第一周：网路爬虫之规则

单元 1：Requests 库入门

1.Requests 库综述

表 1.1 Requests 库的 7 个主要方法

方法	说明
request()	构造一个请求，支撑以下各方法的基础方法
get()	获取 HTML 网页的主要方法，对应于 HTTP 的 GET
head()	获取 HTML 网页头的主要方法，对应于 HTTP 的 HEAD
post()	向 HTML 网页提交 POST 请求的方法，对应于 HTTP 的 POST
put()	向 HTML 网页提交 PUT 请求的方法，对应于 HTTP 的 PUT
patch()	向 HTML 网页提交局部修改请求，对应于 HTTP 的 PATCH
delete()	向 HTML 页面提交删除请求，对应于 HTTP 的 DELETE

2.Requests 库的 get()方法

最简单的获取网址的方式是：

r= requests.get("http://www.yushuai.me")

这之中，requests 是构造了一个向服务器请求资源的 Request 对象，r 是返回一个包含服务器资源的 Response 对象。get 完全的使用方法是：

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

其中，url 是拟获取页面的 url 链接；params 是 url 中的额外参数，字典或字节流格式，可选；**kwargs 是 13 个控制参数。

表 1.2 Response 对象的属性

方法	说明
r.status_code	HTTP 请求的返回状态。200 表示连接成功，404 表示失败。
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（备用编码方式）
r.content	HTTP 响应内容的二进制形式

注意：

r.encoding：如果 header 中不存在 charset，则认为编码为 ISO-8859-1，所以 header 中编码不一定正确。

r.apparent_encoding：根据网页内容分析出的编码方式。

2.爬取网页的通用代码框架

表 1.3 Requests 库的异常

异常	说明
requests.ConnectionError	网络连接错误异常，如 DNS 查询失败、拒绝连接等
requests.HTTPError	HTTP 错误异常
requests.URLRequired	URL 缺失异常
requests.TooManyRedirects	超过最大重定向次数，产生重定向异常

requests.ConnectTimeout	连接远程服务器超时异常
requests.Timeout	请求 URL 超时，产生超时异常
r.raise_for_status()	如果不是 200，产生异常 requests.HTTPError

爬取网页的通用代码框架如下所示：

```
import requests

def getHTMLText(url):
    try:
        r=requests.get(url,timeout=30)
        r.raise_for_status()#如果状态不是 200，引发 HTTPError 异常
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return "产生异常"

if __name__ =="__main__":
    url = "http://www.baidu.com"
    print(getHTMLText(url))
```

3.HTTP 协议及 Requests 库方法

表 1.4 HTTP 协议对资源的操作

方法	说明
GET	请求获取 URL 位置的资源
HEAD	请求获取 URL 位置资源的相应消息报告，即获取头部信息
POST	请求向 URL 位置的资源后附加新的数据（**不改变原有，只增加**）
PUT	请求向 URL 位置存储一个资源，**覆盖原 URL 位置的资源**
PATCH	请求局部更新 URL 位置的资源，即改变该处资源的部分内容
DELETE	请求删除 URL 位置存储的资源

注意：理解 PATCH 和 PUT 的区别

假设 URL 位置有一组数据 UserInfo，包括 UserID、UserName 等 20 个字段。需求：用户修改了 UserName，其他不变。

（1）若使用 PATCH，仅向 Url 提交 UserName 的局部更新请求。

（2）若使用 PUT，必须将所有 20 个字段一并提交到 URL，未提及的字段会被删除。

4.Requests 库主要方法解析

（1）requests.request(method,url,**kwargs)

其中，method 是请求方式；url 是拟获取页面的 url 链接；**kwargs 是控制访问的参数，共 13 个。

method 请求方式包括：

①r=requests.request('GET',url,**kwargs)

②r=requests.request('HEAD',url,**kwargs)

③r=requests.request('POST',url,**kwargs)

④r=requests.request('PUT',url,**kwargs)

⑤r=requests.request('PATCH',url,**kwargs)

⑥r=requests.request('delete',url,**kwargs)

