requests库

1. requests库的get()用法

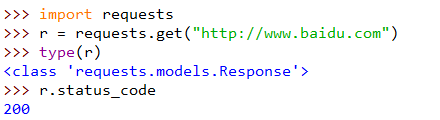
R = requests.get(url)

右边: 构造一个向服务器请求资源的**Requests**对象

左边：返回一个包含服务器资源的**Response**对象

**Response**对象包含爬虫返回的内容

1.1 **Response**对象

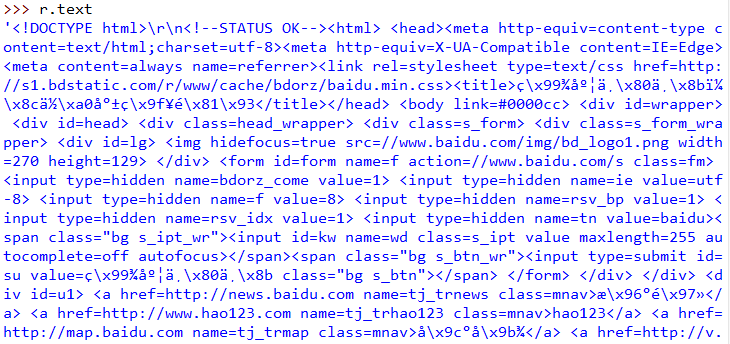


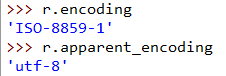
这里的r即为Response对象（参见type(r)）

status\_code为Response对象的一个属性，表示HTTP请求的返回状态，200表示连接成功，404表示失败

1.1.1 **Response**对象的属性

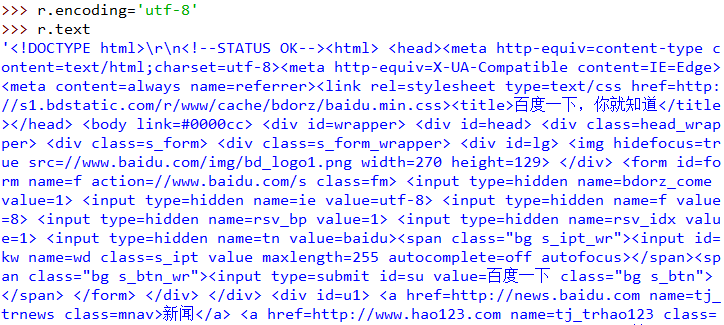
|  |  |
| --- | --- |
| 属性 | 说明 |
| r.status\_code | HTTP请求的返回状态，200表示连接成功，404表示失败 |
| r.text | HTTP相应内容的字符串形式，即url对应的页面内容 |
| r.encoding | 从HTTP header中猜测的响应内容编码方式 |
| r.apparent\_coding | 从内容中分析出的响应内容编码方式 |
| r.content | HTTP响应内容的二进制形式 |





注意：r.encoding只是从header中猜测响应内容的编码方式，r.text根据r.encoding显示网页内容，则此时可能出现不了中文

而r.apparent\_coding是根据网页内容分析出编码方式，所以将r.encoding改为’utf-8’后，可以显示出中文



1. 爬取网页的通用代码框架

|  |
| --- |
| **import** requests  **def get\_HTML\_Text(**url**):  try:** r **=** requests.get**(**url,timeout**=**30**)** r.raise\_for\_status**()** r.encoding **=** r.apparent\_encoding  **return** r.text  **except:  return "产生异常"  if** \_\_name\_\_**=="\_\_main\_\_":** url **= "http://www.baidu.com"** print**(**get\_HTML\_Text**(**url**))** |

HTTP协议

HTTP ：Hpertext Transfer Protocol ,超文本传输协议

HTTP是一个基于“请求与响应”模式的、无状态的应用层协议，HTTP协议采用URL作为定位网络资源的标识

URL：URL是通过HTTP协议存取资源的Internet路径，一个URL对应一个数据资源

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位置资源的相映消息报告，即获得该资源的头部信息 |
| POST | 请求向URL位置资源后附加新的数据 |
| 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 |

HTTP协议与Requests库

|  |  |  |
| --- | --- | --- |
| HTTP协议方法 | Requests库方法 | 功能一致性 |
| GET | requests.get() | 一致 |
| HEAD | requests.head() | 一致 |
| POST | requests.post() | 一致 |
| PUT | requests.put() | 一致 |
| PATCH | requests.patch() | 一致 |
| DELETE | requests.delete() | 一致 |

1.requests库的get()用法(续)

get(url, params=None, \*\*kwargs)

Sends a GET request.

