一、Urllib库 【古老】

2020年4月6日 21:36

爬虫准备工作：

1、你想爬取什么数据?

2、找到数据对应的网页

3、分析网页结构找到数据所在的标签位置

找到你想爬取的页面，按F12,按小箭头找到你要爬取的内容

模拟HTTP请求，向服务器发出这个请求，获取到服务器返回给我们HTML

用正则表达式提取我们要的数据（名字，人气）

1.1 URL 格式

2020年4月9日　12:06

protocol://hostname[:port]/path/[;parameters][?query]#fragment

URL由三部分组成：

- **第一部分协议：http、https、ftp、file（访问本地文件夹）、ed2k（电驴）等**
- **第二部分域名[:端口号]：网址和端口号**
- **第三部分：资源的具体地址，如目录和文件名**

第一部分和第二部分使用://隔开

第二部分和第三部分使用/隔开

1.2 urllib库

2020年4月10日 10:47

我们首先了解一下 Urllib 库，它是 Python 内置的 HTTP 请求库，也就是说我们不需要额外安装即可使用，它包含四个模块：

第一个模块 request，它是最基本的 HTTP 请求模块，我们可以用它来模拟发送一请求，就像在浏览器里输入网址然后敲击回车一样，只需要给库方法传入 URL 还有额外的参数，就可以模拟实现这个过程了。

第二个 error 模块即异常处理模块，如果出现请求错误，我们可以捕获这些异常，然后进行重试或其他操作保证程序不会意外终止。

第三个 parse 模块是一个工具模块，提供了许多 URL 处理方法，比如拆分、解析、合并等等的方法。

第四个模块是 robotparser，主要是用来识别网站的 robots.txt 文件，然后判断哪些网站可以爬，哪些网站不可以爬的，其实用的比较少。

在这里重点对前三个模块进行下讲解。

1.2.1 发送请求 urlopen

2020年4月10日 10:48

使用 Urllib 的 request 模块我们可以方便地实现 Request（请求） 的发送并得到 Response（答复）。

1、urlopen( )
urllib.request 模块提供了最基本的构造 HTTP 请求的方法，利用它可以模拟浏览器的一个请求发起过程，同时它还带有处理authenticaton（授权验证），redirections（重定向），cookies（浏览器Cookies）以及其它内容。
我们来感受一下它的强大之处，以 Python 官网为例，我们来把这个网页抓下来：

```
import urllib.request
response = urllib.request.urlopen('https://www.python.org')
print(response.read().decode('utf-8'))
```

read()全部读取，括号内可以指定字节数
readline()读取一行
readlines()读取多行

运行一下看看结果是什么?

接下来我们看下它返回的到底是什么，利用 type() 方法输出 Response 的类型。返回：<class 'http.client.HTTPResponse'>

通过输出结果可以发现它是一个 HTTPResposne 类型的对象，它主要包含的方法有 read()、readinto()、getheader(name)、getheaders()、fileno() 等方法和 msg、version、status、reason、debuglevel、closed 等属性。得到这个对象之后，我们把它赋值为 response 变量，然后就可以调用这些方法和属性，得到返回结果的一系列信息了。

例如调用 read() 方法可以得到返回的网页内容，调用 status 属性就可以得到返回结果的状态码，如 200 代表请求成功，404 代表网页未找到等。

下面再来一个实例感受一下：

```
import urllib.request
response = urllib.request.urlopen('https://www.python.org')
print(response.status)
print(response.getheaders())
print(response.getheader('Server'))
```

可见，三个输出分别输出了响应的状态码，响应的头信息，以及通过调用 getheader() 方法并传递一个参数 Server 获取了 headers 中的 Server 值，结果是 nginx，意思就是服务器是 nginx 搭建的。
利用以上最基本的 urlopen() 方法，我们可以完成最基本的简单网页的 GET 请求抓取。
如果我们想给链接传递一些参数该怎么实现呢？我们首先看一下 urlopen() 函数的API：

urllib.request.urlopen(url, data=None, [timeout,]*, cafile=None, capath=None, cadefault=False, context=None)

可以发现除了第一个参数可以传递 URL 之外，我们还可以传递其它的内容，比如 data（附加数据）、timeout（超时时间）等等。
下面我们详细说明下这几个参数的用法:

- **data参数**

data 参数是可选的，如果要添加 data，它要是字节流编码格式的内容，即 bytes 类型，通过 bytes() 方法可以进行转化，另外如果传递了这个 data 参数，它的请求方式就不再是 GET 方式请求，而是 POST。
下面用一个实例来感受一下：

```
import urllib.parse
import urllib.request
data = bytes(urllib.parse.urlencode({'word': 'hello'}), encoding='utf8')
response = urllib.request.urlopen('http://httpbin.org/post', data=data)
print(response.read())
```

- timeout参数

timeout 参数可以设置超时时间，单位为秒，意思就是如果请求超出了设置的这个时间还没有得到响应，就会抛出异常，如果不指定，就会使用全局默认时间。它支持 HTTP、HTTPS、FTP 请求。
因此我们可以通过设置这个超时时间来控制一个网页如果长时间未响应就跳过它的抓取，利用 try except 语句就可以实现这样的操作，代码如下：

```
import socket
```

```
import urllib.request
import urllib.error
try:
    response = urllib.request.urlopen('http://httpbin.org/get', timeout=0.1)
except urllib.error.URLError as e:
    if isinstance(e.reason, socket.timeout):
        print('TIME OUT')
```

- 其他参数

还有 context 参数，它必须是 ssl.SSLContext 类型，用来指定 SSL 设置。
cafile 和 capath 两个参数是指定 CA 证书和它的路径，这个在请求 HTTPS 链接时会有用。
cadefault 参数现在已经弃用了，默认为 False。
以上讲解了 urlopen() 方法的用法，通过这个最基本的函数可以完成简单的请求和网页抓取，如需更加详细了解，可以参见官方文档：
https://docs.python.org/3/library/urllib.request.html。

