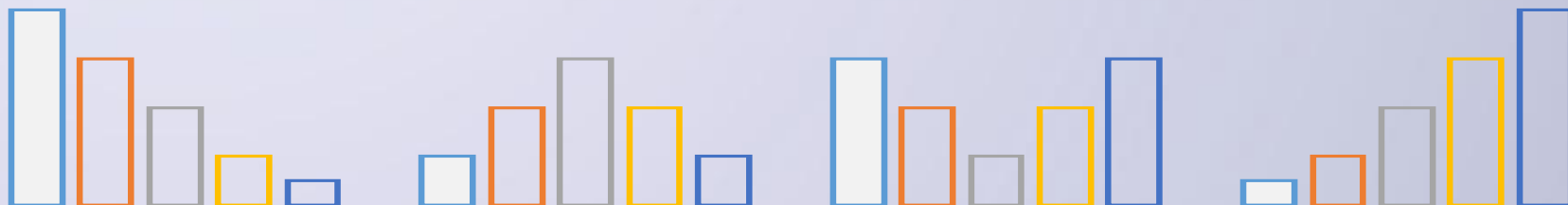
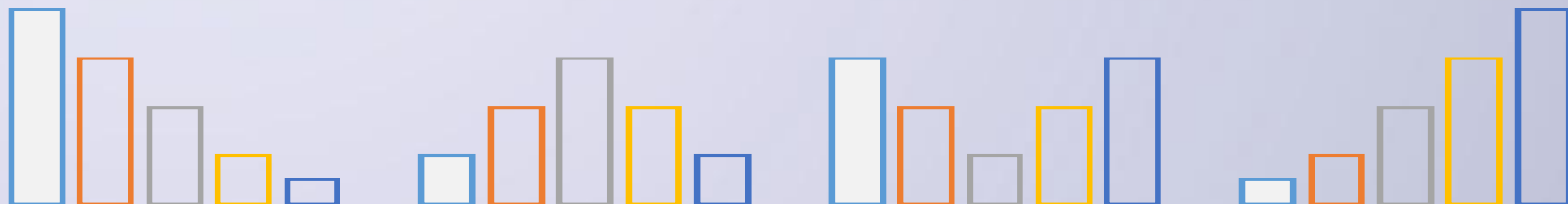


# Python语言程序设计

北京理工大学 嵩天



# 第12章 网络爬虫和自动化






计算机将按照你给出的指令去执行，这是好事，但也是坏事。

*The good news about computers is that they do what you tell them to do. The bad news is that they do what you tell them to do.*

——泰德.尼尔森(Ted Nelson)

信息技术先驱、超文本传输协议(HTTP)的设计者、哲学家、社会学家



# 问题概述



## 要点：

- Python 语言实现网络爬虫的问题引入。



# Python与网页处理

- Python 语言发展中有一个里程碑式的应用事件，即美国谷歌（GOOGLE）公司在搜索引擎后端采用 Python 语言进行链接处理和开发，这是该语言发展成熟的重要标志。Python 语言的简洁性和脚本特点非常适合链接和网页处理



# Python与网页处理

- 万维网（WWW）的快速发展带来了大量获取和提交网络信息的需求，这产生了“网络爬虫”等一系列应用。
- Python 语言提供了很多类似的函数库，包括urllib、urllib2、urllib3、wget、scrapy、requests 等。这些库作用不同、使用方式不同、用户体验不同。



# Python与网页爬虫

- 对于爬取回来的网页内容，可以通过re（正则表达式）、beautifulsoup4等函数库来处理，随着该领域各函数库的发展，本章将详细介绍其中最重要且最主流的两个函数库：requests 和beautifulsoup4，它们都是第三方库。





# Python与网页爬虫

- 网络爬虫应用一般分为两个步骤：（1）通过网络连接获取网页内容（2）对获得的网页内容进行处理。
- 这两个步骤分别使用不同的函数库：requests 和 beautifulsoup4



# 安装requests 库

- 采用pip 指令安装requests 库，如果在Python 2 和Python 3 并存的系统中，采用pip3 指令

**:\>pip install requests # 或者 pip3 install  
requests**



# 安装requests 库

- 采用pip 或pip3 指令安装beautifulsoup4 库，注意，不要安装beautifulsoup 库，后者由于年久失修，已经不再维护了

**: \> pip install beautifulsoup4**

**# 或者 pip3 install beautifulsoup4**



# 网页爬虫

- 使用Python 语言实现网络爬虫和信息提交是非常简单的事情，代码行数很少，也无须知道网络通信等方面知识，非常适合非专业读者使用。然而，肆意的爬取网络数据并不是文明现象，通过程序自动提交内容争取竞争性资源也不公平。就像那些肆意的推销电话一样，他们无视接听者意愿，不仅令人讨厌也有可能引发法律纠纷。



