsSubstitutions', 'contents', 'convertHTMLEntitie s', 'convertXMLEntities', 'decompose', 'escapeUnrecognizedEntities', 'extract', 'fetch', 'fetchNextSiblings', 'fetchParents', 'fetchPrevious', 'fetchPreviousSiblings', 'fetchText', 'find', 'findAll', 'findAllNext', 'findAllPrevious', 'findChild', 'findChildren', 'findNex t', 'findNextSibling', 'findNextSiblings', 'findParent', 'findParents', 'findPrevious', 'findPreviousSibling', 'findPreviousSiblings', 'first', 'firstText', 'get', 'has_key', 'hidden', 'insert', 'isSelfClosing', 'name', 'next', 'nextGenerator', 'nextSibling', 'nextSiblingGenerator', 'parent', 'parentGenerator', 'parserClass', 'prettify', 'previous', 'previousGenerator', 'previousSibling', 'previousSiblingGenerator', 'recursiveChildGenerator', 'renderContents', 'replaceWith', 'setup', 'substituteEncoding', 'toEncoding']
```

有需要的话，可以对这些属性和方法，都尝试一下，以便更加清楚其含义。

刚写完上面这句话呢，然后自己随便测试了一下 `attrMap`：

```
attrMap = soup.attrMap
print("attrMap=%s" % attrMap)
```

然后就很惊喜的发现，原来此处的attrMap，就是我程序中所需要的属性的dict变量啊：

```
attrMap= {'style': 'line-height:180%', 'class': 'cc-lisence'}
```

这样，就又省去了我程序中将attrs转换为dict变量的操作了，更加提高了效率。

在此感谢Beautifulsoup的开发者。

BeautifulSoup有时候会遇到非法的，不支持的html源码而导致无法解析或无法正常解析html

在使用Beautifulsoup过程中，对于大多数html源码，通过指定正确的编码，或者本身是默认UTF-8编码而无需指定编码类型，其都可以正确解析html源码，得到对应的soup变量。

然后就接着去利用soup实现你所想要的功能了。

但是有时候会发现，有些html解析后，有些标签等内容丢失了，即所得到的soup不是所期望的完整的html的内容。

这时候，很可能遇到了非法的html，即其中可能包含了一些不合法的html标签等内容，导致Beautifulsoup虽然可以解析，没有报错，但是实际上得到的soup变量，内容缺失了一部分了。

比如我就遇到过不少这样的例子：

部分Blogbus的帖子的html中非html5和html5的代码混合导致Beautifulsoup解析错误

之前在[为BlogsToWordPress添加Blogbus支持](#)过程中去解析Blogbus的帖子的时候，遇到一个特殊的帖子：

<http://ronghuihou.blogbus.com/logs/89099700.html>

其中一堆的非html5的代码中，包含了这样一段html5的代码的写法，即标签属性值不加括号的：

```
<SCRIPT language=JavaScript>
document.oncontextmenu=new Function("event.returnValue=false;"); //禁止右键功能,单击右键将无
任何反应
document.onselectstart=new Function( "event.returnValue=false;"); //禁止先择,也就是无法复制
</SCRIPT language=JavaScript>
```

结果导致Beautifulsoup解析错误，得到的soup中，找不到所需要的各种class等属性值。

对应的解决办法就是，把这部分的代码删除掉，然后再解析就可以了：

其中一堆的非html5的代码中，包含了这样一段html5的代码的写法，即标签属性值不加括号的：

```
foundInvliadScript = re.search("<SCRIPT language=JavaScript>.+</SCRIPT language=JavaScript
>", html, re.I | re.S )
logging.debug("foundInvliadScript=%s", foundInvliadScript)
if(foundInvliadScript):
    invalidScriptStr = foundInvliadScript.group(0)
    logging.debug("invalidScriptStr=%s", invalidScriptStr)
    html = html.replace(invalidScriptStr, "")
    logging.debug("filter out invalid script OK")

soup = htmlToSoup(html)
```

判断浏览器版本的相关代码，导致Beautifulsoup解析不正常

之前在给[BlogsToWordpress](#)添加新浪博客的支持的过程中

遇到很多新浪博客的帖子的html中，包含很多判断浏览器版本的相关代码：