1.2.2 Request

2020年4月10日 11:12

由上我们知道利用 urlopen() 方法可以实现最基本请求的发起，但这几个简单的参数并不足以构建一个完整的请求，如果请求中需要加入 Headers 等信息，我们就可以利用更强大的 Request 类来构建一个请求。
首先我们用一个实例来感受一下 Request 的用法：

```
mport urllib.request

request = urllib.request.Request('https://python.org')
response = urllib.request.urlopen(request)
print(response.read().decode('utf-8'))
```

可以发现，我们依然是用 urlopen() 方法来发送这个请求，只不过这次 urlopen() 方法的参数不再是一个 URL，而是一个 Request 类型的对象，通过构造这个这个数据结构，一方面我们可以将请求独立成一个对象，另一方面可配置参数更加丰富和灵活。
下面我们看一下 Request 都可以通过怎样的参数来构造，它的构造方法如下：

```
class urllib.request.Request(url, data=None, headers={}, origin_req_host=None, unverifiable=False, method=None)
```

第一个 url 参数是请求 URL，这个是必传参数，其他的都是可选参数。
第二个 data 参数如果要传必须传 bytes（字节流）类型的，如果是一个字典，可以先用 urllib.parse 模块里的 urlencode() 编码。
第三个 headers 参数是一个字典，这个就是 Request Headers 了，你可以在构造 Request 时通过 headers 参数直接构造，也可以通过调用 Request 实例的 add_header() 方法来添加，Request Headers 最常用的用法就是通过修改 User-Agent 来伪装浏览器，默认的 User-Agent 是 Python-urllib，我们可以通过修改它来伪装浏览器。
第四个 origin_req_host 参数指的是请求方的 host 名称或者 IP 地址。
第五个 unverifiable 参数指的是这个请求是否是无法验证的，默认是False。意思就是说用户没有足够权限来选择接收这个请求的结果。例如我们请求一个 HTML 文档中的图片，但是我们没有自动抓取图像的权限，这时 unverifiable 的值就是 True。
第六个 method 参数是一个字符串，它用来指示请求使用的方法，比如GET，POST，PUT等等。
写个例子：

```
from urllib import request, parse

url = 'http://httpbin.org/post'
headers = {
    'User-Agent': 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)',
    'Host': 'httpbin.org'
}
dict = {
    'name': 'Germey'
}
data = bytes(parse.urlencode(dict), encoding='utf8')
req = request.Request(url=url, data=data, headers=headers, method='POST')
response = request.urlopen(req)
print(response.read().decode('utf-8'))
```

在这里我们通过四个参数构造了一个 Request，url 即请求 URL，在headers 中指定了 User-Agent 和 Host，传递的参数 data 用了 urlencode() 和 bytes() 方法来转成字节流，另外指定了请求方式为 POST。
通过观察结果可以发现，我们成功设置了 data，headers 以及 method。
另外 headers 也可以用 add_header() 方法来添加。

```
req = request.Request(url=url, data=data, method='POST')
req.add_header('User-Agent', 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)')
```

如此一来，我们就可以更加方便地构造一个 Request，实现请求的发送啦。

1.2.3 异常处理

2020年4月10日 11:16

我们了解了 Request 的发送过程，但是在网络情况不好的情况下，出现了异常怎么办呢？这时如果我们不处理这些异常，程序很可能报错而终止运行，所以异常处理还是十分有必要的。
Urllib 的 error 模块定义了由 request 模块产生的异常。如果出现了问题，request 模块便会抛出 error 模块中定义的异常。
主要有这两个处理异常类，URLError,HTTPError
下面写个例子：

```
from urllib import request, error
try:
    response = request.urlopen('http://cuiqingcai.com/index.htm')
except error.URLError as e:
    print(e.reason)
```

我们打开一个不存在的页面，照理来说应该会报错，但是这时我们捕获了 URLError 这个异常，运行结果：Not Found

```
from urllib import request,error
try:
    response = request.urlopen('http://cuiqingcai.com/index.htm')
except error.HTTPError as e:
    print(e.reason, e.code, e.headers, seq='\n')
```

运行结果：

```
Not Found
404
Server: nginx/1.4.6 (Ubuntu)
Date: Wed, 03 Aug 2016 08:54:22 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Vary: Cookie
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
Link: <http://cuiqingcai.com/wp-json/>; rel="https://api.w.org/"
```

HTTPError,它有三个属性。
code，返回 HTTP Status Code，即状态码，比如 404 网页不存在，500 服务器内部错误等等。
reason，同父类一样，返回错误的原因。
headers，返回 Request Headers。
因为 URLError 是 HTTPError 的父类，所以我们可以先选择捕获子类的错误，再去捕获父类的错误，所以上述代码更好的写法如下：

```
from urllib import request, error

try:
    response = request.urlopen('http://cuiqingcai.com/index.htm')
except error.HTTPError as e:
    print(e.reason, e.code, e.headers, sep='\n')
except error.URLError as e:
    print(e.reason)
else:
    print('Request Successfully')
```

这样我们就可以做到先捕获 HTTPError，获取它的错误状态码、原因、Headers 等详细信息。如果非 HTTPError，再捕获 URLError 异常，输出错误原因。最后用 else 来处理正常的逻辑，这是一个较好的异常处理写法。

1.3 requests模块详解

2020年4月9日 13:02

1）导入模块

```
import requests
```

2）发送请求的简洁

示例代码：获取一个网页（个人github）

```
import requests
r = requests.get('https://github.com/Ranxf')      # 最基本的不带参数的get请求
r1 = requests.get(url='http://dict.baidu.com/s', params={'wd': 'python'})      # 带参数的get请求
requests.get(‘https://github.com/timeline.json’)      # GET请求
requests.post(“http://httpbin.org/post”)                 # POST请求
requests.put(“http://httpbin.org/put”)                    # PUT请求
requests.delete(“http://httpbin.org/delete”)            # DELETE请求
requests.head(“http://httpbin.org/get”)                  # HEAD请求
requests.options(“http://httpbin.org/get”)             # OPTIONS请求
```

