

网络爬虫和相关工具

网络爬虫的概念

网络爬虫（web crawler），以前经常称之为网络蜘蛛（spider），是按照一定的规则自动浏览万维网并获取信息的机器人程序（或脚本），曾经被广泛的应用于互联网搜索引擎。使用过互联网和浏览器的人都知道，网页中除了供用户阅读的文字信息之外，还包含一些超链接。网络爬虫系统正是通过网页中的超链接信息不断获得网络上的其它页面。正因如此，网络数据采集的过程就像一个爬虫或者蜘蛛在网络上漫游，所以才被形象的称为网络爬虫或者网络蜘蛛。

爬虫的应用领域

在理想的状态下，所有ICP（Internet Content Provider）都应该为自己的网站提供API接口来共享它们允许其他程序获取的数据，在这种情况下爬虫就不是必需品，国内比较有名的电商平台（如淘宝、京东等）、社交平台（如腾讯微博等）等网站都提供了自己的Open API，但是这类Open API通常会对可以抓取的数据以及抓取数据的频率进行限制。对于大多数的公司而言，及时的获取行业相关数据是企业生存的重要环节之一，然而大部分企业在行业数据方面的匮乏是其与生俱来的短板，合理的利用爬虫来获取数据并从中提取出有商业价值的信息是至关重要的。当然爬虫还有很多重要的应用领域，下面列举了其中的一部分：

1. 搜索引擎
2. 新闻聚合
3. 社交应用
4. 舆情监控
5. 行业数据

合法性和背景调研

爬虫合法性探讨

1. 网络爬虫领域目前还属于拓荒阶段，虽然互联网世界已经通过自己的游戏规则建立起一定的道德规范(Robots协议，全称是“网络爬虫排除标准”)，但法律部分还在建立和完善中，也就是说，现在这个领域暂时还是灰色地带。
2. “法不禁止即为许可”，如果爬虫就像浏览器一样获取的是前端显示的数据（网页上的公开信息）而不是网站后台的私密敏感信息，就不太担心法律法规的约束，因为目前大数据产业链的发展速度远远超过了法律的完善程度。
3. 在爬取网站的时候，需要限制自己的爬虫遵守Robots协议，同时控制网络爬虫程序的抓取数据的速度；在使用数据的时候，必须要尊重网站的知识产权（从Web 2.0时代开始，虽然Web上的数据很多都是由用户提供的，但是网站平台是投入了运营成本的，当用户在注册和发布内容时，平台通常就已经获得了对数据的所有权、使用权和分发权）。如果违反了这些规定，在打官司的时候败诉几率相当高。

Robots.txt文件

大多数网站都会定义robots.txt文件，下面以淘宝的[robots.txt](#)文件为例，看看该网站对爬虫有哪些限制。

```
User-agent: Baiduspider
Allow: /article
Allow: /oshtml
Disallow: /product/
```

```
Disallow: /

User-Agent: Googlebot
Allow: /article
Allow: /oshtml
Allow: /product
Allow: /spu
Allow: /dianpu
Allow: /oversea
Allow: /list
Disallow: /

User-agent: Bingbot
Allow: /article
Allow: /oshtml
Allow: /product
Allow: /spu
Allow: /dianpu
Allow: /oversea
Allow: /list
Disallow: /

User-Agent: 360Spider
Allow: /article
Allow: /oshtml
Disallow: /

User-Agent: Yisouspider
Allow: /article
Allow: /oshtml
Disallow: /

User-Agent: Sogouspider
Allow: /article
Allow: /oshtml
Allow: /product
Disallow: /

User-Agent: Yahoo! Slurp
Allow: /product
Allow: /spu
Allow: /dianpu
Allow: /oversea
Allow: /list
Disallow: /

User-Agent: *
Disallow: /
```

注意上面robots.txt第一段的最后一行，通过设置“Disallow: /”禁止百度爬虫访问除了“Allow”规定页面外的其他所有页面。因此当你在百度搜索“淘宝”的时候，搜索结果下方会出现：“由于该网站的robots.txt文件存在限制指令（限制搜索引擎抓取），系统无法提供该页面的内容描述”。百度作为一个搜索引擎，至少在表面上遵守了淘宝网的robots.txt协议，所以用户不能从百度上搜索到淘宝内部的产品信息。

