1、使用Beautiful Soup 解析网页

通过 requests 库已经可以抓到网页源码，接下来要从源码中找到并提取数据。Beautiful Soup 是 python
的一个库，其最主要的功能是从网页中抓取数据。Beautiful Soup 目前已经被移植到 bs4 库中，也就是说在导 入 Beautiful Soup 时需要先安装 bs4 库。

HTML文件其实就是由一组尖括号构成的标签组织起来的，每一对尖括号形式一个标签，标签之间存在上下关系，形成标签树；因此可以说Beautiful Soup库是解析、遍历、维护“标签树”的功能库。

p标签：

：标签Tag ——一般，标签名都是成对出现的（位于起始和末尾），例如P；在第一个标签名之后可以有0到多个属性，表示标签的特点

...

——中间的class属性，其值为“title ”（属性是由键和值，键值对构成的）

```
import requests
import re
from bs4 import    BeautifulSoup
url='http://www.cntour.cn/'
strhtml=requests.get(url)
soup=BeautifulSoup(strhtml.text,'html.parser', from_encoding='utf-8')
data =
soup.select('#main>div>div.mtop.firstMod.clearfix>div.centerBox>ul.newsList>li>a
')
#将数据
print(data)
#数据清洗和组织数据
for item in data:
    #明确提取的数据为标题和链接
    result={
        #提取标签正文
        'title': item.get_text(),
        #提取标签中的href属性
        'link': item.get('href'),
        #提取链接中的id
        'id':re.findall('\d+', item.get('href'))
    }
print(result)
```

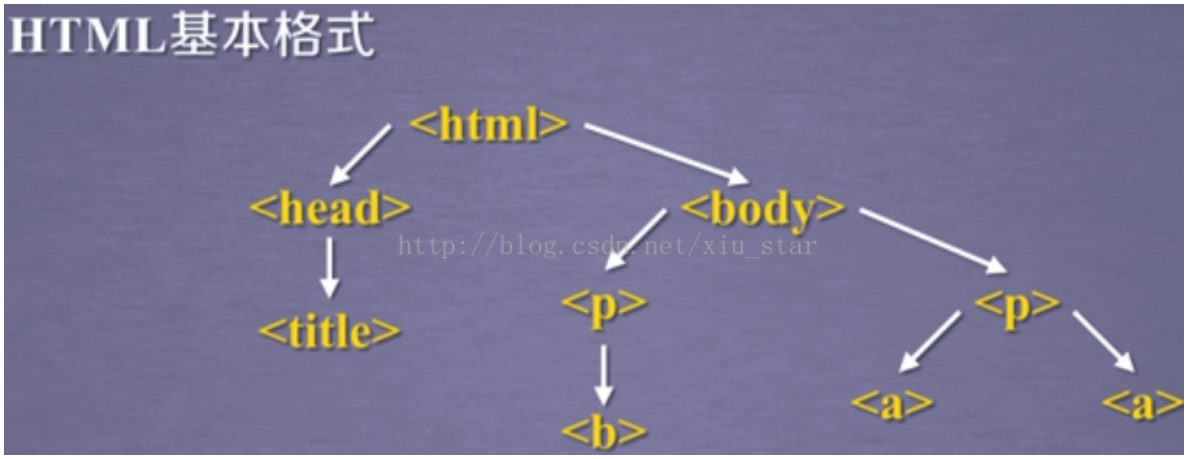
Beautiful Soup库解析器：

bs4的HTML解析器：BeautifulSoup(mk,'html.parser')——条件：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



2、爬虫攻防：

爬虫是模拟人的浏览访问行为，进行数据的批量抓取。当抓取的数据量逐渐增大时，会给被访问的服务器造成很大的压力，甚至有可能崩溃。换句话就是说，服务器是不喜欢有人抓取自己的数据的。那么，网站方面就会针对这些爬虫者，采取一些反爬策略。

服务器第一种识别爬虫的方式就是通过检查连接的 useragent 来识别到底是浏览器访问，还是代码访问的。如果是代码访问的话，访问量增大时，服务器会直接封掉来访 IP。

2.1、应对初级反爬机制

在进行访问时，我们在开发者环境下不仅可以找到 URL、Form Data，还可以在 Request headers 中构造浏览器的请求头，封装自己。服务器识别浏览器访问的方法就是判断 keyword 是否为 Request headers 下的 User-Agent，如图 22 所示。

