**小练习: 爬百度百科**

作者: **莫烦**编辑: **莫烦 2017-12-29**

学习资料:

* [本节学习代码](https://github.com/MorvanZhou/easy-scraping-tutorial/blob/master/notebook/2-4-practice-baidu-baike.ipynb)
* 本节要爬的[百度百科](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB/5162711)

有了前面几节内容了练习, 我们现在完全有能力爬出你想要的信息了. 看吧, 我说很简单的, 只要你用 Python 打开网页, 用 BeautifulSoup 找准地方, 然后这样循环往复, 就叫做爬虫了. 哈哈. 被我抽象得不行了. 不过说到底, 爬虫就这么回事. 今天我们就来爬一爬百度百科, 让我们的爬虫从 “网络爬虫” 这一页开始爬, 然后在页面中寻找其他页面的信息, 然后爬去其他页面, 然后循环这么做, 看看最后我们的爬虫到底爬去了哪.

[**百度百科**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/2-04-practice-baidu-baike/#%E7%99%BE%E5%BA%A6%E7%99%BE%E7%A7%91)

百度百科中有很多名词的解释信息, 我们今天从 “网页爬虫” 的词条开始爬, 然后在页面中任意寻找下一个词条, 爬过去, 再寻找词条, 继续爬. 看看最后我们爬到的词条和 “网页爬虫” 差别有多大.

[](https://morvanzhou.github.io/static/results/scraping/2-4-1.png)

这个练习看起来挺没意义的, 但是对于了解爬虫, 还是挺有意义的. 用最简单的规律解释了爬虫的真谛.

[**观看规律**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/2-04-practice-baidu-baike/#%E8%A7%82%E7%9C%8B%E8%A7%84%E5%BE%8B)

这个爬虫说实在的, 并不难, 只有20+行代码. 但是却能让它游走在百度百科的知识的海洋中. 首先我们需要定义一个起始网页, 我选择了 “[网页爬虫](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB/5162711)”. 我们发现, 页面中有一些链接, 指向百度百科中的另外一些词条, 比如说下面这样.

<a target="\_blank" href="/item/%E8%9C%98%E8%9B%9B/8135707" data-lemmaid="8135707">蜘蛛</a>

<a target="\_blank" href="/item/%E8%A0%95%E8%99%AB">蠕虫</a>

<a target="\_blank" href="/item/%E9%80%9A%E7%94%A8%E6%90%9C%E7%B4%A2%E5%BC%95%E6%93%8E">通用搜索引擎</a>

通过观察, 我们发现, 链接有些共通之处. 它们都是 /item/ 开头, 夹杂着一些 %E9 这样的东西. 但是仔细搜索一下, 发现还有一些以 /item/ 开头的, 却不是词条. 比如

<a href="/item/史记·2016?fr=navbar" target="\_blank">史记·2016</a>

我想, 我们需要对这些链接做一些筛选, [之前提到](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/2-03-beautifulsoup-regex/) 的用 BeautifulSoup 和 正则表达式来筛选应该用得上. 有了些思路, 我们开始写代码吧.

[**制作爬虫**](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/2-04-practice-baidu-baike/#%E5%88%B6%E4%BD%9C%E7%88%AC%E8%99%AB)

导入一些模块, 设置起始页. 并将 /item/... 的网页都放在 his 中, 做一个备案, 记录我们浏览过的网页.

**from** bs4 **import** BeautifulSoup

**from** urllib.request **import** urlopen

**import** re

**import** random

base\_url = "https://baike.baidu.com"

his = ["/item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB/5162711"]

接着我们先不用循环, 对一个网页进行处理, 走一遍流程, 然后加上循环, 让我们的爬虫能在很多网页中爬取. 下面做的事情, 是为了在屏幕上打印出来我们现在正在哪张网页上, 网页的名字叫什么.

url = base\_url + his[-1]

html = urlopen(url).read().decode('utf-8')

soup = BeautifulSoup(html, features='lxml')

print(soup.find('h1').get\_text(), ' url: ', his[-1])

# 网络爬虫 url: /item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB/5162711

接下来我们开始在这个网页上找所有符合要求的 /item/ 网址. 使用一个正则表达式([正则教程](https://morvanzhou.github.io/tutorials/python-basic/basic/13-10-regular-expression/)) 过滤掉不想要的网址形式. 这样我们找到的网址都是 /item/%xx%xx%xx... 这样的格式了. 之后我们在这些过滤后的网页中随机选一个, 当做下一个要爬的网页. 不过有时候很不幸, 在 sub\_urls 中并不能找到合适的网页, 我们就往回跳一个网页, 回到之前的网页中再随机抽一个网页做同样的事.

# find valid urls

sub\_urls = soup.find\_all("a", {"target": "\_blank", "href": re.compile("/item/(%.{2})+$")})

if len(sub\_urls) != 0:

his.append(random.sample(sub\_urls, 1)[0]['href'])

else:

# no valid sub link found

his.pop()

print(his)

# ['/item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB/5162711', '/item/%E4%B8%8B%E8%BD%BD%E8%80%85']

有了这套体系, 我们就能把它放在一个 for loop 中, 让它在各种不同的网页中跳来跳去.

his = ["/item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB/5162711"]

for i in range(20):

url = base\_url + his[-1]

html = urlopen(url).read().decode('utf-8')

soup = BeautifulSoup(html, features='lxml')

print(i, soup.find('h1').get\_text(), ' url: ', his[-1])

# find valid urls

sub\_urls = soup.find\_all("a", {"target": "\_blank", "href": re.compile("/item/(%.{2})+$")})

if len(sub\_urls) != 0:

his.append(random.sample(sub\_urls, 1)[0]['href'])

else:

# no valid sub link found

his.pop()

这样我们就能观看我们的爬虫现在爬去了哪? 是不是爬到了和 “网页爬虫” 起始页完全不相关的地方去了.

0 网络爬虫 url: /item/%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB/5162711

1 路由器 url: /item/%E8%B7%AF%E7%94%B1%E5%99%A8

2 服务等级 url: /item/%E6%9C%8D%E5%8A%A1%E7%AD%89%E7%BA%A7

...

17 呼损率 url: /item/%E5%91%BC%E6%8D%9F%E7%8E%87

18 服务等级 url: /item/%E6%9C%8D%E5%8A%A1%E7%AD%89%E7%BA%A7

19 呼损率 url: /item/%E5%91%BC%E6%8D%9F%E7%8E%87

哈哈哈果然! 经过了20个循环, 它爬到了”呼损率”, 虽然我都不知道这是什么, 看了看这个网页, 又学了新知识了. 这样想起来, 还挺好玩的.

[](https://morvanzhou.github.io/static/results/scraping/2-4-2.png)

[接下来](https://morvanzhou.github.io/tutorials/data-manipulation/scraping/3-01-requests/)来的爬虫, 我们将要了解更多需要知道的功能. 用 requests 代替 urlopen, 还有如何从网页下载等.