⑦r=requests.request('OPTIONS',url,**kwargs)向服务器获取服务器和客户端通信的参数。

**kwargs 参数如表 1.5 所示。

表 1.5 参数表

参数	说明
params	字典或字节序列，作为参数增加到 url 中。
data	字典、字节序列或文件对象，作为 Request 的内容。
json	JSON 格式的数据，作为 Request 的内容。
headers	字典，HTTP 定制头。
cookies	字典或 CookieJar，Request 中的 cookie。
auth	元组，支持 HTTP 认证功能。
files	字典类型，传输文件。
timeout	设定超时时间，秒为单位。
proxies	字典类型，设计访问代理服务器，可以增加登录认证。
allow_redirects	默认为 True，重定向开关。
stream	默认为 True，获取内容立即下载开关。
verify	默认 True，认证 SSL 证书开关。
cert	本地 SSL 证书路径。

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

url：拟获取页面的 url 链接。

params：url 的额外参数，字典或字节流格式，可选。

**kwargs：12 个控制访问参数（上一个中除了 params）。

（3）requets.head(url,**kwargs)

**kwargs 与 request 一样，不再累述。

（4）requests.post(url,data=None,json=None,**kwargs)

和 request 一样，不再累述。

其它也一样，不再累述。

5.例：Requests 库的爬取性能分析

```
import time
import requests

def getHTMLText(url):
    try:
        r=requests.get(url,timeout=30)
        r.raise_for_status()#如果状态不是 200，引发 HTTPError 异常
        r.encoding = r.apparent_encoding
        return r.text
    except:
        return "产生异常"

if __name__ =="__main__":
    url = "http://www.baidu.com/"
    start_time =time.time()
    for i in range(100):
```

```
        getHTMLText(url)
    end_time = time.time()
    dur_time = end_time - start_time
    print("爬取 100 次网页所需时间：{:.3f}s".format(dur_time))
```

单元 2：网络爬虫的“盗亦有道”

1.网络爬虫规模及问题

网络爬虫的尺寸主要分为下面三类：

（1）爬取网页：小规模，数据量小，爬取速度不敏感，使用 Requests 库。

（2）爬取网站：中规模，数据规模较大，爬取速度敏感，使用 Scrapy 库。

（3）爬取全网：大规模，搜索引擎爬取速度关键，定制开发。

网络爬虫带来的问题：

骚扰问题、法律风险、泄漏问题。

2.Robots 协议

告知爬虫哪些可以爬取，哪些不可以爬取。

单元 3：实践

1.向百度搜索提交搜索内容

代码：

```
import requests
keyword = 'Python'
try:
    kv = {'wd':'keyword'}
    r= requests.get("http://www.baidu.com/s",params=kv)
    print(r.raise_for_status)
    print(r.request.url)
    print(len(r.text))
except:
    print("爬取失败")
```

结果：

```
<bound method Response.raise_for_status of <Response [200]>>
http://www.baidu.com/s?wd=keyword
245051
```

（注意：原来字典的'wd':'keyword'，在连接的时候修改为了 wd=keyword）

2.网络图片的爬取和存储

代码：

```
import requests
import os
import time
root = "D://pics//"
url = "http://image.nationalgeographic.com.cn/2018/0327/20180327031900693.jpg"
```

```
path = root +url.split('/')[-1]
start_time = time.time()
try:
    if not os.path.exists(root):
        os.mkdir(root)
    if not os.path.exists(path):
        r= requests.get(url)
        big = len(r.text)
        with open(path,'wb') as f:
            f.write(r.content)
            f.close()
            print("文件保存成功")
    else:
        print("文件已经存在")
except:
    print("爬取失败")
end_time = time.time()
dur_time = end_time-start_time
print("保存图片用时{0:.3f}s，图片大小为{1}".format(dur_time,big))
```

结果：

图片得到保存，且输出：

文件保存成功

保存图片用时 14.035s，图片大小为 682413

实例 5：IP 地址归属地的自动查询

```
import requests

url = "http://m.ip138.com/ip.asp?ip="
try:
    r=requests.get(url+'113.133.219.95')
    r.raise_for_status()
    r.encoding = r.apparent_encoding
    print(r.text[-500:])
except:
    print("爬取失败")
```