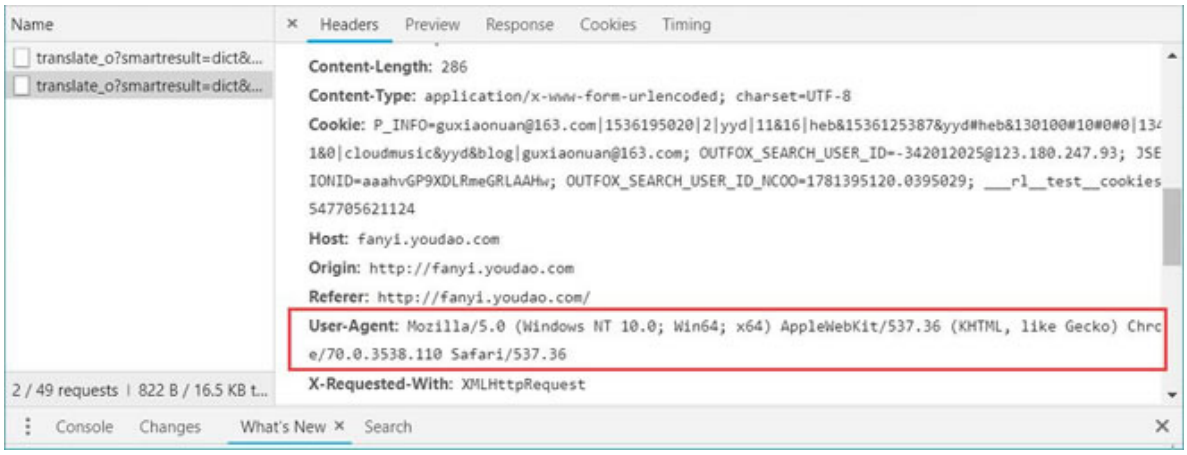


图 22

因此，我们只需要构造这个请求头的参数。创建请求头部信息即可，代码如下：

```
headers={'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36'}
response = request.get(url,headers=headers)
```

如果在一个 IP 下批量访问下载图片，这个行为不符合正常人类的行为，肯定要被封 IP。
其原理也很简单，就是统计每个IP的访问频率，该频率超过阈值，就会返回一个验证码，如果真的是用户访问的话，用户就会填写，然后继续访问，如果是代码访问的话，就会被封 IP。

这个问题的解决方案有两个，第一个就是常用的增设延时，每 3 秒钟抓取一次，代码如下：

```
import time

time.sleep(3)
```

服务器的目的就是查出哪些为代码访问，然后封锁IP 。为避免被封锁ip,数据采集的时候经常会使用代理，

request也有相应的proxies属性。

```
    proxies={
        "http":"http://10.10.1.10:3128",
        "https":"http://10.10.1.10:1080",
    }
    response = requests.get(url, proxies=proxies)
```

3、四大对象：

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:

- `Tag`
- `NavigableString`
- `BeautifulSoup`
- `Comment`

1）CSS选择器：

我们在写 CSS 时，标签名不加任何修饰，类名前加点，id名前加 #，在这里我们也可以利用类似的方法来筛选元素，用到的方法是 **soup.select()**，返回类型是 **list**

1.1、通过标签名查找

```
import requests
import re
from bs4 import    BeautifulSoup
url='http://www.cntour.cn/'
strhtml=requests.get(url)
#soup = BeautifulSoup(open(url))
soup=BeautifulSoup(strhtml.text,'html.parser', from_encoding='utf-8')


print(soup.select('a'))
```

1.2、通过类名查找

```
print soup.select('.sister')
```

1.3、通过id名查找

```
soup.select('#link1')
```

1.4、组合查找

```
print soup.select("head > title")
```

1.5、属性查找

查找时还可以加入属性元素，属性需要用中括号括起来，注意属性和标签属于同一节点，所以中间不能加空格，否则会无法匹配到。

```
print soup.select('a[class="sister"]')
```

select方法返回的结果都是列表形式，可以便利输出，然后用get_text()方法获取它的 内容。

4、pymsql连接mysql数据库

pymysql是python提供的一个mysql客户端模块,用于与mysql服务器建立连接,发送查询,并获取结果等;

