什么是网络爬虫?

网络爬虫（又被称为网页蜘蛛，网络机器人）就是模拟客户端（主要指浏览器）发送网络请求，接收请求响应，一种按照一定的规则，自动地抓取互联网信息的程序。

原则上，浏览器可以实现的功能，爬虫都可以完成。

在当下的大数据时代，数据量是巨大的，各个领域每天都在源源不断产生数据，能够获取数据并对数据进行分析，就可以产生财富。

比如，推荐系统，电商平台根据用户的浏览商品或购买情况，就会自动识别出用户的偏好，在该用户主页推荐的商品就更容易成交，可以大幅度提升购买力。

再比如自媒体平台，创作者生产数据（即发布文章、视频），消费者即企业、组织付费做广告推广，每一个作品经过算法推荐给合适人群，提高平台用户的体验以及提升点击率，增加广告的曝光度。

然而，巧妇难为无米之炊，大数据的基础是数据获取，然后再对原有数据进行清洗、统计，数据量如此大，那么我们如何高效获取这些数据呢?

首先，要清楚目标数据来源，包括用户产生的数据，比如外卖点餐平台，用户订外卖就在产生数据；还有政府统计的数据，GDP、失业率等等；还有专门的数据管理公司搜集数据盈利；还有自己用爬虫或其他手段搜集的数据。

2023年1月份70个大中城市商品住宅销售价格变动情况

来源：国家统计局　　发布时间：2023-02-16　09:30　　关闭窗口 打印本页

表1：2023年1月70个大中城市新建商品住宅销售价格指数

城市	环比 上月=100	同比 上年同月=100	累计 上年同期=100	城市	环比 上月=100	同比 上年同月=100	累计 上年同期=100
北　京	100.4	105.2	105.2	唐　山	100.1	97.4	97.4
天　津	100.3	97.0	97.0	秦皇岛	99.7	94.4	94.4
石家庄	100.1	97.6	97.6	包　头	99.9	95.5	95.5
太　原	100.2	95.6	95.6	丹　东	100.1	96.0	96.0
呼和浩特	99.7	96.7	96.7	锦　州	99.5	96.2	96.2
沈　阳	99.6	95.0	95.0	吉　林	99.2	95.4	95.4
大　连	99.9	95.2	95.2	牡丹江	99.8	97.0	97.0
长　春	99.5	95.0	95.0	无　锡	99.7	99.1	99.1
哈尔滨	99.6	93.0	93.0	徐　州	100.2	98.4	98.4
上　海	100.7	104.2	104.2	扬　州	100.4	97.6	97.6
南　京	100.0	100.2	100.2	温　州	99.0	92.1	92.1
杭　州	100.4	106.5	106.5	金　华	99.8	97.6	97.6
宁　波	100.6	101.9	101.9	蚌　埠	100.4	97.7	97.7
合　肥	100.7	102.4	102.4	安　庆	100.2	95.7	95.7
福　州	100.2	97.5	97.5	泉　州	99.5	96.3	96.3
厦　门	99.7	96.1	96.1	九　江	100.2	98.3	98.3
南　昌	100.3	101.7	101.7	赣　州	99.9	99.7	99.7
济　南	99.8	101.6	101.6	烟　台	99.9	98.2	98.2
青　岛	100.4	101.5	101.5	济　宁	99.8	95.6	95.6
郑　州	100.2	97.0	97.0	洛　阳	99.5	95.0	95.0
武　汉	100.2	94.3	94.3	平顶山	99.4	97.0	97.0
长　沙	100.3	103.4	103.4	宜　昌	99.5	94.8	94.8
广　州	99.8	99.7	99.7	襄　阳	99.7	96.0	96.0
深　圳	99.8	99.1	99.1	岳　阳	100.1	91.9	91.9
南　宁	99.9	96.0	96.0	常　德	99.5	94.4	94.4
海　口	99.9	100.9	100.9	韶　关	100.2	97.4	97.4

本节我们主要考虑网络爬虫获取数据。如上图所示是国家统计局统计2023年1月份70个大中城市商品住宅销售价格变动情况，假设我们只想获取城市以及该城市同比上一年同月的价格变动情况，只需要获取两列标红数据就可以。

当然我们可以手动将需要的数据复制下来，粘贴到本地excel表格当中。然而，如果数据量特别大，手动操作耗时久，而且也容易出错。所以，网络爬虫可以分析网页结构，按照此一定规律解析出目标数据，然后利用循环语句批量处理。