:**param(参数) url**: URL for the new :class:`Request` object.

:**param params**: (optional) Dictionary or bytes to be sent in the query(问号) string for the :class:`Request`.

:**param \\*\\*kwargs**: Optional arguments that ``request`` takes.

:return: :class:`Response <Response>` object

:rtype: requests.Response

None

param \\*\\*kwargs:的12个可选参数

:**param data**: (optional) Dictionary, bytes, or file-like object to send in the body of the :class:`Request`.

:**param json**: (optional) json data to send in the body of the :class:`Request`.

:**param headers**: (optional) Dictionary of HTTP Headers to send with the :class:`Request`.

:**param cookies**: (optional) Dict or CookieJar object to send with the :class:`Request`.

:**param files**: (optional) Dictionary of ``'name': file-like-objects`` (or ``{'name': file-tuple}``) for multipart encoding upload.

``file-tuple`` can be a 2-tuple ``('filename', fileobj)``, 3-tuple ``('filename', fileobj, 'content\_type')``

or a 4-tuple ``('filename', fileobj, 'content\_type', custom\_headers)``, where ``'content-type'`` is a string

defining the content type of the given file and ``custom\_headers`` a dict-like object containing additional headers

to add for the file.

:**param auth**: (optional) Auth tuple to enable Basic/Digest/Custom HTTP Auth.

:**param timeout**: (optional) How long to wait for the server to send data

before giving up, as a float, or a :ref:`(connect timeout, read

timeout) <timeouts>` tuple.

:type timeout: float or tuple

:**param allow\_redirects**: (optional) Boolean. Enable/disable GET/OPTIONS/POST/PUT/PATCH/DELETE/HEAD redirection. Defaults to ``True``.

:type allow\_redirects: bool

:**param proxies**: (optional) Dictionary mapping protocol to the URL of the proxy.

:**param verify**: (optional) whether the SSL cert will be verified. A CA\_BUNDLE path can also be provided. Defaults to ``True``.

:**param stream**: (optional) if ``False``, the response content will be immediately downloaded.

:**param cert**: (optional) if String, path to ssl client cert file (.pem). If Tuple, ('cert', 'key') pair.

1.2各个参数使用方法

:**param params**: (optional) Dictionary or bytes to be sent in the query(问号) string for the :class:`Request`.

例：百度、360搜索引擎的关键词提交

百度关键词接口：

<http://www.baidu.com/s?wd=keyword>

360关键词接口:

<http://www.so.com/s?q=keyword>

这里就是问号表达式，所以使用方法为：

|  |
| --- |
| **import** requests  kv **= {'wd': 'python'}** r **=** requests.get**("http://www.baidu.com/s"**, params**=**kv**)** print**(**r.status\_code**)** print**(**r.text**[:**1000**])**  200  <!DOCTYPE html>  <!--STATUS OK-->  <html>  <head>    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <meta http-equiv="content-type" content="text/html;charset=utf-8">  <meta content="always" name="referrer">  <meta name="theme-color" content="#2932e1">  <link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />  <link rel="icon" sizes="any" mask href="//www.baidu.com/img/baidu.svg">  <link rel="search" type="application/opensearchdescription+xml" href="/content-search.xml" title="百度搜索" />      <title>python\_百度搜索</title>  ...(后面还有很多，就不截取了) |