[淘宝网 - 淘!我喜欢](#) [官网](#)<https://www.taobao.com/> - 8262条评价

由于该网站的robots.txt文件存在限制指令（限制搜索引擎抓取），系统无法提供该页面的内容描述 - [了解详情](#)

[淘宝网触屏版](#)h5.m.taobao.com/ - 74条评价

由于该网站的robots.txt文件存在限制指令（限制搜索引擎抓取），系统无法提供该页面的内容描述 - [了解详情](#)

[淘宝_百度百科](#)

简介：**淘宝网**是亚太地区较大的网络零售、商圈，由阿里巴巴集团在2003年5月创立。**淘宝网**是中国深受欢迎的网购零售平台，拥有近5亿的注册用户数，每天有超过6000万的固定访客，同时每天的在线商品数已经超过了8亿件，平均每分钟售出4.8万件商品。截止2011年年底，**淘宝网**单日交易额峰值达到43.8亿元，创造270.8...

[发展沿革](#) [网站规模](#) [经济收入](#) [组织架构](#) [企业文化](#) [更多>>](#)<https://baike.baidu.com/>[淘宝网 - 淘!我喜欢](#)<https://login.taobao.com/> - 8262条评价

由于该网站的robots.txt文件存在限制指令（限制搜索引擎抓取），系统无法提供该页面的内容描述 - [了解详情](#)

相关工具介绍

HTTP协议

在开始讲解爬虫之前，我们稍微对HTTP（超文本传输协议）做一些回顾，因为我们在网页上看到的内容通常是浏览器执行HTML语言得到的结果，而HTTP就是传输HTML数据的协议。HTTP和其他很多应用级协议一样是构建在TCP（传输控制协议）之上的，它利用了TCP提供的可靠的传输服务实现了Web应用中的数据交换。按照维基百科上的介绍，设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法，也就是说这个协议是浏览器和Web服务器之间传输的数据的载体。关于这个协议的详细信息以及目前的发展状况，大家可以阅读阮一峰老师的[《HTTP协议入门》](#)、[《互联网协议入门》](#)系列以及[《图解HTTPS协议》](#)进行了解，下图是我在四川省网络通信技术重点实验室工作期间用开源协议分析工具Ethereal（抓包工具WireShark的前身）截取的访问百度首页时的HTTP请求和响应的报文（协议数据），由于Ethereal截取的是经过网络适配器的数据，因此可以清晰的看到从物理链路层到应用层的协议数据。

HTTP请求（请求行+请求头+空行+消息体）：

```
Frame 4 (563 bytes on wire, 563 bytes captured)
  Ethernet II, Src: Elitegro_f9:5d:82 (00:11:5b:f9:5d:82), Dst: Cisco_50:14:71 (00:1b:2a:50:14:71)
  Internet Protocol, Src: 192.168.58.136 (192.168.58.136), Dst: 119.75.213.51 (119.75.213.51)
  Transmission Control Protocol, Src Port: voicemail-port (3541), Dst Port: http (80), Seq: 1, Ack: 1, Len: 509
  Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Accept: */*\r\n
    Accept-Language: zh-cn\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727)\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: www.baidu.com\r\n
    Connection: Keep-Alive\r\n
    [truncated] Cookie: BAIDUID=72675E110453F51BEAC13B6277CE022F:FG=1; BDLFONT=0; BDUSS=VFJenJqBgZus21EM1dja3vyv\r\n
```