4.1、查询

```
import pymysql

def conn_mysql():
    #1、建立连接
    conn = pymysql.connect(host='localhost', port=3306, user='root',
password='zhujn', db='test1', charset='utf8')
    #2.创建游标
    cur = conn.cursor()
    #3.执行sql语句
    #cur.execute('select * from test')
    id = 3
    sql = "select * from test where id = '%s'"%id
    #result = cur.fetchall()#查询所有符合条件数据
    r = cur.execute(sql)#返回查询成功的记录数
    cur.close()
    conn.close()
    print(r)

if __name__ == '__main__':
    conn_mysql()
```

4.2.1、fetchall

```
# 获取所有的数据
rows = cursor.fetchall()
```

4.2.1、fetchmany

```
#查询多个
row=cursor.fetchmany(3)
print(row)
```

4.2.1、fetchone

```
# 查询第一行的数据
row = cursor.fetchone()
```

5、增、删、改、查

commit()方法：在数据库里增、删、改的时候，必须要进行提交，否则插入的数据不生效。

5.1、增加一组数据

```
sql='insert into test(id,name) values(%s,%s)'

effect_row=cursor.execute(sql,(username,pwd))   # effect_row=1
```

5.2、增加多组数据

```
    sql = "insert into test(id,name) values(%s,%s)"
    effect_row = cur.executemany(sql,[('1','a'),('2','b'),('4','c')])
```

5.3、删除

```
sql='delete from userinfo  where pwd like "%2"'
effect_row=cursor.execute(sql)
```

5.4、修改

```
sql='update userinfo set username = %s  where pwd="114"'
effect_row=cursor.execute(sql,'niu')
```

6、数据可视化

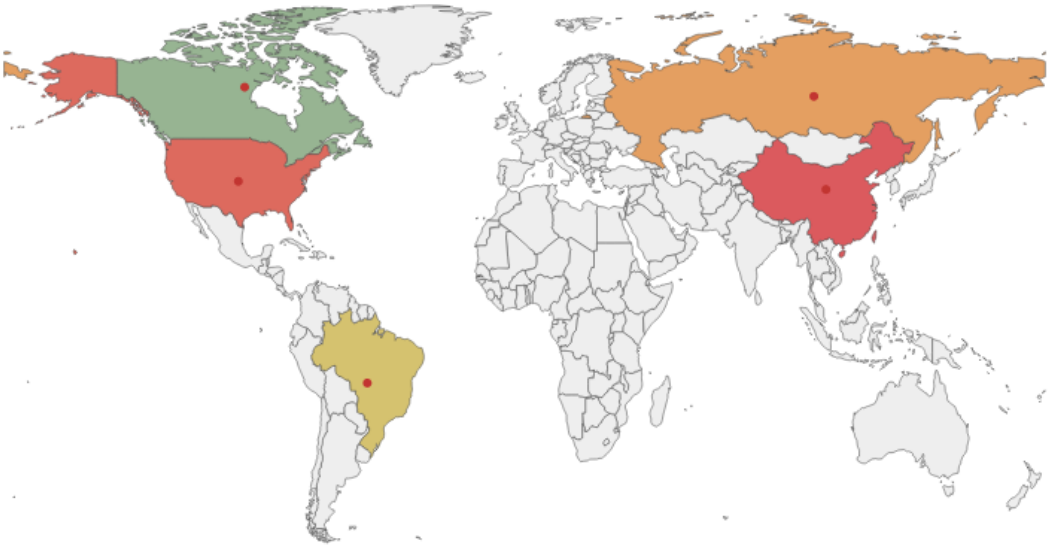
ECharts是一个纯Javascript的图表库，可以流畅的运行在PC和移动设备上，兼容当前绝大部分浏览器，底层依赖轻量级的Canvas类库ZRender，提供直观、生动、可交互、可高度个性化定制的数据可视化图表。ECharts提供了常规的折线图、柱状图、散点图、饼图、K线图，用于统计的盒形图，用于地理数据可视化的地图、热力图、线图，用于关系数据可视化的关系图、treemap，多维数据可视化的平行坐标，还有用于BI的漏斗图、仪表盘，并且支持图与图之间的混搭。