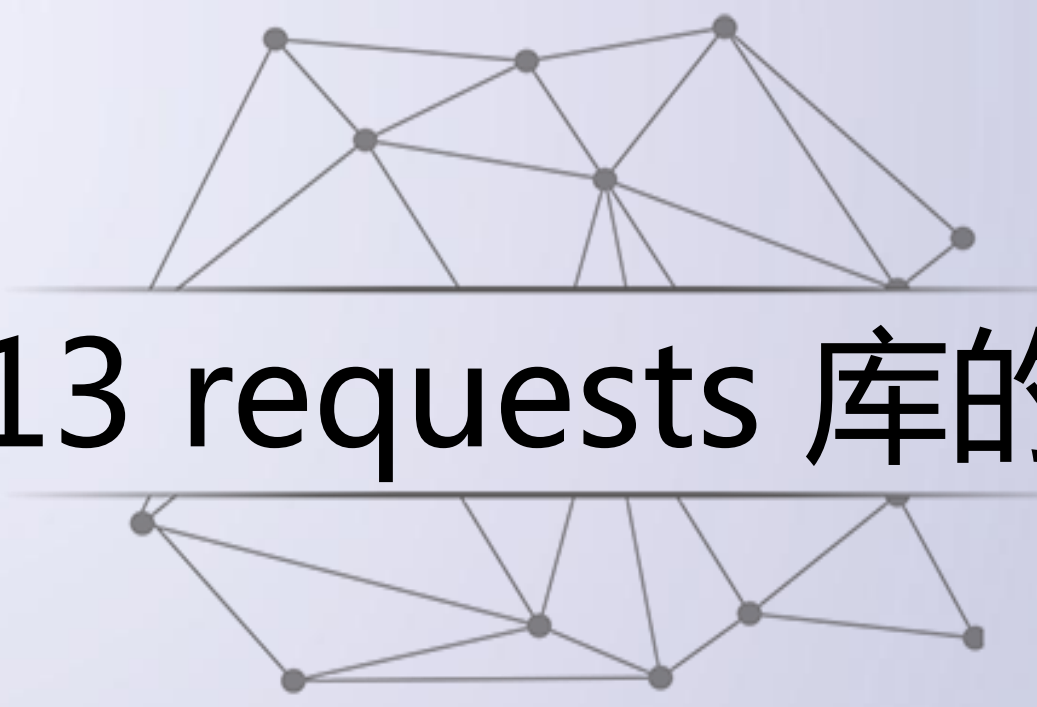
# 拓展：Robots 排除协议

- Robots 排除协议（Robots Exclusion Protocol），也被称为爬虫协议，它是网站管理者表达是否希望爬虫自动获取网络信息意愿的方法。管理者可以在网站根目录放置一个robots.txt 文件，并在文件中列出哪些链接不允许爬虫爬取。一般搜索引擎的爬虫会首先捕获这个文件，并根据文件要求爬取网站内容。Robots 排除协议重点约定不希望爬虫获取的内容，如果没有该文件则表示网站内容可以被爬虫获得，然而，Robots 协议不是命令和强制手段，只是国际互联网的一种通用道德规范。绝大部分成熟的搜索引擎爬虫都会遵循这个协议，建议个人也能按照互联网规范要求合理使用爬虫技术。



# 思考与练习：

- [E12.1]请思考网络爬虫的可能应用？



# 模块13 requests 库的使用



- requests 库是一个简洁且简单的处理HTTP请求的第三方库。





# requests 库概述

- requests的最大优点是程序编写过程更接近正常URL 访问过程。



# requests 库概述

- 这个库建立在Python 语言的urllib3 库基础上，类似这种在其他函数库之上再封装功能提供更友好函数的方式在Python 语言中十分常见。在Python 的生态圈里，任何人都有通过技术创新或体验创新发表意见和展示才华的机会。



# requests 库概述

- request 库支持非常丰富的链接访问功能，包括：国际域名和 URL 获取、HTTP 长连接和连接缓存、HTTP 会话和Cookie 保持、浏览器使用风格的SSL 验证、基本的摘要认证、有效的键值对Cookie 记录、自动解压缩、自动内容解码、文件分块上传、HTTP(S)代理功能、连接超时处理、流数据下载等。有关 requests 库的更多介绍请访问：
  - <http://docs.python-requests.org>



# requests 库中的网页请求函数

函数	描述
<code>get(url [, timeout=n])</code>	对应于 HTTP 的 GET 方式，获取网页最常用的方法，可以增加 <code>timeout=n</code> 参数，设定每次请求超时时间为 n 秒
<code>post(url, data = {'key': 'value'})</code>	对应于 HTTP 的 POST 方式，其中字典用于传递客户数据
<code>delete(url)</code>	对应于 HTTP 的 DELETE 方式
<code>head(url)</code>	对应于 HTTP 的 HEAD 方式
<code>options(url)</code>	对应于 HTTP 的 OPTIONS 方式
<code>put(url, data = {'key': 'value'})</code>	对应于 HTTP 的 PUT 方式，其中字典用于传递客户数据



# requests 库中的网页请求函数

- `get()` 是获取网页最常用的方式，在调用 `requests.get()` 函数后，返回的网页内容会保存为一个 `Response` 对象，其中，`get()` 函数的参数 `url` 必须链接采用 `HTTP` 或 `HTTPS` 方式访问



# 网页请求函数

```
>>>import requests
```

```
>>>r=requests.get("http://www.baidu.com") #使用 get 方法打开百度链接
```

```
>>>type(r)
```

```
<class 'requests.models.Response'>      #返回 Response 对象
```



# 网页请求函数

- 和浏览器的交互过程一样，`requests.get()`代表请求过程，它返回的`Response` 对象代表响应。返回内容作为一个对象更便于操作，`Response` 对象的属性如下表所示，需要采用`<a>.<b>`形式使用。



# Response 对象的属性

属性	描述
status_code	HTTP 请求的返回状态，整数，200 表示连接成功，404 表示失败
text	HTTP 响应内容的字符串形式，即，也是 url 对应的页面内容
encoding	HTTP 响应内容的编码方式
content	HTTP 响应内容的二进制形式





# Response 对象的属性

- `status_code` 属性返回请求HTTP 后的状态，在处理数据之前要先判断状态情况，如果请求未被响应，需要终止内容处理。
- `text` 属性是请求的页面内容，以字符串形式展示。



# Response 对象的属性

- encoding 属性非常重要，它给出了返回页面内容的编码方式，可以通过对encoding 属性赋值更改编码方式，以便于处理中文字符
- content 属性是页面内容的二进制形式



# Response 对象的属性

```
>>>r = requests.get("http://www.baidu.com")

>>>r.status_code      #返回状态
200

>>>r.text              #观察返回的内容，中文字符是否能正常显示
(输出略)

>>>r.encoding          #默认的编码方式是 ISO-8859-1，所以中文是乱码
'ISO-8859-1'

>>> r.encoding = 'utf-8'    # 更改编码方式为 utf-8

>>> r.text              # 更改完成，返回内容中的中文字符可以正常显示了
(输出略)
```



# Response 对象的方法

方法	描述
<code>json()</code>	如果 HTTP 响应内容包含 JSON 格式数据，该方法解析 JSON 数据
<code>raise_for_status()</code>	如果不是 200，那么这个方法就会产生异常



# Response 对象的方法

- `json()`方法能够在HTTP 响应内容中解析存在的JSON 数据，这将带来解析HTTP的便利。



# Response 对象的方法

- `raise_for_status()`方法能在非成功响应后产生异常，即只要返回的请求状态`status_code` 不是200，这个方法会产生一个异常，用于`try...except` 语句。使用异常处理语句可以避免设置一堆复杂的`if` 语句，只需要在收到响应调用这个方法，就可以避开状态字200 以外的各种意外情况。