鼠标右键，选择检查，可以发现网页显示的数据和源码中的数据，然后分析网页源码结构，批量爬取，存储到excel或数据库中。

网络爬虫合法性问题

爬虫工程师是个热门职业，薪水也比较高，但是，也是一个高风险行业，媒体上经常会看到某某工程师由于写了一段代码，**进了。**

爬虫的风险体现：爬虫干扰了被访问网站的正常运营，把网站服务器搞瘫痪，或者爬虫获取到法律保护的特定数据信息。

那么如何规避风险？首先要严格遵守网站设置的robots协议，一般在网站后面加上robots.txt就可以看到网站允许爬取哪些数据和禁止爬取哪些数据。


```
User-agent: *
Disallow: /subject_search
Disallow: /amazon_search
Disallow: /search
Disallow: /group/search
Disallow: /event/search
Disallow: /celebrities/search
Disallow: /location/drama/search
Disallow: /forum/
Disallow: /new_subject
Disallow: /service/iframe
Disallow: /j/
Disallow: /link2/
Disallow: /recommend/
Disallow: /doubanapp/card
Disallow: /update/topic/
Disallow: /share/
Disallow: /people/*/collect
Disallow: /people/*/wish
Disallow: /people/*/all
Disallow: /people/*/do
Allow: /ads.txt
Sitemap: https://www.douban.com/sitemap_index.xml
Sitemap: https://www.douban.com/sitemap_updated_index.xml
# Crawl-delay: 5

User-agent: Wandoujia Spider
Disallow: /

User-agent: Mediapartners-Google
Disallow: /subject_search
Disallow: /amazon_search
Disallow: /search
Disallow: /group/search
Disallow: /event/search
Disallow: /celebrities/search
Disallow: /location/drama/search
Disallow: /j/
```

但是robots协议是一个防君子不防小人的协议，对于真正想要窃取信息的人来说，形同虚设。**但是，我们自己在使用爬虫技术时，一旦爬取私密信息要立即删除，不要传播甚至售卖数据。**

爬虫分类

通用网络爬虫：百度、Google等搜索引擎，从一些初始URL扩展到整个网站，主要为门户站点搜索引擎和大型网站服务采集数据。

聚焦网络爬虫：又称主题网络爬虫，选择性地爬行根据需求的主题相关页面的网络爬虫。

增量式网络爬虫：对已下载网页采取增量式更新知识和只爬行新产生或者已经发生变化的网页爬虫。

urllib爬虫模块

urllib是Python自带的标准库中用于网络请求的库，无需安装，可以直接使用。

urllib.request库：模拟浏览器发起一个HTTP请求，并获取请求响应结果。

urllib.request.urlopen的语法格式：

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

参数说明：

url：url参数是str类型的地址，也就是要访问的URL,例如:https://www.baidu.com

data：默认值为None,urllib判断参数data是否为None从而区分请求方式。如果参数data为None，则代表请求方式为Get，反之请求方式为Post，发送Post请求。参数data以字典形式存储数据，并将参数data由字典类型转换成字节类型才能完成Post请求。

urlopen函数返回的结果是一个http.client.HTTPResponse对象。

案例：请求百度首页

```
import urllib.request

request = urllib.request.Request("http://www.baidu.com")
response = urllib.request.urlopen(request)

html = response.read(50).decode('utf-8')

print("type(html) =", type(html))
print("html =", html)

# 响应状态码 200代表OK
```

```
print("response.getcode() =", response.getcode())
print("response.geturl() =", response.geturl())
print("response.info() =", response.info())
```