6.1、中国地图

```
from pyecharts import Map
province_distribution = {'河南': 45.23, '北京': 37.56, '河北': 21, '辽宁': 12, '江西': 6, '上海': 20, '安徽': 10, '江苏': 16, '湖南': 9,'浙江': 13, '海南': 2, '广东': 22, '湖北': 8, '黑龙江': 11, '澳门': 1, '陕西': 11, '四川': 7, '内蒙古': 3, '重庆': 3,'云南': 6, '贵州': 2, '吉林': 3, '山西': 12, '山东': 11, '福建': 4, '青海': 1, '天津': 1,'其他': 1}
provice = list(province_distribution.keys())
values = list(province_distribution.values())
map = Map("中国地图","中国地图",width=1200,height=600)
map.add("",provice,values,visual_range=
[0,50],maptype='china',is_visualmap=True,visual_text_color='#000')
map.render(path='中国地图.html')
```

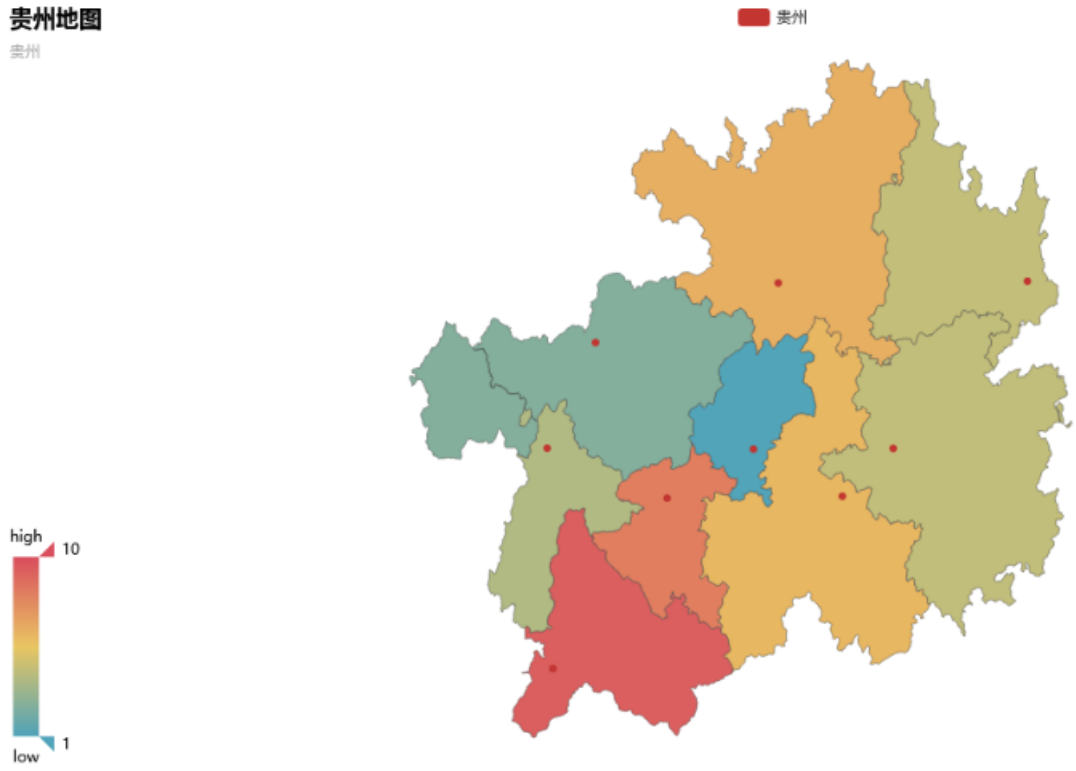
6.1、世界地图

```
from pyecharts import Map
value = [95.1,23.2,43.3,66.4,88.5]
attr = ["China","Canada","Brazil","Russia","United States"]
map = Map("世界地图",width=1200,height=600)
map.add("世界地
图",attr,value,maptype="world",is_visualmap=True,visual_text_color='#000')
map.render(path="世界地图.html")
```