# Response 对象的方法

- requests 会产生几种常用异常。当遇到网络问题时，如：  
DNS 查询失败、拒绝连接等，requests 会抛出 `ConnectionError` 异常；遇到无效HTTP 响应时，requests 则会抛出 `HTTPError` 异常；若请求url 超时，则抛出 `Timeout` 异常；若请求超过了设定的最大重定向次数，则会抛出一个 `TooManyRedirects` 异常



# 获取一个网页内容

```
1  import requests
2  def getHTMLText():
3      try:
4          r = requests.get(url, timeout=30)
5          r.raise_for_status() #如果状态不是 200, 引发异常
6          r.encoding = 'utf-8' #无论原来用什么编码, 都改成 utf-8
7          return r.text
8      except:
9          return ""
10 url = "http://www.baidu.com"
11 print(getHTMLText(url))
```





# 拓展：HTTP 的GET 和POST

- HTTP 协议定义了客户端与服务器交互的不同方法，最基本的方法是GET 和POST。顾名思义，GET 可以根据某链接获得内容，POST 用于发送内容。然而，GET 也可以向链接提交内容



# 拓展：HTTP 的GET 和POST

- 1) GET 方式可以通过URL 提交数据，待提交数据是URL 的一部分；采用POST 方式，待提交数据放置在HTML HEADER 内；



# 拓展：HTTP 的GET 和POST

- 2 ) GET 方式提交的数据最多不超过1024 字节，  
POST 没有对提交内容的长度限制。
- 3 ) 安全性问题。如（ 1 ）所述，使用GET 时参数会  
显示在URL 中，而POST不会。所以，如果这些数据  
是非敏感数据，那么使用GET；如果提交数据是敏感  
数据，建议采用POST 方式。



## 思考与练习：

- [E12.2]查阅资料更多了解HTTP协议中post和get功能的区别和联系。
- [E12.3]requests 库提供了一个post()函数，查阅资料了解该函数的用法。

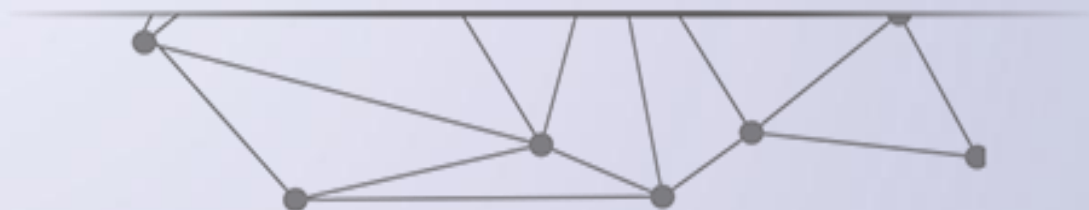


## 思考与练习：

- [E12.4]用get()访问百度页面，用len()函数分别计算text 属性和content 属性所返回网页内容的长度，思考长度差异产生的原因。



# 模块14 beautifulsoup4库的使用





- beautifulsoup4 库是一个解析和处理HTML 和 XML 的第三方库。



# beautifulsoup4 库概述

- 使用requests 库获取HTML 页面并将其转换成字符串后，需要进一步解析HTML页面格式，提取有用信息，这需要处理HTML 和XML 的函数库。
- beautifulsoup4 库，也称为Beautiful Soup 库或bs4 库，用于解析和处理HTML和XML。





# beautifulsoup4 库概述

- 需要注意，它不是BeautifulSoup 库。它的最大优点是能根据HTML 和XML 语法建立解析树，进而高效解析其中的内容。



# beautifulsoup4 库概述

- HTML 建立的Web 页面一般非常复杂，除了有用的内容信息外，还包括大量用于页面格式的元素，直接解析一个Web 网页需要深入了解HTML 语法，而且比较复杂。beautifulsoup4 库将专业的Web 页面格式解析部分封装成函数，提供了若干有用且便捷的处理函数。



# beautifulsoup4 库概述

- beautifulsoup4 库采用面向对象思想实现，简单说，它把每个页面当做一个对象，通过<a>.<b>的方式调用对象的属性（即包含的内容），或者通过<a>.<b>()的方式调用方法（即处理函数）。



# beautifulsoup4 库概述

- 在使用beautifulsoup4 库之前，需要进行引用，由于这个库的名字非常特殊且采用面向对象方式组织，可以用 from...import 方式从库中直接引用 BeautifulSoup 类，方法如下。

```
>>>from bs4 import BeautifulSoup
```



# beautifulsoup4 库概述

- 有关beautifulsoup4 库的更多介绍请参考这个第三方库主页：  
<http://www.crummy.com/software/BeautifulSoup/bs4/>



# beautifulsoup4 库解析

- beautifulsoup4 库中最主要的是BeautifulSoup 类，每个实例化的对象相当于一个页面。采用 `from...import` 导入库中类后，使用 `BeautifulSoup()` 创建一个BeautifulSoup对象。



# 创建一个BeautifulSoup对象

```
>>>import requests
>>>from bs4 import BeautifulSoup
>>>r = requests.get("http://www.baidu.com")
>>>r.encoding = "utf-8"    #为了简化代码，没有考虑异常情况
>>>soup = BeautifulSoup(r.text)    #soup 就是一个 BeautifulSoup 对象
>>> type(soup)
<class 'bs4.BeautifulSoup'>
```



# 创建一个BeautifulSoup对象

- 创建的BeautifulSoup 对象是一个树形结构，它包含HTML 页面里的每一个Tag（标签）元素，如<head>、<body>等。具体来说，HTML 中的主要结构都变成了BeautifulSoup 对象的一个属性，可以直接用<a>.<b>形式获得，其中<b>的名字采用HTML 中标签的名字。





# BeautifulSoup 对象的常用属性

属性	描述
head	HTML 页面的<head>内容
title	HTML 页面标题，在<head>之中，由<title>标记
body	HTML 页面的<body>内容
p	HTML 页面中第一个<p>内容
strings	HTML 页面所有呈现在 Web 上的字符串，即标签的内容
stripped_strings	HTML 页面所有呈现在 Web 上的非空格字符串



# BeautifulSoup 对象的常用属性

```
>>>soup.head    #略去<style>标签输出
<head><meta          content="text/html; charset=utf-8"          http-
equiv="content-  type"><meta content="IE=Edge"  http-equiv="X-UA-
Compatible"><title>百度一下, 你就知道</title></meta></meta></head>
>>>title=soup.title
<title>百度一下, 你就知道</title>
>>>type(title)    #每个对应 HTML Tag 的属性是一个 Tag 类型
<class 'bs4.element.Tag'>
>>>soup.p
<p id="lh"><a href="http://www.baidu.com/cache/sethelp/help.html"
target="_blank">把百度设为主页</a><a href="http://home.baidu.com">关
于百度</a><a href="http://ir.baidu.com">About Baidu</a></p>
```