HTTP响应（响应行+响应头+空行+消息体）：

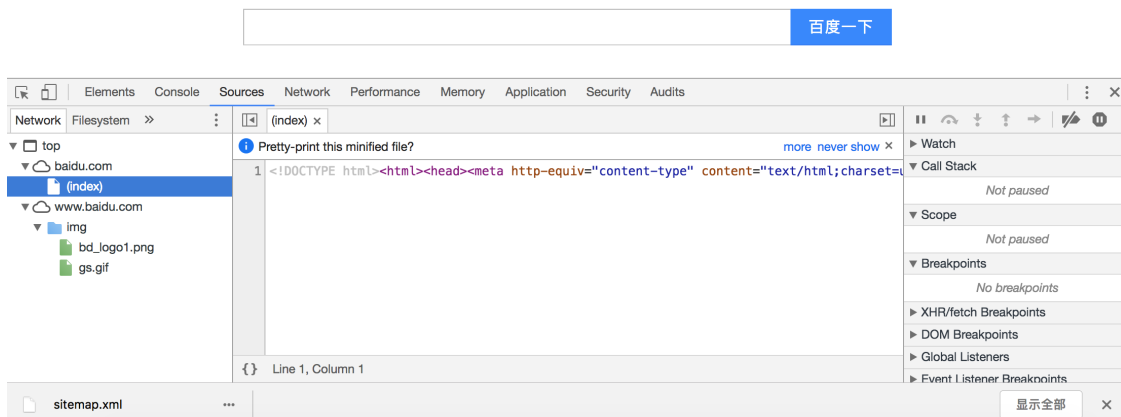
```
Frame 7 (668 bytes on wire, 668 bytes captured)
Ethernet II, Src: Cisco_50:14:71 (00:1b:2a:50:14:71), Dst: Elitegro_f9:5d:82 (00:11:5b:f9:5d:82)
Internet Protocol, Src: 119.75.213.51 (119.75.213.51), Dst: 192.168.58.136 (192.168.58.136)
Transmission Control Protocol, Src Port: http (80), Dst Port: voicemail (3541), Seq: 1421, Ack: 510, Len:
[Reassembled TCP Segments (2034 bytes): #6(1420), #7(614)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Date: Thu, 10 Sep 2009 04:02:47 GMT\r\n
    Server: BWS/1.0\r\n
  Content-Length: 1826\r\n
  Content-Type: text/html\r\n
  Cache-Control: private\r\n
  Expires: Thu, 10 Sep 2009 04:02:47 GMT\r\n
  Content-Encoding: gzip\r\n
  \r\n
  Content-encoded entity body (gzip): 1826 bytes -> 3719 bytes
Line-based text data: text/html
```