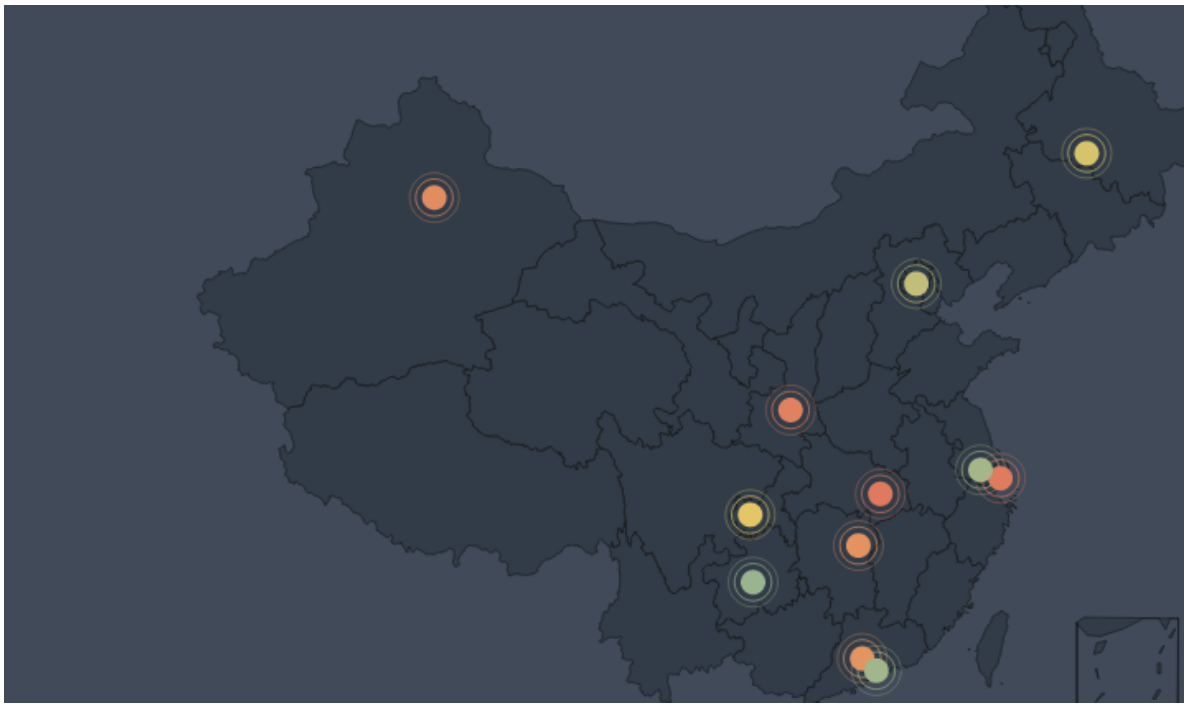
6.2、省市地图

贵州地图



```
from pyecharts import Map

map2 = Map("贵州地图", '贵州', width=1200, height=600)

city = ['贵阳市', '六盘水市', '遵义市', '安顺市', '毕节市', '铜仁市', '黔西南布依族苗族自治州', '黔东南苗族侗族自治州', '黔南布依族苗族自治州']

values2 = [1.07, 3.85, 6.38, 8.21, 2.53, 4.37, 9.38, 4.29, 6.1]

map2.add('贵州', city, values2, visual_range=[1, 10], maptype='贵州', is_visualmap=True, visual_text_color='#000')

map2.render(path="贵州地图.html")
```

6.3、热力图



```
from pyecharts import Geo

keys = ['上海', '北京', '合肥', '哈尔滨', '广州', '成都', '无锡', '杭州', '武汉', '深圳', '西安', '郑州', '重庆', '长沙', '贵阳', '乌鲁木齐']

values = [4.07, 1.85, 4.38, 2.21, 3.53, 4.37, 1.38, 4.29, 4.1, 1.31, 3.92, 4.47, 2.40, 3.60, 1.2, 3.7]

geo = Geo("全国主要城市空气质量热力图", "data from pm2.5", title_color="#fff",title_pos="left",width=1200,height=600,background_color='#404a59')

geo.add("空气质量热力图", keys, values, visual_range=[0, 5], type='effectScatter',visual_text_color="#fff", symbol_size=15,is_visualmap=True, is_roam=True)

# type有scatter, effectScatter, heatmap三种模式可选，可根据自己的需求选择对应的图表模式

geo.render(path="全国主要城市空气质量热力图.html")
```