# BeautifulSoup 对象的常用属性

- 每一个Tag 标签在beautifulsoup4 库中也是一个对象，称为Tag 对象。上例中，title 是一个标签对象。每个标签对象在HTML 中都有类似的结构：

`<a class="mnav" href="http://www.nuomi.com">糯米</a>`




# BeautifulSoup 对象的常用属性

- 其中，尖括号 ( `<>` ) 中的标签的名字是 `name`，尖括号内其他项是 `attrs`，尖括号之间的内容是 `string`。  
因此，可以通过 `Tag` 对象的 `name`、`attrs` 和 `string` 属性获得相应内容，采用 `<a>.<b>` 的语法形式。
- 标签 `Tag` 有 4 个常用属性



# 标签对象的常用属性

属性	描述
name	字符串，标签的名字，比如 div
attrs	字典，包含了原来页面 Tag 所有的属性，比如 href
contents	列表，这个 Tag 下所有子 Tag 的内容
string	字符串，Tag 所包围的文本，网页中真实的文字



# 标签对象的常用属性

```
>>>soup.a
<a class="mnav" href="http://www.nuomi.com">糯米</a>
>>>soup.a.name
'a'
>>>soup.a.attrs
{'href': 'http://www.nuomi.com', 'class': ['mnav']}
>>>soup.a.string
'糯米'
>>>title.name #title 变量在上段例子中已经定义
'title'
>>>title.string
'百度一下，你就知道'
>>>soup.p.contents
[<a href="http://www.baidu.com/cache/sethelp/help.html" target=
"_blank">把百度设为主页</a>, <a href="http://home.baidu.com">关于百度
</a>, <a href="http://ir.baidu.com">About Baidu</a>]
```



# 标签对象的常用属性

- 由于HTML 语法可以在标签中嵌套其他标签，所以，string 属性的返回值遵循如下原则：
  - 如果标签内部没有其他标签，string 属性返回其中的内容；
  - 如果标签内部有其他标签，但只有一个标签，string 属性返回最里面标签的内容；
  - 如果标签内部有超过1 层嵌套的标签，string 属性返回None（空字符串）。



# 标签对象的常用属性

- HTML 语法中同一个标签会有很多内容，例如<a> 标签，百度首页一共有13 处，直接调用soup.a 只能返回第一个。





# 查找对应标签

- HTML 语法中同一个标签会有很多内容，例如<a> 标签，百度首页一共有13 处，直接调用soup.a 只能返回第一个。

```
<a class="mnav" href="http://www.nuomi.com">糯米</a>
```

```
<a class="mnav" href="http://news.baidu.com">新闻</a>
```

```
<a class="mnav" href="http://www.hao123.com">hao123</a>
```


```
<a class="mnav" href="http://map.baidu.com">地图</a>
```

```
<a class="mnav" href="http://v.baidu.com">视频</a> ....
```



# 查找对应标签

- 当需要列出标签对应的所有内容或者需要找到非第一个标签时，需要用到BeautifulSoup 的find()和find\_all()方法。这两个方法会遍历整个HTML 文档，按照条件返回标签内容。



# 查找对应标签

**`BeautifulSoup.find_all(name, attrs, recursive, string, limit)`**

作用：根据参数找到对应标签，返回列表类型。

参数：

`name`：按照 Tag 标签名字检索，名字用字符串形式表示，例如：`div`，`li`；

`attrs`：按照 Tag 标签属性值检索，需要列出属性名称和值，采用 JSON 表示；

`recursive`：设置查找层次，只查找当前标签下一层时使用 `recursive=False`；

`string`：按照关键字检索 `string` 属性内容，采用 `string=开始`；

`limit`：返回结果的个数，默认返回全部结果。

# 查找对应标签

```
>>>a = soup.find_all('a')      #查找所有的<a>
>>> len(a)
13
>>>soup.find_all('script')
[<script>var      md5="230CFBxBZBXCCCDBYCEDREADTEHDREIDZ"</script>,
<script src="http://www.zgxiangxin.com/jquery/jquery-1.10.4.min.
js"></script>]
>>>soup.find_all('script',{'src':'http://www.zgxiangxin.com/\
jquery/jquery-1.10.4.min.js'}) #筛选，只找 src=字符串的标签
[<script src="http://www.zgxiangxin.com/jquery/jquery-1.10.4
.min.js"></script>]
>>>import re  # 使用正则表达式库，可以用这个库实现字符串片段匹配
>>>soup.find_all('script',{'src':re.compile('jquery')})
[<script src="http://www.zgxiangxin.com/jquery/jquery-1.10.4
.min.js"></script>]
>>>soup.find_all(string=re.compile('百度'))
['百度一下，你就知道', '把百度设为主页', '关于百度', '使用百度前必读']
```



# 查找对应标签

- 简单说，BeautifulSoup 的find\_all()方法可以根据标签名字、标签属性和内容检索并返回标签列表，通过片段字符串检索时需要使用正则表达式re 函数库，re 是Python 标准库，直接通过import re 即可使用。



# 查找对应标签

- 采用`re.compile('jquery')`实现对片段字符串（如 `'jquery'`）的检索。当对标签属性检索时，属性和对应的值采用JSON格式，例如：

**`'src':re.compile('jquery')`**

- 其中，键值对中值的部分可以是字符串或者正则表达式。



# 拓展：正则表达式

- 正则表达式是表达和操作字符串的一种逻辑表达，一般在计算机编译器中使用。Python 语言采用正则表达式辅助字符串查找。正则表达式是一种规则，只要字符串符合这个规则，就算作匹配。例如，通过 `re.compile()` 函数注册一个正则表达式 `'jquery'`，则所有包含表达式的字符串都与它匹配。除了字符串，正则表达式还可以通过 `*+{}` 等符号扩展功能。有兴趣的读者可以查阅资料了解 Python 中正则表达式函数库 `re` 的更多高级使用。