说明：但愿这两张如同泛黄照片般的截图帮助你大概的了解到HTTP是一个怎样的协议。

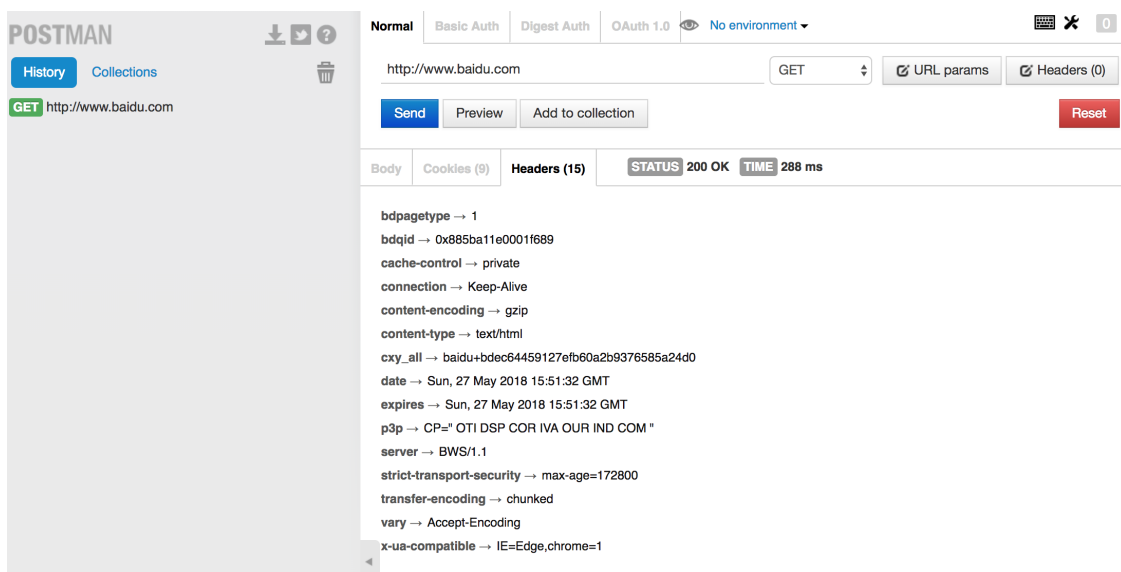
相关工具

1. Chrome Developer Tools：谷歌浏览器内置的开发者工具。

糯米 新闻 hao123 地图 视频 贴吧 登录 设置 更多产品



2. POSTMAN：功能强大的网页调试与RESTful请求工具。



3. HTTPie：命令行HTTP客户端。

```
pip3 install httpie
```

```
http --header http://www.scu.edu.cn
```

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: private, max-age=600
Connection: Keep-Alive
Content-Encoding: gzip
Content-Language: zh-CN
Content-Length: 14403
Content-Type: text/html
Date: Sun, 27 May 2018 15:38:25 GMT
ETag: "e6ec-56d3032d70a32-gzip"
Expires: Sun, 27 May 2018 15:48:25 GMT
Keep-Alive: timeout=5, max=100
Last-Modified: Sun, 27 May 2018 13:44:22 GMT
Server: VWebServer
Vary: User-Agent,Accept-Encoding
X-Frame-Options: SAMEORIGIN
```

4. BuiltWith: 识别网站所用技术的工具。

```
pip3 install builtwith
```

```
>>> import builtwith
>>> builtwith.parse('http://www.bootcss.com/')
{'web-servers': ['Nginx'], 'font-scripts': ['Font Awesome'], 'javascript-
frameworks': ['Lo-dash', 'Underscore.js', 'Vue.js', 'Zepto', 'jQuery'],
'web-frameworks': ['Twitter Bootstrap']}
>>>
>>> import ssl
>>> ssl._create_default_https_context = ssl._create_unverified_context
>>> builtwith.parse('https://www.jianshu.com/')
{'web-servers': ['Engine'], 'web-frameworks': ['Twitter Bootstrap', 'Ruby
on Rails'], 'programming-languages': ['Ruby']}
```

5. python-whois: 查询网站所有者的工具。

```
pip3 install python-whois
```

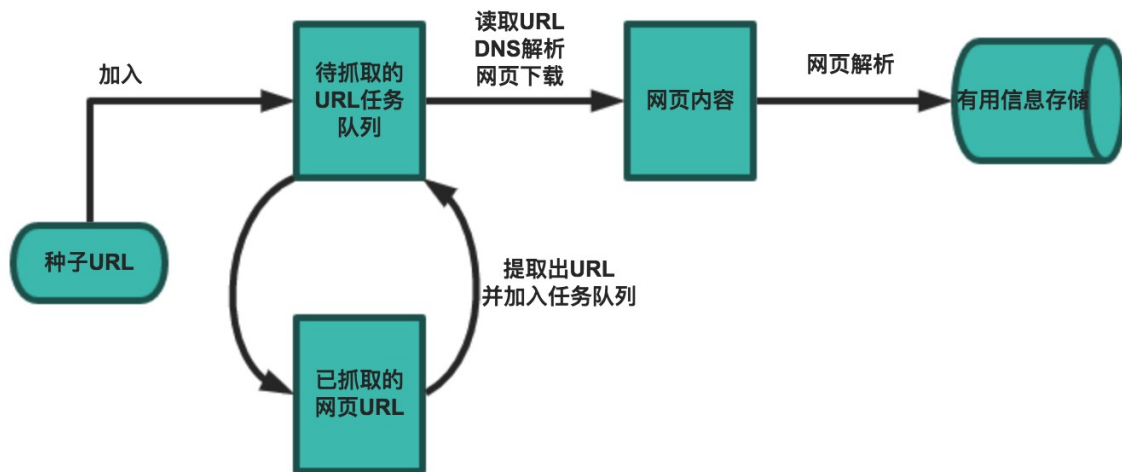
```
>>> import whois
>>> whois.whois('baidu.com')
{'domain_name': ['BAIDU.COM', 'baidu.com'], 'registrar': 'MarkMonitor, Inc.', 'whois_server': 'whois.markmonitor.com', 'referral_url': None, 'updated_date': [datetime.datetime(2017, 7, 28, 2, 36, 28), datetime.datetime(2017, 7, 27, 19, 36, 28)], 'creation_date': [datetime.datetime(1999, 10, 11, 5, 17), datetime.datetime(1999, 10, 11, 4, 5, 17)], 'expiration_date': [datetime.datetime(2026, 10, 11, 5, 17), datetime.datetime(2026, 10, 11, 0, 0)], 'name_servers': ['DNS.BAIDU.COM', 'NS2.BAIDU.COM', 'NS3.BAIDU.COM', 'NS4.BAIDU.COM', 'NS7.BAIDU.COM', 'dns.baidu.com', 'ns4.baidu.com', 'ns3.baidu.com', 'ns7.baidu.com', 'ns2.baidu.com'], 'status': ['clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited', 'clientTransferProhibited https://icann.org/epp#clientTransferProhibited', 'clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited', 'serverDeleteProhibited https://icann.org/epp#serverDeleteProhibited', 'serverTransferProhibited https://icann.org/epp#serverTransferProhibited', 'serverUpdateProhibited https://icann.org/epp#serverUpdateProhibited', 'clientUpdateProhibited (https://www.icann.org/epp#clientUpdateProhibited)', 'clientTransferProhibited (https://www.icann.org/epp#clientTransferProhibited)', 'clientDeleteProhibited (https://www.icann.org/epp#clientDeleteProhibited)', 'serverUpdateProhibited (https://www.icann.org/epp#serverUpdateProhibited)', 'serverTransferProhibited (https://www.icann.org/epp#serverTransferProhibited)', 'serverDeleteProhibited (https://www.icann.org/epp#serverDeleteProhibited)'], 'emails': ['abusecomplaints@markmonitor.com', 'whoisrelay@markmonitor.com'], 'dnssec': 'unsigned', 'name': None, 'org': 'Beijing Baidu Netcom Science Technology Co., Ltd.', 'address': None, 'city': None, 'state': 'Beijing', 'zipcode': None, 'country': 'CN'}
```

6. robotparser: 解析robots.txt的工具。