3）为url传递参数

```
>>> url_params = {'key':'value'}      #    字典传递参数，如果值为None的键不会被添加到url中
>>> r = requests.get('your url',params = url_params)
>>> print(r.url)
    your url?key=value
```

4）响应的内容

```
r.encoding                #获取当前的编码
r.encoding = 'utf-8'      #设置编码
r.text                    #以encoding解析返回内容。字符串方式的响应体，会自动根据响应头部的字符编码进行解码。
r.content                 #以字节形式（二进制）返回。字节方式的响应体，会自动为你解码 gzip 和 deflate 压缩。

r.headers                 #以字典对象存储服务器响应头，但是这个字典比较特殊，字典键不区分大小写，若键不存在则返回None

r.status_code             #响应状态码
r.raw                     #返回原始响应体，也就是 urllib 的 response 对象，使用 r.raw.read()
r.ok                      # 查看r.ok的布尔值便可以知道是否登陆成功
 #*特殊方法*#
r.json()                  #Requests中内置的JSON解码器，以json形式返回,前提返回的内容确保是json格式的，不然解析出错会抛异常
r.raise_for_status()      #失败请求(非200响应)抛出异常
```

1.4 获取一张图片

2020年4月10日 11:37

预热一下：先下载一张图片

```
import urllib.request
网址 = 'http://pic1.win4000.com/wallpaper/e/58dc65d9e5edf.jpg'
接收 = urllib.request.urlopen(网址)
读取 = 接收.read()
# 使用with打开不需要手动关闭，Python基础第15课讲过
with open('赵丽颖.jpg','wb') as 变量名:
    变量名.write(读取)  # 文件写入，Python基础第10课讲过
```

1.5 天气预报

2020年4月10日 11:54

```
import urllib.request
import gzip
import json
import re
print('------天气查询------')
def 获取天气信息() :
    城市名称 = input('请输入要查询的城市名称：')
    url1 = 'http://wthrcdn.etouch.cn/weather_mini?city='+urllib.parse.quote(城市名称)
    读取 = urllib.request.urlopen(url1).read()  #读取网页数据
    读取 = gzip.decompress(读取).decode('utf-8')  #解压网页数据
    天气_字典 = json.loads(读取)  #将json数据转换为dict数据
    return 天气_字典


def 显示天气信息(天气_字典):
    if 天气_字典.get('desc') == 'invilad-citykey':  # Python基础第5课
        print('你输入的城市名有误，或者天气中心未收录你所在城市')
    elif 天气_字典.get('desc') =='OK':
        天气预报 = 天气_字典.get('data').get('forecast')
        print('城市：',天气_字典.get('data').get('city'))
        print('温度：',天气_字典.get('data').get('wendu')+'℃ ')
        print('感冒：',天气_字典.get('data').get('ganmao'))
        print('风向：',天气预报[0].get('fengxiang'))
        风力 = 天气预报[0].get('fengli')
        r = re.findall('<!\[CDATA\[(.*)\]\]>', 风力)  # 正则表达式2.9
        print('风级：', "".join(r))  # Python基础第2节
        print('高温：',天气预报[0].get('high'))
        print('低温：',天气预报[0].get('low'))
        print('天气：',天气预报[0].get('type'))
        print('日期：',天气预报[0].get('date'))
        print('********************************')
        未来四天 =input('是否要显示未来四天天气，是/否：')
        if 未来四天 == '是' or 'Y' or 'y':
            for i in range(1,5):
```

```
            print('日期：',天气预报[i].get('date'))
            print('风向：',天气预报[i].get('fengxiang'))
            风力 = 天气预报[i].get('fengli')
            r = re.findall('<!\[CDATA\[(.*)\]\]>',风力)  # 正则表达式2.9
            print('风级：',"".join(r))    # Python基础第2节
            print('高温：',天气预报[i].get('high'))
            print('低温：',天气预报[i].get('low'))
            print('天气：',天气预报[i].get('type'))
            print('------------------------')
    print('************************************')
```

显示天气信息(获取天气信息())

1.5.1 涉及新知识点

2020年4月10日 12:44

import gzip

读取 = gzip.decompress(读取).decode('utf-8')　#解压网页数据，设置编码格式

URL只允许一部分ASCII字符，其他字符（如汉字）是不符合标准的，此时就要进行编码。
因为我在构造URL的过程中要使用到中文：

url1 = 'http://wthrcdn.etouch.cn/weather_mini?city='+urllib.parse.quote(城市名称)

天气_字典 = json.loads(读取) #将json数据转换为dict数据

补充：JSON

三、Josn与Python类型之间的转换

json	python
object	dict
array	list
string	str
number	int
number	float
true	True
false	False
null	None

1.6 urlretrieve 从网站下载

2020年4月10日 20:40

import urllib.request

网址 = 'http://pic1.win4000.com/wallpaper/e/58dc65d9e5edf.jpg'

urllib.request.urlretrieve(网址,'网址.jpg')

1.7 urlencode编码和parse_qs解码

2020年4月10日 20:56

```
from urllib import request  # import urllib.request, Python基础第16课
url= 'https://www.baidu.com/s?cl=3&wd=摄影'
发送请求 = request.urlopen(url)
print(发送请求.read())  # ASCII码不能识别，因为有中文

# urllib.parse.urlencode 将字典中key:value转换为key=编码后的value
from urllib import request,parse
url= 'https://www.baidu.com/s?cl=3&'+parse.urlencode({"wd":"摄影"})
读取 = request.urlopen(url).read()
print(读取)

# 不建议使用parse_qs解码
from urllib import request,parse
url= 'https://www.baidu.com/s?cl=3&'+parse.urlencode({"wd":"摄影"})
读取 = request.urlopen(url).read()
print(url)
print(parse.parse_qs(url))
print(parse.unquote(url))
```