# 查找对应标签

- 除了 `find_all()` 方法，BeautifulSoup 类还提供一个 `find()` 方法，它们的区别只是前者返回全部结果而后者返回找到的第一个结果，`find_all()` 函数由于可能返回更多结果，所以采用列表形式；`find()` 函数返回字符串形式。





# 查找对应标签

**`BeautifulSoup.find(name, attrs, recursive, string)`**

作用：根据参数找到对应标签，采用字符串返回找到的第一个值。

参数：与 `find_all()` 方法一样，略。



# beautifulsoup4库

- 处理网页需要对HTML 有一定的理解，然而实现爬虫就不算复杂，这里仅介绍beautifulsoup4 库中与爬虫相关的一些属性和操作。beautifulsoup4 库是一个非常完备且活跃的HTML 解析函数库，它还可以完成更多复杂操作。深入使用请阅读beautifulsoup4 库官方网站上提供的使用文档。



## 思考与练习：

- [E12.5] 下列哪个不属于HTML 的Tag ？

A title      B a      C class      D head

- [E12.6] 这是一个简单的HTML 页面，请保存为字符串，完成后面的计算要求。



# 思考与练习：

**<html>**

**<head>**

**<title>Simple test</title>**

**</head>**

**<body>**

**<p id="China">中国，<b>你好！</b>。 </p>**

**<p id="World">世界，<b>大同！</b>。 </p>**

**</body>**

**</html>**

- ( 1 ) 打印head 标签的内容；
- ( 2 ) 获取body 标签的内容；
- ( 3 ) 获取id 为China 的标签对象  
；
- ( 4 ) 获取并打印HTML 页面中的  
中文字符。



# 实例24 中国大学排名爬虫



# 要点：

- 这是一个获取中国大学排名的爬虫实例，采用了requests 和beautifulsoup4函数库。



# 中国大学排名爬虫

- 全球有很多份大学排名，这里以上海交通大学研发的“软科中国最好大学排名2016”为例，如图12.1所示，编写“大学排名爬虫”，从网络上获取数据，这个大学排名网址如下。
- <http://www.zuihaodaxue.cn/zuihaodaxuepaiming2016.html>



# 软科中国大学排名2016网页片段

排名	学校名称	省市	总分	指标得分
				生源质量（新生高考成绩得分） ▼
1	清华大学	北京市	95.9	100.0
2	北京大学	北京市	82.6	98.9
3	浙江大学	浙江省	80	88.8
4	上海交通大学	上海市	78.7	90.6
5	复旦大学	上海市	70.9	90.4
6	南京大学	江苏省	66.1	90.7
7	中国科学技术大学	安徽省	65.5	90.1
8	哈尔滨工业大学	黑龙江省	63.5	80.9
9	华中科技大学	湖北省	62.9	83.5
10	中山大学	广东省	62.1	81.8





# 中国大学排名爬虫

- 拟从该网址爬取该名单上310 所国内大学的排名数据，并将它们打印出来。读者可以对这些数据开展其他操作。



# 中国大学排名爬虫

- 大学排名爬虫的构建需要三个重要步骤：
- 第一，从网络上获取网页内容；
- 第二，分析网页内容并提取有用数据到恰当的数据结构中；
- 第三，利用数据结构展示或进一步处理数据。
  - 由于大学排名是一个典型的二维数据，因此，采用二维列表存储该排名所涉及的表单数据。



# 爬取网页内容

- 具体来说，采用requests 库爬取网页内容，使用beautifulsoup4 库分析网页中数据，提取310 个学校的排名及相关数据，存储到二维列表中，最后采用用户偏好的方式打印出来。



# 爬取网页内容

- 为了解析网页上数据，首先需要程序编写者观察爬虫页面的特点，即找到拟获取数据在HTML 页面中的格式。打开大学排名页面，在浏览器菜单中选择“查看网页源代码”，该选项在所有浏览器中都存在，得到的HTML 源代码如下图所示（为了便于阅读，该源代码做过一定排版）

# 爬取网页内容

```
<tbody class="hidden_zhpm" style="text-align:center;">
  <tr class="alt">
    <td>1</td><td><div align="left">清华大学</div></td><td>北京市</td><td>95.9</td><td class="hidden-xs need-hidden indicator5">100.0</td><td class="hidden-xs need-hidden indicator6" style="display:none;">97.90%</td><td class="hidden-xs need-hidden indicator7" style="display:none;">37342</td><td class="hidden-xs need-hidden indicator8" style="display:none;">1.298</td><td class="hidden-xs need-hidden indicator9" style="display:none;">1177</td><td class="hidden-xs need-hidden indicator10" style="display:none;">109</td><td class="hidden-xs need-hidden indicator11" style="display:none;">1137711</td><td class="hidden-xs need-hidden indicator12" style="display:none;">1187</td><td class="hidden-xs need-hidden indicator13" style="display:none;">593522</td>
  </tr>
  <tr>
    <td>2</td><td><div align="left">北京大学</div></td><td>北京市</td><td>82.6</td><td class="hidden-xs need-hidden indicator5">98.9</td><td class="hidden-xs need-hidden indicator6" style="display:none;">95.96%</td><td class="hidden-xs need-hidden indicator7" style="display:none;">36137</td><td class="hidden-xs need-hidden indicator8" style="display:none;">1.294</td><td class="hidden-xs need-hidden indicator9" style="display:none;">986</td><td class="hidden-xs need-hidden indicator10" style="display:none;">87</td><td class="hidden-xs need-hidden indicator11" style="display:none;">439403</td><td class="hidden-xs need-hidden indicator12" style="display:none;">799</td><td class="hidden-xs need-hidden indicator13" style="display:none;">7343</td>
  </tr>
  <tr class="alt">
    <td>3</td><td><div align="left">浙江大学</div></td><td>浙江省</td><td>80</td><td class="hidden-xs need-hidden indicator5">88.8</td><td class="hidden-xs need-hidden indicator6" style="display:none;">96.46%</td><td class="hidden-xs need-hidden indicator7" style="display:none;">41188</td><td class="hidden-xs need-hidden indicator8" style="display:none;">1.059</td><td class="hidden-xs need-hidden indicator9" style="display:none;">803</td><td class="hidden-xs need-hidden indicator10" style="display:none;">86</td><td class="hidden-xs need-hidden indicator11" style="display:none;">959511</td><td class="hidden-xs need-hidden indicator12" style="display:none;">833</td><td class="hidden-xs need-hidden indicator13" style="display:none;">64392</td>
  </tr>
```