所以这里使用params 后，与之前直接访问百度主界面是不一样的，它跳到了百度搜索关键词python的界面

同理用360搜索python

|  |
| --- |
| **import** requests  kv **= {'q': 'python'}** r **=** requests.get**("http://www.so.com/s"**, params**=**kv**)** print**(**r.status\_code**)** print**(**r.text**[:**1000**])**  200  <!DOCTYPE html>  <!--[if lt IE 7 ]><html class="ie6"><![endif]-->  <!--[if IE 7 ]><html class="ie7"><![endif]-->  <!--[if IE 8 ]><html class="ie8"><![endif]-->  <!--[if IE 9 ]><html class="ie9"><![endif]-->  <!--[if (gt IE 9)|!(IE)]><!--><html><!--<![endif]-->  <head>  <meta charset="utf-8">  <meta content="always" name="referrer">  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">  <title>python\_360搜索</title>  <link rel="dns-prefetch" href="//p.ssl.qhimg.com"><link rel="dns-prefetch" href="//s.ssl.qhimg.com"><link rel="dns-prefetch" href="//s.ssl.qhres.com">  <link rel="shortcut icon" href="https://s.ssl.qhres.com/static/52166db8c450f68d.ico" type="image/x-icon">  <link rel="search" type="application/opensearchdescription+xml" href="https://www.so.com/soopensearch.xml" title="360搜索">  ... |

\*\*kwargs里的

:**param headers**: (optional) Dictionary of HTTP Headers to send with the :class:`Request`.

|  |
| --- |
| **import** requests  r **=** requests.get**("http://www.baidu.com")** print**(**r.status\_code**)** print**(**r.request.headers**)**  200  {'User-Agent': 'python-requests/2.13.0', 'Accept-Encoding': 'gzip, deflate', 'Accept': '\*/\*', 'Connection': 'keep-alive'}  Process finished with exit code 0 |

当不使用headers参数时，用r.request.headers查看向服务器提交请求的头文件，发现'User-Agent': 'python-requests/2.13.0'，这相当于告诉服务器这是一个爬虫发送的请求，有的网站会识别，并阻止访问，所以想修改头文件是需要用到headers参数

|  |
| --- |
| **import** requests  kv **= {'User-Agent': 'Mozilla/5.0'}** r **=** requests.get**("http://www.baidu.com"**, headers**=**kv**)** print**(**r.status\_code**)** print**(**r.request.headers**)**  200  {'User-Agent': 'Mozilla/5.0', 'Accept-Encoding': 'gzip, deflate', 'Accept': '\*/\*', 'Connection': 'keep-alive'}  Process finished with exit code 0 |

这里的User-Agent就相当于被修改了，相当于模仿了一个浏览器的访问

3.requests库的head()用法

|  |
| --- |
| **import** requests  r **=** requests.head**("http://www.baidu.com")** print**(**r.headers**)**  {'Server': 'bfe/1.0.8.18', 'Date': 'Thu, 23 Mar 2017 14:16:38 GMT', 'Content-Type': 'text/html', 'Last-Modified': 'Mon, 13 Jun 2016 02:50:45 GMT', 'Connection': 'Keep-Alive', 'Cache-Control': 'private, no-cache, no-store, proxy-revalidate, no-transform', 'Pragma': 'no-cache', 'Content-Encoding': 'gzip'}  Process finished with exit code 0 |

|  |
| --- |
| **import** requests  r **=** requests.head**("http://www.baidu.com")** print**(**r.text**)**  Process finished with exit code 0 |

注意只用head方法，r.text什么都没有

|  |
| --- |
| **import** requests  r **=** requests.head**("http://www.baidu.com")** print**(**r.encoding,r.apparent\_encoding**)**  ISO-8859-1 None  Process finished with exit code 0 |

注意只用head方法，r.apparent\_encoding为None

1. requests库的post()用法

????

应用实例

1. 爬取网上的一张图片，并下载下来

|  |
| --- |
| **import** requests  r **=** requests.get**("http://pic.58pic.com/58pic/12/40/48/158PICT58PICEQt.jpg")** print**(**r.status\_code**)** f **=** open**('F://123.jpg'**, **'wb')** f.write**(**r.content**)** f.close**()** |

注意网上的图片（以.jpg格式为例）应当以.jpg结尾，然后再利用python的文件读写知识，将r的二进制写进文件里即可。所以如上代码，F盘根目录下就会有一张123.jpg文件