1.8 quote编码和unquote解码【推荐】

2020年4月10日 21:45

```
# urllib.parse.quote将str数据转换为对应编码
from urllib import request,parse
url1 = 'http://wthrcdn.etouch.cn/weather_mini?city='+parse.quote('摄影')
读取 = request.urlopen(url1).read()  #读取网页数据
print(读取)


# urllib.parse.unquote将编码后的数据转换为编码前数据
from urllib import request,parse
url= 'https://www.baidu.com/s?cl=3&'+parse.urlencode({"wd":"摄影"})
读取 = request.urlopen(url).read()
print(url)
print(parse.unquote(url))
```

1.9 urlparse和urlsplit网址分割

2020年4月10日 21:52

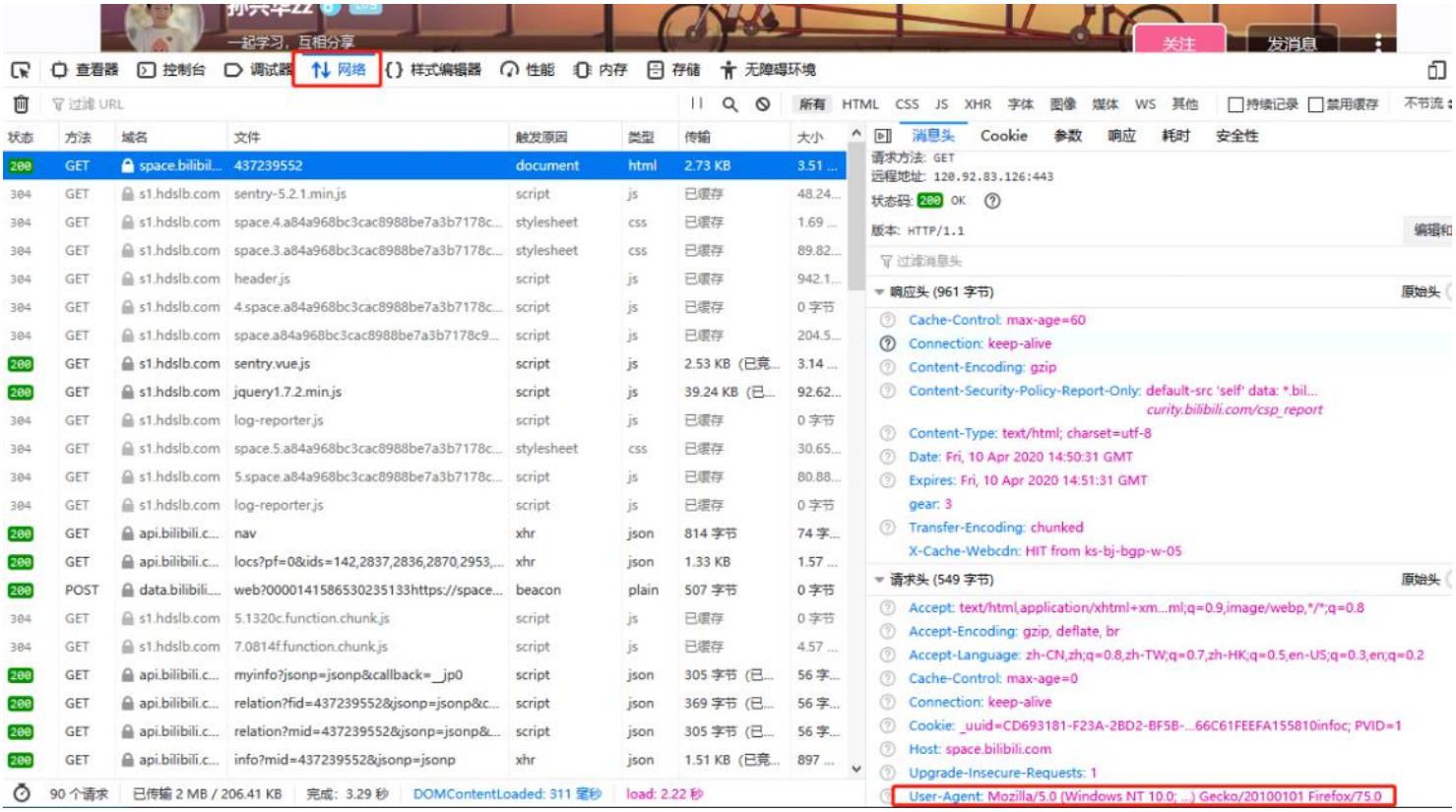
```
# urlparse拿到url中的某一项
from urllib import parse
url = 'https://www.bilibili.com/video/BV1kp4y1C7c8'
分割 = parse.urlparse(url)
print(分割)
print('协议：',分割.scheme)
print('域名：',分割.netloc)
print('目录：',分割.path)

params：参数 （基本用不到的东西，忽略它吧）
query：查询字符
fragment：锚点

# urlsplit拿到url中的某一项，与urlparse区别：没有params
from urllib import parse
url = 'https://www.bilibili.com/video/BV1kp4y1C7c8'
分割 = parse.urlsplit(url)
print(分割)
print('协议：',分割.scheme)
print('域名：',分割.netloc)
print('目录：',分割.path)

总结：urlparse和urlsplit功能一样，只是urlparse里多了一个params属性
```

