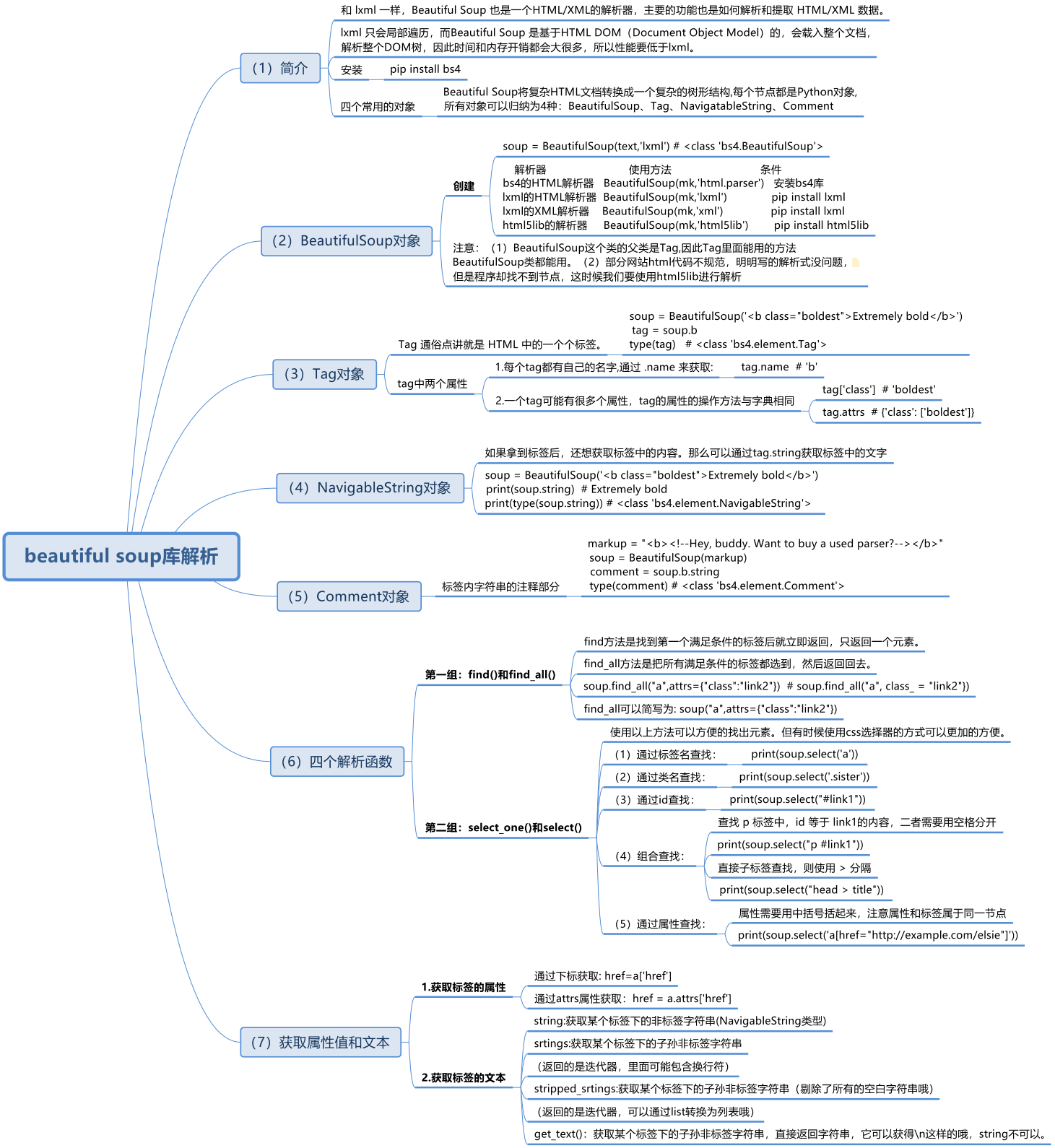
beautiful soup库解析
（1）简介
和 lxml 一样，Beautiful Soup 也是一个HTML/XML的解析器，主要的功能也是如何解析和提取 HTML/XML 数据。
lxml 只会局部遍历，而Beautiful Soup 是基于HTML DOM（Document Object Model）的，会载入整个文档，解析整个DOM树，因此时间和内存开销都会大很多，所以性能要低于lxml。
安装 pip install bs4
四个常用的对象
Beautiful Soup将复杂HTML文档转换成一个复杂的树形结构,每个节点都是Python对象，所有对象可以归纳为4种：BeautifulSoup、Tag、NavigatableString、Comment
（2）BeautifulSoup对象
soup = BeautifulSoup(text,'lxml') # <class 'bs4.BeautifulSoup'>
创建
解析器 使用方法 条件
bs4的HTML解析器 BeautifulSoup(mk,'html.parser') 安装bs4库
lxml的HTML解析器 BeautifulSoup(mk,'lxml') pip install lxml
lxml的XML解析器 BeautifulSoup(mk,'xml') pip install lxml
html5lib的解析器 BeautifulSoup(mk,'html5lib') pip install html5lib
注意：（1）BeautifulSoup这个类的父类是Tag,因此Tag里面能用的方法BeautifulSoup类都能用。（2）部分网站html代码不规范，明明写的解析式没问题，但是程序却找不到节点，这时候我们要使用html5lib进行解析
（3）Tag对象
soup = BeautifulSoup('Extremely bold')
tag = soup.b
type(tag) # <class 'bs4.element.Tag'>
Tag 通俗点讲就是 HTML 中的一个个标签。
tag中两个属性
1.每个tag都有自己的名字,通过 .name 来获取: tag.name # 'b'
2.一个tag可能有很多个属性，tag的属性的操作方法与字典相同
tag['class'] # 'boldest'
tag.attrs # {'class': ['boldest']}
（4）NavigableString对象
如果拿到标签后，还想获取标签中的内容。那么可以通过tag.string获取标签中的文字
soup = BeautifulSoup('Extremely bold')
print(soup.string) # Extremely bold
print(type(soup.string)) # <class 'bs4.element.NavigableString'>
（5）Comment对象
markup = "<!--Hey, buddy. Want to buy a used parser?-->"
soup = BeautifulSoup(markup)
comment = soup.b.string
type(comment) # <class 'bs4.element.Comment'>
标签内字符串的注释部分
（6）四个解析函数
第一组：find()和find_all()
find方法是找到第一个满足条件的标签后就立即返回，只返回一个元素。
find_all方法是把所有满足条件的标签都选到，然后返回回去。
soup.find_all("a",attrs={"class":"link2"}) # soup.find_all("a", class_ = "link2"})
find_all可以简写为: soup("a",attrs={"class":"link2"})
第二组：select_one()和select()
使用以上方法可以方便的找出元素。但有时候使用css选择器的方式可以更加的方便。
（1）通过标签名查找： print(soup.select('a'))
（2）通过类名查找： print(soup.select('.sister'))
（3）通过id查找： print(soup.select("#link1"))
（4）组合查找：
查找 p 标签中，id 等于 link1的内容，二者需要用空格分开
print(soup.select("p #link1"))
直接子标签查找，则使用 > 分隔
print(soup.select("head > title"))
（5）通过属性查找：
属性需要用中括号括起来，注意属性和标签属于同一节点
print(soup.select('a[href="http://example.com/elsie"]'))
（7）获取属性值和文本
1.获取标签的属性
通过下标获取: href=a['href']
通过attrs属性获取： href = a.attrs['href']
2.获取标签的文本
string:获取某个标签下的非标签字符串(NavigableString类型)
srtings:获取某个标签下的子孙非标签字符串
（返回的是迭代器，里面可能包含换行符）
stripped_srtings:获取某个标签下的子孙非标签字符串（剔除了所有的空白字符串哦）
（返回的是迭代器，可以通过list转换为列表哦）
get_text()：获取某个标签下的子孙非标签字符串，直接返回字符串，它可以获得\n这样的哦，string不可以。