# 爬取网页内容

- 对比两张图，每个大学排名的数据信息被封装在一个<tr></tr>之间的结构中。这是HTML 语言表示表格中一行的标签，在这行中，每列内容采用<td></td>表示。以“清华大学”为例，它对应一行信息的HTML 代码如下。



# 爬取网页内容

```
<tr class="alt">

  <td>1</td><td><div align="left">清华大学</div></td><td>北京市</td>

  <td>95.9</td><td class="hidden-xs need-hidden indicator5">100.0</td>

  <td class="hidden-xs need-hidden indicator6" style="display:none;">97.90%</td>

  <td class="hidden-xs need-hidden indicator7" style="display:none;">37342</td>

  <td class="hidden-xs need-hidden indicator8" style="display:none;">1.298</td>

  <td class="hidden-xs need-hidden indicator9" style="display:none;">1177</td>

  <td class="hidden-xs need-hidden indicator10" style="display:none;">109</td>

  <td class="hidden-xs need-hidden indicator11" style="display:none;">1137711</td>

  <td class="hidden-xs need-hidden indicator12" style="display:none;">1187</td>

  <td class="hidden-xs need-hidden indicator13" style="display:none;">593522</td>

</tr>
```



# 爬取网页内容

- 这个代码中每个td 标签包含大学排名表格的一个列数值，与表头一一对应。因此，如果要获得其中的数据，需要首先找到<tr> </tr> 标签，并遍历其中每个<td> </td> 标签，获取其值写入程序的数据结构中，这个代码封装成函数表示如下：





# 爬取网页内容

```
1 allUniv=[] #存储全部表格数据, 二维列表
2 def fillUnivList(soup):
3     data = soup.find_all('tr') #找到所有 tr 标签
4     for tr in data:
5         singleUniv = []
6         ltd = tr.find_all('td') #在每个 tr 标签中找到所有 td 标签
7         for td in ltd:
8             singleUniv.append(td.string) #提取 td 标签中信息
9         allUniv.append(singleUniv) data
```



# 爬取网页内容

- 上述逻辑尽管不错，却不完全。HTML 页面中除了显示大学排名的地方，其他位置也可能有表格和 `<tr> </tr>` 标签，应该尽量剔除这种情况。由于爬虫针对特定网页，程序编写也不必考虑所有情况，只要能应对当前页面即可。在这个大学排名页面中，还有一处用到了表格，包含 `<tr>` 标签，但这个标签内部不包括 `<td>` 标签。因此，可以通过增加一个判断语句剔除这种情况，观察下面代码的第6行和第7行。



# 爬取网页内容

```
1 allUniv=[] #存储全部表格数据，二维列表
2 def fillUnivList(soup):
3     data = soup.find_all('tr') #找到所有 tr 标签
4     for tr in data:
5         ltd = tr.find_all('td') #在每个 tr 标签中找到所有 td 标签
6         if len(ltd)==0:
7             continue
8         singleUniv = []
9         for td in ltd:
10             singleUniv.append(td.string) #提取 td 标签中信息
11         allUniv.append(singleUniv)
```

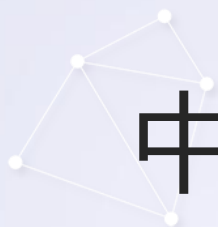


- 除了增加两行代码外，请将原函数的第5行调整为第8行，  
singleUniv=[]语句真实创建了一个空列表对象，它用于存储当前<tr>  
标签表示大学的数据。如果第6行if语句条件成立，说明当前读取的  
标签内容不是大学数据，如果创建了空列表将不再有用。因此，该语  
句调整到if语句后，只有在if语句判断该<tr>标签表示大学数据时才  
生成空列表，这样编写代码有利用占用更少的内存。



# 中国大学排名爬虫

- 也许在其他HTML 页面中会出现更多需要剔除的情况，而在这个大学排名网页中，仅需要剔除一种情况就能准确获得数据。大学排名爬虫完整源代码如下。



# 中国大学排名爬虫

实例代码 24.1

代码文件名 e24.1CrawUnivRanking.py

```
1  #e24.1CrawUnivRanking.py
2  import requests
3  from bs4 import BeautifulSoup
4  allUniv = []
5  def getHTMLText(url):
```



# 中国大学排名爬虫

```
6     try:
7         r = requests.get(url, timeout=30)
8         r.raise_for_status()
9         r.encoding = 'utf-8'
10        return r.text
11    except:
12        return ""
13    def fillUnivList(soup):
14        data = soup.find_all('tr')
15        for tr in data:
16            ltd = tr.find_all('td')
17            if len(ltd)==0:
18                continue
19            singleUniv = []
20            for td in ltd:
21                singleUniv.append(td.string)
22            allUniv.append(singleUniv)
```



# 中国大学排名爬虫

```
23 def printUnivList(num):
24     print("{:^4}{:^10}{:^5}{:^8}{:^10}".format("排名",\
        "学校名称", "省市", "总分", "培养规模"))
25     for i in range(num):
26         u=allUniv[i]
27         print("{:^4}{:^10}{:^5}{:^8}{:^10}".format(u[0],\
            u[1],u[2],u[3],u[6]))
28 def main(num):
29     url = 'http://www.zuihaodaxue.cn/\
        zuihaodaxuepaiming2016.html'
30     html = getHTMLText(url)
31     soup = BeautifulSoup(html, "html.parser")
32     fillUnivList(soup)
33     printUnivList(num)
34 main(10)
```





# 运行结果

>>>

排名	学校名称	省市	总分	培养规模
1	清华大学	北京市	95.9	37342
2	北京大学	北京市	82.6	36137
3	浙江大学	浙江省	80	41188
4	上海交通大学	上海市	78.7	40417
5	复旦大学	上海市	70.9	25519
6	南京大学	江苏省	66.1	20722
7	中国科学技术大学	安徽省	65.5	18507
8	哈尔滨工业大学	黑龙江省	63.5	25249
9	华中科技大学	湖北省	62.9	23503
10	中山大学	广东省	62.1	23837



# 爬虫程序修改

- 尽管实例代码完成了中国大学排名爬虫功能，但输出效果却不尽人意，各列内容并没有对齐。这是因为中文和数字字符占用的宽度不同，利用format()方法中{:N}方式只是约定了输出某个变量占用的字符个数，而没有实际上约束占用的字符宽度。这是中文和西文字符混排输出时经常遇到的问题。