1.10 urllib.request.Request请求头

2020年4月10日　22:35

如果想要在请求时候增加一些请求头，那么就必须使用urllib.request.Request来实现，因为很多网站都在反爬虫

```
# 网站发现正在访问服务器的是程序
from urllib import request
url = 'https://space.bilibili.com/437239552'
获取 = request.urlopen(url)
print(获取.read())
```




```
通过User-Agent传递
from urllib import request
url = 'https://space.bilibili.com/437239552'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0'}
获取 = request.Request(url,headers=headers)
读取 = request.urlopen(获取).read()
print(读取)
```

二、Requests模块 【简洁高效】

2020年4月10日 13:12

安装：

1、找到pip3.exe所在的文件夹，复制路径

我的路径是： C:\Users\孙艺航\AppData\Local\Programs\Python\Python37\Scripts

2、按Win+R,输入CMD确定,如图1

3、进入后，先输入cd 路径 回车,如图2

4、输入 pip3 install requests 回车,如图2　　(用同样的方法再安装 pip3 install bs4) 如图4

5、在Python编译器中，导入模块不报错就证明安装成功了。如图3

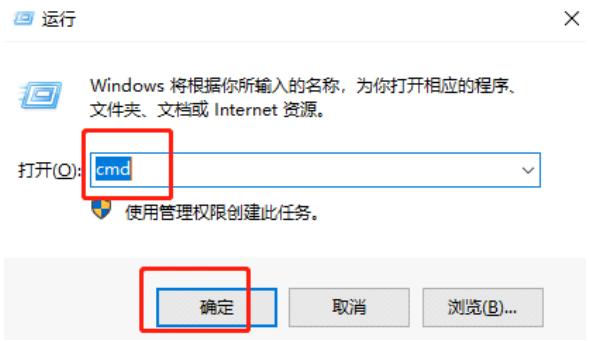


图1

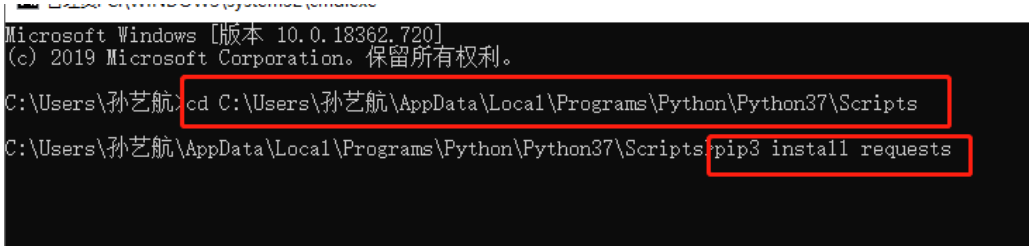


图2

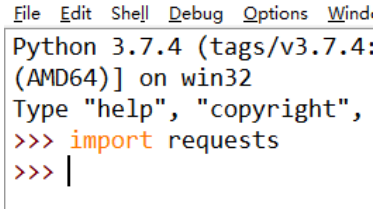


图3

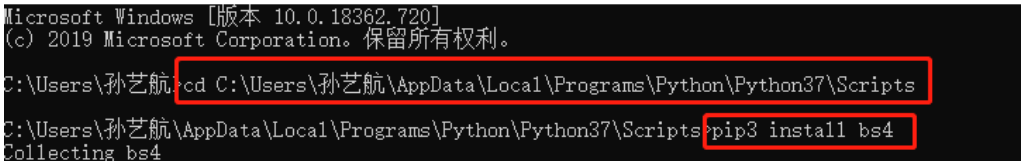


图4

2.1 Requests七个方法和响应对象属性

2020年4月11日 10:57

七个方法	说明
☐ requests.request()	构造一个请求，支撑一下各方法的基础方法
☑ requests.get()	获取HTML网页的主要方法，对应HTTP的GET
☐ requests.head()	获取HTML网页头的信息方法，对应HTTP的HEAD
☑ requests.post()	向HTML网页提交POST请求方法，对应HTTP的POST
☐ requests.put()	向HTML网页提交PUT请求的方法，对应HTTP的RUT
☐ requests.patch()	向HTML网页提交局部修改请求，对应于HTTP的PATCH
☐ requests.delete()	向HTML页面提交删除请求,对应HTTP的DELETE

响应对象（r）属性	说明
☐ r.status_code	HTTP请求的返回状态
☑ r.text	HTTP响应内容的字符串形式，即：url对应的页面内容
☐ r.encoding	从HTTP header中猜测的响应内容编码方式
☐ r.apparent_encoding	从内容中分析出的响应内容编码方式（备选编码方式）
☐ r.content	HTTP响应内容的二进制形式

2.1 Requests模块编码流程

2020年4月10日 23:34

学会Requests模块，你就掌握了爬虫领域的半壁江山

```
# 导入模块
import requests

# 指定Url
url = 'https://www.bilibili.com'

# 发起请求：使用get方法发起get请求，该方法会返回一个响应对象
响应对象 = requests.get(url)

# 获取响应数据：通过调用响应对象的text属性，返回响应对象中存储的字符串形式的响应数据（页面源码）
响应数据 = 响应对象.text

# 持久化存储
with open('./bilibili.html','w',encoding='utf-8') as 变量名:
    变量名.write(响应数据)
print('爬虫数据完毕！！！')
```

2.2 获取某个网页requests.get()

2020年4月11日 10:20

get()方法

r = requests.get(url,params = None,**kwargs)

url:网址

params：url中的额外参数，字典或字节流格式，可选

**kwargs：12个控制访问参数

```
import requests
url = 'https://www.baidu.com'
响应对象 = requests.get(url)
print(响应对象)                  # <Response [200]>
print(响应对象.status_code)      # 查看状态,200为正常，404为错误
print(type(响应对象) )           # 查看它的类  <class 'requests.models.Response'>  这是response类
print(响应对象.headers)          # 获得头部信息
```