优化不规则 html 代码的方法：

```
from bs4 import BeautifulSoup
def prettify_html(html_text):
    soup = BeautifulSoup(html_text,'html5lib')
    return soup.prettify()
```

prettify()为 HTML 文本<>及其内容增加更加'\n'

一个简单的实例的部分代码：

```
def bs4Paeser(urltext,goodlist):
    soup = BeautifulSoup(urltext,'lxml')
    #默认的解析器为 html.parser.
    #print(type(soup))
# 第一步：四种方法得到保存我们需要的数据的那一部分
    # 注意，find 返回的是<class 'bs4.element.Tag'>
    goodsTag = soup.find('div',class_="sousuoListBox clearfix")
    # 注意，find_all 返回的是<class 'bs4.element.ResultSet'>
    # 其可以看成一个列表，里面的每一个元素都是一个'bs4.element.Tag'类型
    goodsTag = soup.find_all('div',attrs={"class":"sousuoListBox clearfix"})[0]
    # 注意 select 利用 css 选择器语法，返回的是<class 'list'>列表
    # 里面的每一个元素都是一个'bs4.element.Tag'类型
    #但是，select_one 只返回第一个符合条件的，所以是'bs4.element.Tag'类型
    goodsTag = soup.select_one('div.sousuoListBox.clearfix')
    #这个 div 有两个类名,css 选择器连着写就行
    goodsTag = soup.select('body > div:nth-child(11) > div')
    #使用浏览器自带的 selector
# 第二步：找到每一种商品数据所在的那一部分，其可以视为返回一个列表
     goodslist = soup.find_all('div',class_ = "ssCardItem")
     #<class 'bs4.element.ResultSet'>
     goodslist = soup('div',class_ = "ssCardItem")
     #因为 find_all 这个方法特别常用，我们可以简写哦
    goodslist = soup.select('div.ssCardItem')  #<class 'list'>
    for everygood in goodslist:
        name = everygood.find('a',attrs={'class':"siteCardICH3"}).string
        #<class 'bs4.element.NavigableString'>
        href = everygood.find('a',attrs ={'class':"siteCardICH3"})['href']
        introduction = everygood.select_one('p.siteCardIC_p.souSuo').get_text()
        #<class 'str'>
```