```
<!--[if lte IE 6]>
```

```
xxx  
xxx  
<![endif]->
```

由此导致Beautifulsoup解析html不正常。

font标签嵌套层次太多，导致Beautifulsoup无法解析html

接上面那个解析新浪博客帖子的例子，期间又遇到另外一个问题，对于一些特殊帖子：

http://blog.sina.com.cn/s/blog_5058502a01017j3j.html

其包含特殊的好几十个font标签且是一个个嵌套的代码，导致无法Beautifulsoup无法解析html，后来把对应嵌套的font标签删除掉，才可以正常解析。

相关python代码为：

```
# handle special case for http://blog.sina.com.cn/s/blog_5058502a01017j3j.html
processedHtml = processedHtml.replace('<font COLOR="#6D4F19"><font COLOR="#7AAF5A"><font COLOR="#7AAF5A"><font COLOR="#6D4F19"><font COLOR="#7AAF5A"><font COLOR="#7AAF5A"><font COLOR="#7AAF5A">', '')
processedHtml = processedHtml.replace("</FONT></FONT></FONT></FONT></FONT></FONT>", '')
```

遇到其他类似的问题，也可以去删除或替换出错代码，即可解决问题。

不过需要说明的是，很多时候，你未必很容易就找到出错的代码。

想要找到出错的代码，更多的时候，需要你一点点调试，一点点的删除看似可疑的一些html源码，然后最终才能定位到出错的代码，然后删除掉后，才可以正常工作的。

使用心得

crifan的Beautifulsoup函数

之前已把一些常用功能封装成函数，放到自己的库中了，详见：

[\[crifanLibPython/crifanBeautifulsoup.py at master · crifan/crifanLibPython\]](#)

find 和 find_all 中支持正则写法

如之前例子中所提到的：

```
ahrefNonEmptyP = re.compile("\S+") # ahref="/xsglxt/f/jyxt/anony/showZwxx?zpxxid=104719161
&type="
styleColorP = re.compile("color:#[a-zA-Z0-9]+;") # style="color:#ff0000;""
foundAList = soup.find_all('a', attrs {"fbfw": "外", "ahref": ahrefNonEmptyP, "style": style
ColorP})
```

soup.find 和 soup.find_all 中，都支持正则的写法：

```
# also match a-incontent a-title cs-contentblock-hoverlink
titleP = re.compile("a-incontent a-title(\s+\w+)?")
foundIncontentTitle = soup.find(attrs={"class":titleP})
```

就可以同时匹配多种情况了：

- a-incontent a-title
- a-incontent a-title cs-contentblock-hoverlink
- a-incontent a-title xxx

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-16 19:49:53

附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2020-02-16 17:49:08

参考资料

- 实现博客搬家
- Python专题教程：BeautifulSoup详解
- re — Regular expression operations — Python 3.8.2rc1 documentation
- re --- 正则表达式操作 — Python 3.8.2rc1 文档
- 【教程】Python中第三方的用于解析HTML的库：BeautifulSoup – 在路上
- 【已解决】Python3中，已经安装了bs4（Beautifulsoup 4）了，但是却还是出错：ImportError: No module named BeautifulSoup
- 【教程】抓取网并提取网页中所需要的信息 之 Python版
- 【总结】Python的第三方库BeautifulSoup的使用心得
- 【整理】关于Python中的html处理库函数BeautifulSoup使用注意事项
- 【已解决】BeautifulSoup已经获得了Unicode的Soup但是print出来却是乱码
- 【教程】BeautifulSoup中使用正则表达式去搜索多种可能的关键字
- 【整理】用BeautifulSoup查找属性值未知的标签
- [crifanLib/crifanLib.py at master · crifan/crifanLib](#)
-

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2020-02-16 19:47:06