```
>>> from urllib import robotparser
>>> parser = robotparser.RobotFileParser()
>>> parser.set_url('https://www.taobao.com/robots.txt')
>>> parser.read()
>>> parser.can_fetch('Baiduspider', 'http://www.taobao.com/article')
True
>>> parser.can_fetch('Baiduspider', 'http://www.taobao.com/product')
False
```

一个简单的爬虫

一个基本的爬虫通常分为数据采集（网页下载）、数据处理（网页解析）和数据存储（将有用的信息持久化）三个部分的内容，当然更为高级的爬虫在数据采集和处理时会使用并发编程或分布式技术，这就需要有调度器（安排线程或进程执行对应的任务）、后台管理程序（监控爬虫的工作状态以及检查数据抓取的结果）等的参与。



一般来说，爬虫的工作流程包括以下几个步骤：

1. 设定抓取目标（种子页面/起始页面）并获取网页。
2. 当服务器无法访问时，按照指定的重试次数尝试重新下载页面。
3. 在需要的时候设置用户代理或隐藏真实IP，否则可能无法访问页面。
4. 对获取的页面进行必要的解码操作然后抓取出需要的信息。
5. 在获取的页面中通过某种方式（如正则表达式）抽取出页面中的链接信息。
6. 对链接进行进一步的处理（获取页面并重复上面的动作）。
7. 将有用的信息进行持久化以备后续的处理。

下面的例子给出了一个从“搜狐体育”上获取NBA新闻标题和链接的爬虫。

```
from urllib.error import URLError
from urllib.request import urlopen

import re
import pymysql
import ssl

from pymysql import Error

def decode_page(page_bytes, charsets=('utf-8',)):
    """通过指定的字符集对页面进行解码(不是每个网站都将字符集设置为utf-8)"""
    page_html = None
    for charset in charsets:
        try:
            page_html = page_bytes.decode(charset)
            break
        except UnicodeDecodeError:
            pass
        # logging.error('Decode:', error)
    return page_html

def get_page_html(seed_url, *, retry_times=3, charsets=('utf-8',)):
    """获取页面的HTML代码(通过递归实现指定次数的重试操作)"""
    page_html = None
    try:
        page_html = decode_page(urlopen(seed_url).read(), charsets)
    except URLError:
        # logging.error('URL:', error)
```