2.3 爬取baidu首页

2020年4月11日 11:12

例如：爬取baidu首页

```
import requests
# 1.指定url
url = 'https://www.baidu.com/'
# 2.发起请求
响应对象 = requests.get(url)
# 3.获取响应数据.text返回的是字符串形式的响应数据
响应数据 = 响应对象.text
print(响应数据)
# 4.存储
with open('./bilibili.html','w',encoding='utf-8') as 变量名:
    变量名.write(响应数据)
print('爬虫数据完毕！！！')
```

2.4 网页采集器

2020年4月11日 11:13

```
import requests
url= 'https://www.baidu.com/s?cl=3&'   # 原url= 'https://www.baidu.com/s?cl=3&wd=摄影'
# 处理url携带的参数：封装到字典
用户输入 = input('请输入要搜索的内容：')
字典 = {'wd':用户输入}
# 对指定url发起请求
响应对象 = requests.get(url,字典)
# 获取响应数据
响应数据 = 响应对象.text
# 存储
文件名 = 字典['wd']+'.html'
with open(文件名,'w',encoding='utf-8') as 变量名:
    变量名.write(响应数据)
print(文件名,'保存成功！！！')
```

2.5 UA伪装 【必需用】

2020年4月11日 11:29

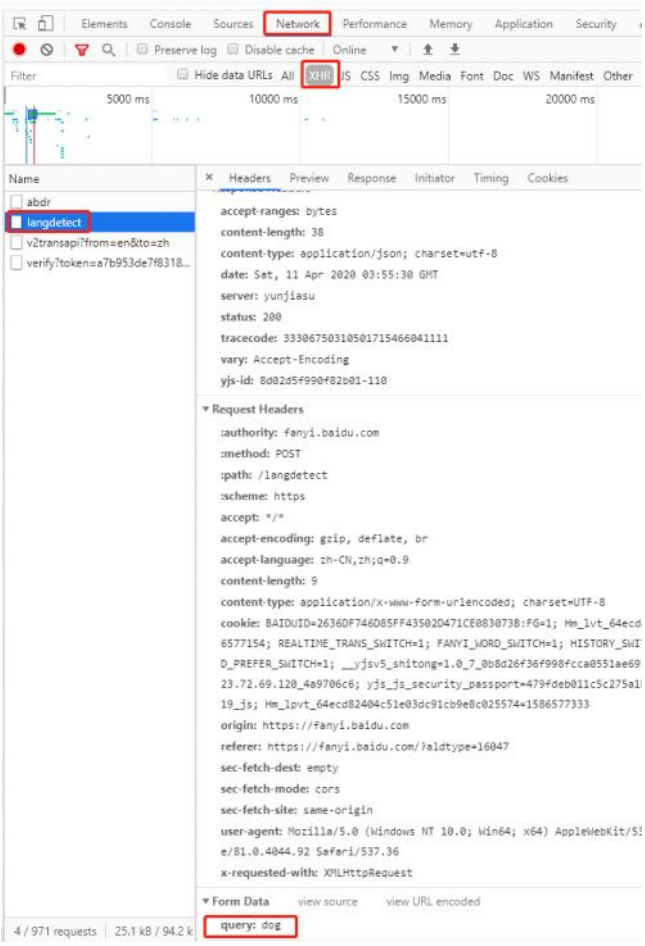
使用get发请求不是浏览器，是爬虫程序，网站为了服务器压力问题，它们不欢迎非人类访问。
所以我们要把程序伪装成浏览器去访问网站。简称UA伪装

```
import requests
# UA伪装
UA伪装 = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.92 Safari/537.36'}
url= 'https://www.baidu.com/s?cl=3&'
# 处理url携带的参数：封装到字典
用户输入 = input('请输入要搜索的内容：')
字典 = {'wd':用户输入}
# 对指定url发起请求
响应对象 = requests.get(url,字典,headers=UA伪装)     # url=url,params=字典,headers=UA伪装
# 获取响应数据
响应数据 = 响应对象.text
# 存储
文件名 = 字典['wd']+'.html'
with open(文件名,'w',encoding='utf-8') as 变量名:
    变量名.write(响应数据)
print(文件名,'保存成功！！！')
```

2.6 GET和POST区别

2020年4月11日 11:57

GET是我们都熟悉的。它用于请求网页文本。当你在浏览器输入某个网站地址，它会直接访问该网站的web服务器，用GET 。 第二个最有名的是POST，它经常被用在提交信息，比如请求买什么东西。每当提交一个去做什么事情(像使用信用卡处理一笔交易)的请求时，你可以使用POST。这是关键，因为GET的URL可以被搜索引擎索引，并通过搜索引擎访问。虽然大部分页面你希望被索引，但是少数类似订单处理的页面你是不希望被索引的，那么就需要用到POST。



2.7 有道翻译

2020年4月11日 12:09

```
import requests
# 1.UA伪装
UA伪装 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.92 Safari/537.36'}
# 2.指定url
post_url = 'http://fanyi.youdao.com/translate?smartresult=dict&smartresult=rule' # 来自抓包工具Request URL
# 3.post请求参数处理（与get请求一样）
输入 = input('请输入你要翻译的内容：')
data = {}
data['i'] = 输入
data['doctype'] = 'json'
data['from'] = 'AUTO'
data['to'] = 'AUTO'
data['client'] = 'fanyideskweb'
# 4.请求发送
响应对象 = requests.post(url=post_url,params=data,headers=UA伪装)
# 5.获取响应数据：json()方法返回是一个对象（只有服务器是json类型才能使用）
# content-type: application/json; charset=utf-8
字典对象 = 响应对象.json()
# 存储
# 文件名 = 输入 + '.json'
# fp = open(文件名,'w',encoding='utf-8')
# json.dump(字典对象,fp=fp,ensure_ascii=False) # 有中文不能用ASCII
# print(字典对象)
print(f'翻译结果：{字典对象["translateResult"][0][0]["tgt"]}')
```