设置User-Agent

User-Agent也叫用户代理，是请求载体的身份标识，也就是浏览器的名字。

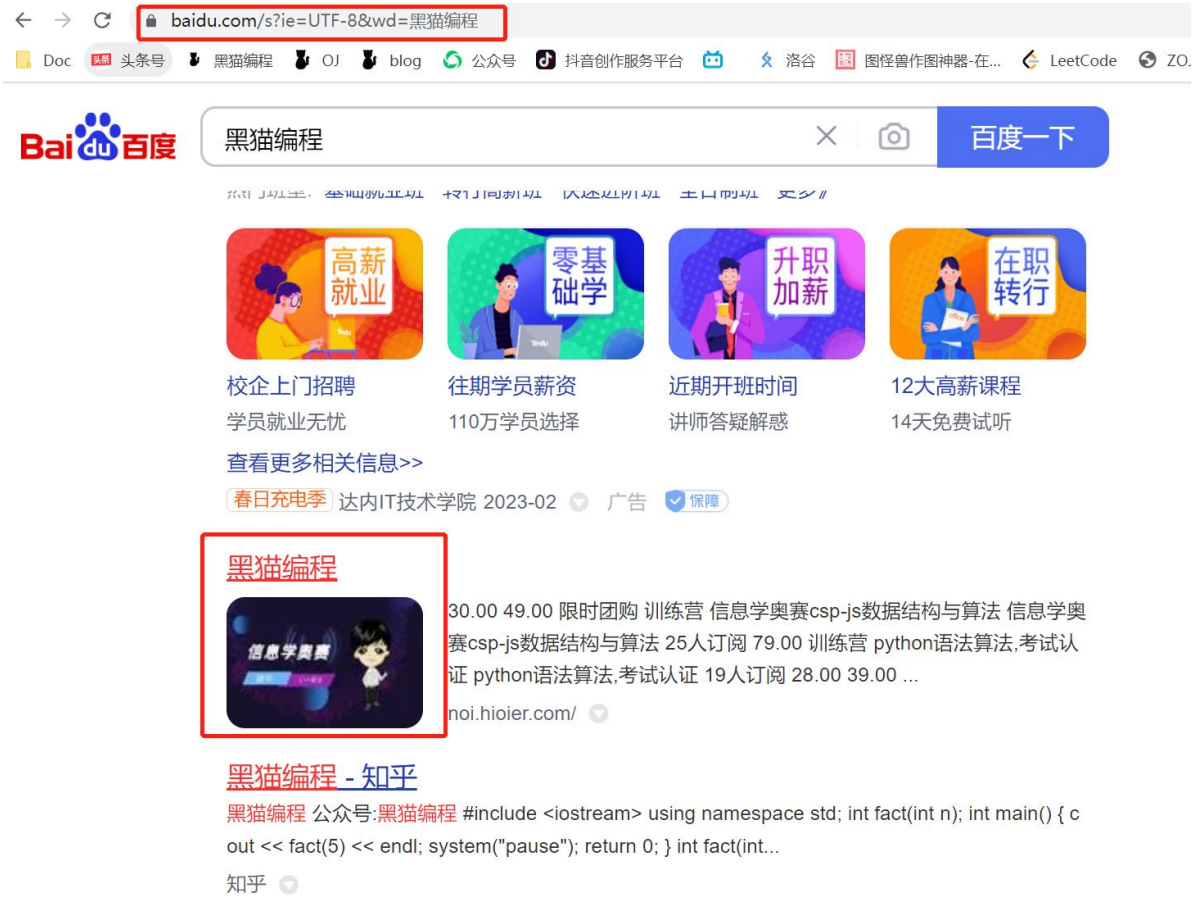
在网络爬虫技术领域，没有技术上限和固定不变的方法，因为除了常规的爬取手段，网站管理员也会出于安全因素考虑设置反爬策略。所以，爬虫工程师就是不断和网络管理员斗智斗勇的过程，学无止境。

对请求主体限制是简单常用的手段，可以过滤掉很多无效爬虫请求。**如果爬虫程序不设置代理，默认代理会显示Python。**

```
import urllib.request

request = urllib.request.Request("http://www.baidu.com")
response = urllib.request.urlopen(request)

html = response.read(50).decode('utf-8')

print(request.get_header("User-agent"))    # Python-urllib/3.6
```

编码解码

在百度浏览器搜索框输入中文内容“黑猫编程”：



url末尾是“wd=黑猫编程”，但是复制到本地发现是乱码：

```
https://www.baidu.com/s?ie=UTF-8&wd=%E9%BB%91%E7%8C%AB%E7%BC%96%E7%A8%8B
```

这是由于url不支持中文，网络传输过程需要进行编码处理：

```
import urllib.parse
kw = {'wd': '黑猫编程'}
# 编码
ret = urllib.parse.urlencode(kw)
print(ret)

# 解码
ret = urllib.parse.unquote(ret)
print(ret)
```

```
wd=%E9%BB%91%E7%8C%AB%E7%BC%96%E7%A8%8B
wd=黑猫编程
```

```
import urllib.request
import random

rawURL = "http://www.baidu.com"

uaList = [
          "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv2.0.1) Gecko/20100101 Firefox/4.0.1",
          "Mozilla/5.0 (Windows NT 6.1; rv2.0.1) Gecko/20100101 Firefox/4.0.1",
          "Opera/9.80 (Macintosh; Intel Mac OS X 10.6.8; U; en) Presto/2.8.131 Version/11.11",
          "Opera/9.80 (Windows NT 6.1; U; en) Presto/2.8.131 Version/11.11",
          "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_0) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11"
          ]

userAgent = random.choice(uaList)
request = urllib.request.Request(rawURL)

#第一个字母大写，其他都小写
request.add_header("User-Agent", userAgent)
print(request.get_header("User-agent"), '\n')

keyword = input("请输入要查询都关键字：  ")
word = {"wd": keyword}
word = urllib.parse.urlencode(word)

fullURL = rawURL + "?" + word
headers = {'User-Agent': userAgent}
request = urllib.request.Request(fullURL, headers=headers)
response = urllib.request.urlopen(request)
html = response.read(100).decode()

```

```
print(request.get_header("User-agent"), '\n')
print(html)
```