# 爬虫程序修改

- 输出的每一列都有显著的类型特点，或者全是中文字符，或者全是数字。对于这类混排对齐问题，可以从填充字符角度考虑解决。



# 爬虫程序修改

24

```
print("{:^4}{:^10}{:^5}{:^8}{:^10}".format("排名",\
      "学校名称", "省市", "总分", "培养规模"))
```

27

```
print("{:^4}{:^10}{:^5}{:^8}{:^10}".format(u[0],\
      u[1],u[2],u[3],u[6]))
```



# 爬虫程序修改

- 解决这个问题一个简单的方法是替换填充字符，采用“中文全角空格”代替默认使用的“西文半角空格”，这能够对齐中文字符出现的列。修改后的printUnivList()函数代码如下，请用该代码替换实例代码中的printUnivList()函数。



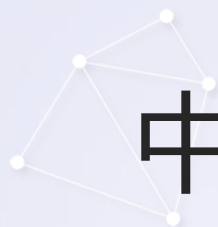
# 中国大学排名爬虫

```
1 def printUnivList(num):
2     print("{1:^2}{2:{0}^10}{3:{0}^6}{4:{0}^4}{5:{0}^10}".format
          \ (chr(12288), "排名", "学校名称", "省市", "总分", "培养规模"))
3     for i in range(num):
4         u = allUniv[i]
5         print("{1:^4}{2:{0}^10}{3:{0}^5}{4:{0}^8.1f}{5:{0}^10}".
          \ format(chr(12288), u[0], u[1], u[2], eval(u[3]), u[6]))
```



# 中国大学排名爬虫

- 上述函数中，中文全角空格采用chr(12288)表示，  
format()函数在标题栏和每行输出的参数经过手工调整。程序修改后运行输出效果如下



# 中国大学排名爬虫

>>>

排名	学校名称	省市	总分	培养规模
1	清华大学	北京市	95.9	37342
2	北京大学	北京市	82.6	36137
3	浙江大学	浙江省	80.0	41188
4	上海交通大学	上海市	78.7	40417
5	复旦大学	上海市	70.9	25519
6	南京大学	江苏省	66.1	20722
7	中国科学技术大学	安徽省	65.5	18507
8	哈尔滨工业大学	黑龙江省	63.5	25249
9	华中科技大学	湖北省	62.9	23503
10	中山大学	广东省	62.1	23837





# 拓展：大学排名

- 世界大学排名起源于美国，从1983 年开始每两年一次。由于排名对大学声誉和招生有非常深远的影响，大学排名逐渐成为一个产业。当前几大主流世界大学排名分别为：英国《泰晤士高等教育》杂志THE 世界大学排名、英国QS 世界大学排名、美国USNEWS 世界大学排名、荷兰莱顿大学世界大学排名、上海交通大学软科世界大学排名。

——这么多排名，该相信哪一个？

——角度不同、侧重不同、结果不同。要不写个爬虫抓回数据取个平均数？



# 思考与练习：

- [E12.7]思考实例代码24.1 中还可能有哪些改进？
- [E12.8]修改代码输出排名后50 位的大学。
- [E12.9]修改代码输出你所在省份学校大学的排名。



# 实例25 搜索关键词自动提交



## 要点：

- 这是一个向搜索引擎自动提交检索关键词并获取返回结果的实例。



# 搜索引擎

- 搜索引擎是日常工作常用的工具，也是访问互联网的门户。有时候需要自动向搜索引擎提交关键字并获得查询结果。
- 本节以百度为例介绍搜索关键字自动提交并获得返回结果的方法。




# 搜索关键词

- 百度搜索引擎首页为：<http://www.baidu.com>，当输入一个待查询关键词keyword时，百度程序将这个查询自动转换为链接：<http://www.baidu.com/s?wd=keyword>。读者可以在浏览器上手工输入这个链接，将keyword 换成任意想查询的关键字，都能获得查询结果。



# 搜索关键词自动提交

- 利用百度搜索提供的这个链接接口，可以通过 requests 的 get() 函数提交查询，响应结果为百度搜索结果。这个问题的 IPO 描述如下：
  - 输入：待查询关键字
  - 处理：自动获得百度搜索结果页面，并对页面内容解析处理
  - 输出：返回链接的标题列表



# 百度查询结果页面HTML 代码片段

- 首先人工分析百度查询结果页面HTML 代码，部分片段参考下图。
- 由于这些HTML 代码由机器自动生成，可读性较差，需要对比网页上的搜索结果和代码仔细寻找。



# 百度查询结果页面 HTML片段

```
> 《<em>程序设计基础 (Python语言)</em> 嵩天,黄天羽,礼欣 高等教育出... 京东</a></h3><div class="c-abstract"><span class="
newTimeFactor before_abs m">2014年7月1日 &nbsp;  </span> 《<em>程序设计基础 (Python语言)</em>
》系统地介绍了Pytnon程序设计和程序设计的基本方法。从<em>Python语言</em>发展历史、配置环境开始,详细介绍了 Pytnon语言
</div><div class="f13"><a target="_blank" href="
http://www.baidu.com/link?url=X98Iaztc9uf5P09PNDq8mlxNb0WV5sRHf8GFewJwJ0cHLn9NZoXP00050OYxDRZp" class="c-showurl" style=
"text-decoration:none;">item_id.com/17309803...&nbsp;  </a><div class="c-tools" id="tools_8214315112877157528_2" data-tools=
'{"title": "《程序设计基础 (Python语言) 嵩天,黄天羽,礼欣 高等教育出..._京东", "url": "
http://www.baidu.com/link?url=X98Iaztc9uf5P09PNDq8mlxNb0WV5sRHf8GFewJwJ0cHLn9NZoXP00050OYxDRZp"}'><a class="c-tip-icon"><i cl
"c-icon c-icon-triangle-down-g"></i></a></div><span class="c-icons-outer"><span class="c-icons-inner"></span></span>&nbsp;  -&
data-click="{ 'rsv_snapshot': '1' }" href="
http://cache.baiducontent.com/c?m=9d78d513d9810ae902b0c8690c66c0101843f7172ba1d30209d28449e3732b40501792ac51200770d3d27d1716c
2102461451b38cc9f85dadcb855c249f5732676f855612a20edfcc5153b237912ae9b81897ad803084d9d6c798140dc509433cc2e78a291d099129a74b26e
5f67ebb06f78fc5c762b942942b744f3f732195dc7a983&p=8b2a970293af0aff57ed9774504192&newp=c349c54ad6c04beb42a2c7710f4a83231610db21
98ffe0cc4241a1a3aecbf26231000d2c07a630aaf425cebf43777320634f1f689df08d2ecce7e6b9268&user=baidu&fm=sc&query=Python%D3%E
D0%F2%C9%E8%BC%C6%BB%F9%B4%A1%28%B5%DA2%B0%E6%29&qid=878cdb30000304eb&p1=2"
target="_blank" target="_blank"
class="m">百度快照</a><span class="c-pingjia">&nbsp;  <a href="
http://www.baidu.com/tools?url=http%3A%2F%2Fwww.baidu.com%2Flink%3Furl%3DX98Iaztc9uf5P09PNDq8mlxNb0WV5sRHf8GFewJ
xP00050OYxDRZp&iump=http%3A%2F%2Fkounbei.baidu.com%2F%2Fsentrv%3Ftitle%3D%01%E3%80%8A%02%E7%A8%8B%E5%BA%8F%01%E8
```