```

        if retry_times > 0:
            return get_page_html(seed_url, retry_times=retry_times - 1,
                                charsets=charsets)

    return page_html

def get_matched_parts(page_html, pattern_str, pattern_ignore_case=re.I):
    """从页面中提取需要的部分(通常是链接也可以通过正则表达式进行指定)"""
    pattern_regex = re.compile(pattern_str, pattern_ignore_case)
    return pattern_regex.findall(page_html) if page_html else []

def start_crawl(seed_url, match_pattern, *, max_depth=-1):
    """开始执行爬虫程序并对指定的数据进行持久化操作"""
    conn = pymysql.connect(host='localhost', port=3306,
                           database='crawler', user='root',
                           password='123456', charset='utf8')

    try:
        with conn.cursor() as cursor:
            url_list = [seed_url]
            # 通过下面的字典避免重复抓取并控制抓取深度
            visited_url_list = {seed_url: 0}
            while url_list:
                current_url = url_list.pop(0)
                depth = visited_url_list[current_url]
                if depth != max_depth:
                    # 尝试用utf-8/gbk/gb2312三种字符集进行页面解码
                    page_html = get_page_html(current_url, charsets=('utf-8',
                                                                    'gbk', 'gb2312'))

                    links_list = get_matched_parts(page_html, match_pattern)
                    param_list = []
                    for link in links_list:
                        if link not in visited_url_list:
                            visited_url_list[link] = depth + 1
                            page_html = get_page_html(link, charsets=('utf-8',
                                                                    'gbk', 'gb2312'))

                            headings = get_matched_parts(page_html, r'<h1>(.*?)
<span>')

                            if headings:
                                param_list.append((headings[0], link))
                            cursor.executemany('insert into tb_result values (default,
%s, %s)',
                                              param_list)

                    conn.commit()

    except Error:
        pass
    # logging.error('SQL:', error)
    finally:
        conn.close()

def main():
    """主函数"""
    ssl._create_default_https_context = ssl._create_unverified_context
    start_crawl('http://sports.sohu.com/nba_a.shtml',
                r'<a[^\>]+test=a\s[^\>]*href=["\'](.*)["\']',
                max_depth=2)

```



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

由于使用了MySQL实现持久化操作，所以要先启动MySQL服务器并创建名为 `crawler` 的数据库和名为 `tb_result` 的二维表才能运行该程序。

爬虫注意事项

通过上面的例子，我们对爬虫已经有了一个感性的认识，在编写爬虫时有以下一些注意事项：

1. 处理相对链接。有的时候我们从页面中获取的链接不是一个完整的绝对链接而是一个相对链接，这种情况下需要将其与URL前缀进行拼接（`urllib.parse` 中的 `urljoin()` 函数可以完成此项操作）。
2. 设置代理服务。有些网站会限制访问的区域（例如美国的Netflix屏蔽了很多国家的访问），有些爬虫需要隐藏自己的身份，在这种情况下可以设置使用代理服务器，代理服务器有免费的服务器和付费的商业服务器，但后者稳定性和可用性都更好，强烈建议在商业项目中使用付费的代理服务器。可以通过修改 `urllib.request` 中的 `ProxyHandler` 来为请求设置代理服务器。
3. 限制下载速度。如果我们的爬虫获取网页的速度过快，可能会面临被封禁或者产生“损害动产”的风险（这个可能会导致吃官司且败诉），可以在两次下载之间添加延时从而对爬虫进行限速。
4. 避免爬虫陷阱。有些网站会动态生成页面内容，这会导致产生无限多的页面（例如在线万年历通常会有无穷无尽的链接）。可以通过记录到达当前页面经过了多少个链接（链接深度）来解决该问题，当达到事先设定的最大深度时爬虫就不再像队列中添加该网页中的链接了。
5. SSL相关问题。在使用 `urlopen` 打开一个HTTPS链接时会验证一次SSL证书，如果不做出处理会产生错误提示“SSL: CERTIFICATE_VERIFY_FAILED”，可以通过以下两种方式加以解决：
 - 使用未经验证的上下文

```
import ssl  
  
request = urllib.request.Request(url='...', headers={...})  
context = ssl._create_unverified_context()  
web_page = urllib.request.urlopen(request, context=context)
```

- 设置全局性取消证书验证

```
import ssl  
  
ssl._create_default_https_context = ssl._create_unverified_context
```