2.8 豆瓣电影

2020年4月11日 14:47

```
import requests
import json
UA伪装 = UA伪装 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.92 Safari/537.36'}
url = 'https://movie.douban.com/j/chart/top_list' # ?type=20&interval_id=100%3A90&action=&start=40&limit=20
data = {
    "type":"20",
    "interval_id":"100:90",
    "action":"",
    "start":"0", # 从库中的第几部电影去取,索引从0开始
    "limit":"20"  # 一次取出的个数
}
响应对象 = requests.get(url,data,headers=UA伪装)
响应数据 = 响应对象.json()
fp = open('./douban.json','w',encoding='utf-8')
json.dump(响应数据,fp=fp,ensure_ascii=False)
print('结束！！！')
```

总结

2020年4月14日 16:15

3.1 get请求

参数是字典，我们也可以传递json类型的参数：

```
import requests

url = "http://www.baidu.com/s"
params = {'wd': '尚学堂'}
response = requests.get(url, params=params)
print(response.url)
response.encoding = 'utf-8'
html = response.text
# print(html)
```

3.2 post请求

参数是字典，我们也可以传递json类型的参数：

```
url = "http://www.sxt.cn/index/login/login.html"
formdata = {
    "user": "17703181473",
    "password": "123456"
}
response = requests.post(url, data=formdata)
response.encoding = 'utf-8'
html = response.text
# print(html)
```

3.3 自定义请求头部

> 伪装请求头部是采集时经常用的，我们可以用这个方法来隐藏：

```
headers = {'User-Agent': 'python'}
r = requests.get('http://www.zhidaow.com', headers = headers)
print(r.request.headers['User-Agent'])
```

3.4 设置超时时间

> 可以通过timeout属性设置超时时间，一旦超过这个时间还没获得响应内容，就会提示错误

```
requests.get('http://github.com', timeout=0.001)
```

3.5 代理访问

> 采集时为避免被封IP，经常会使用代理，requests也有相应的proxies属性

```
import requests

proxies = {
  "http": "http://10.10.1.10:3128",
  "https": "https://10.10.1.10:1080",
}

requests.get("http://www.zhidaow.com", proxies=proxies)
```

> 如果代理需要账户和密码，则需这样

```
proxies = {
    "http": "http://user:pass@10.10.1.10:3128/",
}
```

3.6 session自动保存cookies

> seesion的意思是保持一个会话，比如 登陆后继续操作(记录身份信息) 而requests是单次请求的请求，身份信息不会被记录

```
# 创建一个session对象
s = requests.Session()
# 用session对象发出get请求，设置cookies
s.get('http://httpbin.org/cookies/set/sessioncookie/123456789')
```

3.7 ssl验证

```
# 禁用安全请求警告
requests.packages.urllib3.disable_warnings()

resp = requests.get(url, verify=False, headers=headers)
```

4 获取响应信息

代码	含义
resp.json()	获取响应内容（以json字符串）
resp.text	获取响应内容 (以字符串)
resp.content	获取响应内容（以字节的方式）
resp.headers	获取响应头内容
resp.url	获取访问地址
resp.encoding	获取网页编码
resp.request.headers	请求头内容
resp.cookie	获取cookie

三、数据解析

2020年4月14日 11:36

3.1 【正则】批量下载图片【京客隆超市】

2020年4月14日 18:54

```
import requests
import re
import os
# 创建一个文件夹，保存所有图片
if not os.path.exists('./海报'):
    os.mkdir('./海报')    # Python基础第10课
url = 'http://www.jkl.com.cn/cn/phoLis.aspx?id=697'
for i in range(1,5):
    data = {'__EVENTARGUMENT':'i'}
    UA伪装 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/81.0.4044.92 Safari/537.36'}
    # 使用通过爬虫对url对应的一整张页面进行爬取
    响应数据 = requests.get(url=url,params=data,headers=UA伪装).text
    # 使用聚焦爬虫将页面中所有的图片进行解析/提取
    # (?=exp):零宽度正回顾后发断言，它断言自身出现的位置的前面能匹配表达式exp
    # (?<=exp): 零宽度正回顾后发断言，它断言自身出现的位置的前面能匹配表达式exp
    正则 = '(?<=\<img src\=").+?(?=" width\="165" height\="117")'
    图片地址列表 = re.findall(正则,响应数据)
    for ulr2 in 图片地址列表:
        # 拼接出一个完整的图片地址
        ulr2 ='http://www.jkl.com.cn'+ulr2
        # 请求到了图片的二进制数据
        图片数据 = requests.get(url=ulr2,params=data,headers=UA伪装).content
        # 生成图片名称
        图片名字 = str(i)+ulr2.split('/')[-1]    # Python基础 第2课
        # 图片存储的路径
        图片路径 = './海报/'+图片名字
        # 持久化存储
        with open(图片路径,'wb') as 变量名:
            变量名.write(图片数据)
            print(图片名字,'下载成功！！！')
```

3.2 bs4进行数据解析

2020年4月14日 11:37

- bs4数据解析的原理：

1. 实例化一个BeautifulSoup对象，并且将页面源码数据加载到该对象中
2. 通过调用BeautiflSoup对象中相关的属性或方法进行标签定位和数据提取

- 环境安装：

```
pip3 install bs4
pip3 install lxml
```

- 如果实例化BeautiflSoup对象：

```
from bs4 import BeautifulSoup
```

a. 将本地的html文档中的数据加载到该对象中

b. 将互联网上获取的页面源码加载到该对象中

```
from bs4 import BeautifulSoup
# 将本地的html文档中的数据加载到该对象中
fp = open('./test.html','r',encoding='utf-8')
soup = BeautifulSoup(fp,'lxml')
print(soup)


# 将互联网上获取的页面源码加载到该对象中
page_text = 响应对象.text
soup = BeautifulSoup(page_text,'lxml')
```