# 搜索关键词自动提交

- 经过分析发现，页面上返回结果标题被封装在如下

结构中：

- `<div...data-tools=`

`'{"title":"...", "url":"..."}'>...</div>`



# 搜索关键词自动提交

- 利用beautifulsoup4 库找到data-tools 属性值，提取带有title 的字符串，可以看到，data-tools 内部由{}形成的数据是典型的JSON 格式，可以用JSON 库将其转换成字典，便于操作。



# 搜索关键词自动提交

实例代码 25.1

e25.1AutoKeywordSearch.py

```
1  #e25.1AutoKeywordSearch.py
2  import requests
3  from bs4 import BeautifulSoup
4  import re
5  import json
6  def getKeywordResult(keyword):
7      url = 'http://www.baidu.com/s?wd='+keyword
8      try:
9          r = requests.get(url, timeout=30)
10         r.raise_for_status()
11         r.encoding = 'utf-8'
```



# 搜索关键词自动提交

```
12         return r.text
13     except:
14         return ""
15 def parserLinks(html):
16     soup = BeautifulSoup(html, "html.parser")
17     links = []
18     for div in soup.find_all('div', {'data-tools':\
                                   re.compile('title')}):
19         data = div.attrs['data-tools'] #获得属性值
20         d = json.loads(data)          #将属性值转换成字典
21         links.append(d['title'])      #将返回链接的题目返回
22     return links
```



# 搜索关键词自动提交

```
23 def main():
24     html = getKeywordResult('Python 语言程序设计基础(第2版)')
25     ls = parserLinks(html)
26     count = 1
27     for i in ls:
28         print("[{:^3}]{}".format(count, i))
29         count += 1
30 main()
```



# 运行结果

```
>>>
```

- [ 1 ] C 程序设计语言第 2 版·新版& 习题解答 - 下载频道 - CSDN.NET
- [ 2 ] 《程序设计基础(Python 语言) 嵩天,黄天羽,礼欣 高等教育出...\_京东
- [ 3 ] 程序设计基础 (PYTHON 语言) PDF 电子书下载 带书签目录 samp...\_微盘
- [ 4 ] Python 语言程序设计\_北京理工大学\_中国大学 MOOC (慕课)
- [ 5 ] python 基础教程(第 2 版·修订版) 中文版 高清 pdf 版[30MB]...\_脚本之家
- [ 6 ] 《程序设计基础(Python 语言)|4232695》【摘要 书评 试读】-...\_京东
- [ 7 ] 清华大学出版社-图书详情-《Python 程序设计基础》
- [ 8 ] python 基础教程(第 2 版)|python 基础教程中文高清 pdf【第...\_东坡下载
- [ 9 ] 程序设计入门—Python - 网易云课堂
- [10 ] Python 语言程序设计 python 基础教程



- 在技术层面，除了搜索引擎，还可以向其它可以查询数据信息的网页提交查询关键词。正是因为有这类自动提交程序，当今开发的服务网站不得不增加图片或声音类型的验证码，用来区分用户是计算机的自动程序还是人。技术是反映人类思想的手段，掌握了所谓“更有能力”和“更先进”的技术没什么大不了的，最为可贵的是去思考如何通过技术手段为人类和世界带来更美好的未来。





# 拓展：CAPTCHA 验证码

- CAPTCHA 验证码是 “全自动区分计算机和人类的图灵测试”（ Completely Automated Public Turing test to tell Computers and Humans Apart ）的缩写，它是一种区分用户是计算机程序或是人的方法，用于防止程序肆意向网络自动提交请求。



# 拓展：CAPTCHA 验证码

PPNIG

svierpe

just example



# 拓展：CAPTCHA 验证码

- CAPTCHA 验证码的原理是通过图片或语音形式展现人类容易识别但程序较难识别的信息，反馈信息与生成验证码的真实信息对比能够判断输入反馈的“用户”是程序或者是人类。CAPTCHA 验证码已经成为现代网络服务系统的标准配置。



# 思考与练习：

- [E12.10] 仔细观察百度搜索返回页面的HTML 代码，找到右侧“相关术语”部分对应的的代码。
- [E12.11] 仔细观察并解释百度图片搜索页面的HTML 代码。



# 本章小结

- 本章主要介绍了设计并实现网络爬虫的基本方法，结合requests 和beautifulsoup4 两个库的使用，讲述了如何处理HTTP协议以及解析和处理网页HTML和XML 页面信息的方法。本章通过中国大学排名爬虫和搜索关键词提交两个实例，展示了现代网络社会中快速抓取定向网页数据的重要价值。



# 程序练习题

- [P12.1] 参考实例24，实现按照省份输出中国大学排名功能。
- [P12.2] 参考实例24，实现USNews 美国大学排名的爬虫，并打印结果。
- [P12.3] 编写视频网站视频播放链接的爬虫。



# 程序练习题

- [P12.4] 编写爬取robots.txt 文件的爬虫，并分析爬取的内容。
- [P12.5] 编写根据robots.txt 文件内容爬取网站的程序。



# 程序练习题

- [P12.6] 分析百度图片搜索返回结果的HTML 代码，编写爬虫抓取图片并下载形成专题图片库。
- [P12.7] 编写程序测量30 秒内成功获得百度主页的次数。