- 提供的用于数据解析的方法和属性
  a. soup.tagName:返回的是文档中第一次出现的tagName对应的标签，例如soup.a或soup.div
  b. soup.find():
    i. soup.find('tagName'):等同于soup.div
    ii. 属性定位 soup.find('div',class_/id/attr='song')
  c. soup.findall('tagName'):返回符合要求的所有标签（列表）
  d. soup.select():
    i. soup.select('某种选择器（id,class,标签...选择器）')，返回的是一个列表
    ii. soup.select('.tang > ul > li > a')[0]　大于号表示的是一个层级
    iii. soup.select('.tang > ul a')[0]　空格表式多个层级

- 获取标签之间的文本数据
  - soup.a.text/srting/get_text()
  - text/get_text():可以获取某一个标签中所有的文本内容

    - **string:只能获取该标签下面直系的文本内容**
- **获取标签中属性值:**
    - **soup.a['href']**

3.2.1 【bs4】爬取西游记

2020年4月14日 21:19

```
import requests
from bs4 import BeautifulSoup
# 对首页的页面数据进行爬取
url = 'http://www.shicimingju.com/book/xiyouji.html'
UA伪装 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.92
Safari/537.36'}
响应数据 = requests.get(url=url,headers=UA伪装).text
# 在首页中解析出章节的标题和详情页的url
# 1.实例化BeautifulSoup对象，需要将页面源码数据加载到该对象中
soup = BeautifulSoup(响应数据,'lxml')
# 2.解析章节标题和详情页的url
li标签列表 = soup.select('.book-mulu > ul > li')
变量名 = open('./xiyouji.txt','w',encoding='utf-8')
# 要获取所有li标签中的a标签
for li in li标签列表:
    章节标题 = li.a.string
    详情页的url = 'http://www.shicimingju.com'+li.a['href']
    # 对详情页发起请求，解析出章节内容
    详情页数据 = requests.get(url=详情页的url,headers=UA伪装).text
    # 解析出详情页中相关的章节内容
    详情页soup = BeautifulSoup(详情页数据,'lxml')
    div标签 = 详情页soup.find('div',class_='chapter_content')
    # 将div标签中所有的文本内容获取
    章节内容 = div标签.text
    变量名.write(章节标题+':'+章节内容+'\n')
    print(章节标题,'爬取成功！！！')
```

3.2.2 【bs4】批量下载图片【京客隆超市】

2020年4月16日 2:12

```
import requests
import os
from bs4 import BeautifulSoup
# 创建一个文件夹，保存所有图片
if not os.path.exists('./海报'):
    os.mkdir('./海报')    # Python基础第10课
url = 'http://www.jkl.com.cn/cn/phoLis.aspx?id=697'
for i in range(1,5):
    data = {'__EVENTARGUMENT':'i'}
    UA伪装 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.92
Safari/537.36'}
    响应数据 = requests.get(url=url,params=data,headers=UA伪装).text
    soup = BeautifulSoup(响应数据,'lxml')
    标签列表 = soup.select('.proLis > ul > li')
    for li in 标签列表:
        ulr2 = 'http://www.jkl.com.cn/' + li.img['src']
        图片数据 = requests.get(url=ulr2,params=data,headers=UA伪装).content
        图片名字 = str(i)+ulr2.split('/')[-1]    # Python基础 第2课
        图片路径 = './海报/'+图片名字
        with open(图片路径,'wb') as 变量名:
            变量名.write(图片数据)
            print(图片名字,'下载成功！！！')
```

3.3 xpath解析

2020年4月14日 21:19

一、解析原理

1、实例化一个etree的对象，且需要将被解析的页面源码数据加载到该对象中。

2、调用etree对象中的xpath方法结合着xpath表达式实现标签的定位和内容的捕获。

二、环境安装

pip3 install lxml

三、导入类

from lxml import etree

四、如何实例化一个etree对象

1、将本地的html文档中的源码数据加载到etree对象中

etree.parse(文件路径)

2、可以将从互联网上获取源码数据加载到该对象中

etree.HTML(响应数据)

五、xpath表达式

/：表示的是从根节点开始定位，表示的是一个层级

//：表示是多个层级，可以表示从任意位置开始定位

属性定位：//div[@class='song'] tag[@attrName="attrValue"]

索引定准：//div[@class='song']/p[3] 索引从1开始的

取文本：

```
r = tree.xpath('//div[@class="tang"]//li[5]/a/text()')[0]
```

/text():获取的是标签中真系的文本内容

//text():标签中非真系的文本内容（所有的文本内容）

取属性：

```
r = tree.xpath('//div[@class="song"]/img/@src')
```

/@arrtName 例如 ==>img/src

3.3.1 【xpath】批量下载图片【京客隆超市】

2020年4月16日 4:25

```
import requests
import os
from lxml import etree

# 创建一个文件夹，保存所有图片
if not os.path.exists('./海报'):
    os.mkdir('./海报')
url = 'http://www.jkl.com.cn/cn/phoLis.aspx?id=697'
for i in range(1, 5):
    data = {'__EVENTARGUMENT': i}
    UA伪装 = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.92
Safari/537.36'}
    响应数据 = requests.get(url=url, params=data, headers=UA伪装).text
    tree = etree.HTML(响应数据)
    标签列表 = tree.xpath('//div[@class="proLis"]//img/@src')
    for li in 标签列表:
        ulr2 = 'http://www.jkl.com.cn' + li
        图片数据 = requests.get(url=ulr2, params=data, headers=UA伪装).content
        图片名字 = str(i) + ulr2.split('/')[-1]
        图片路径 = './海报/' + 图片名字
        with open(图片路径, 'wb') as 变量名:
            变量名.write(图片数据)
            print(图片名字, '下载成功！！！')
```