


Python数据抓取与爬虫基础 II

微信公众号爬虫实战

林平之老师

扫描二维码关注微信/小程序，获取最新面试题及权威解答



 微信扫一扫，使用小程序



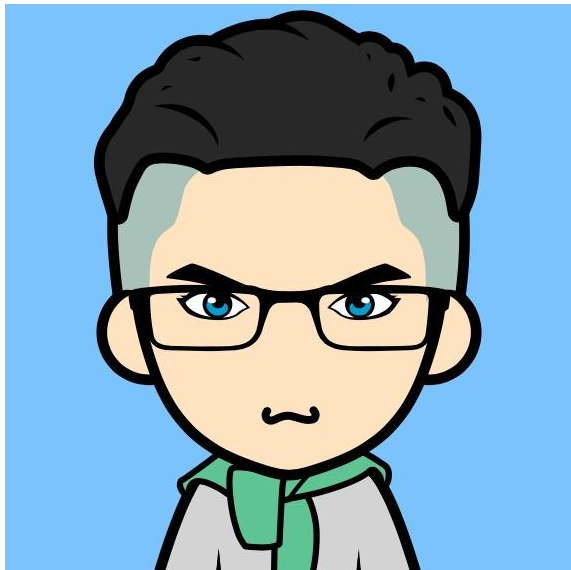
微信扫一扫，关注微信公众号

版权声明

九章课程严禁录制视频的侵权行为

否则将追求法律责任和经济赔偿

请一定不要缺课



林老师

全国算法竞赛一等奖

国内TOP2名校毕业

参加国家信息学竞赛NOI

前FLAG工程师

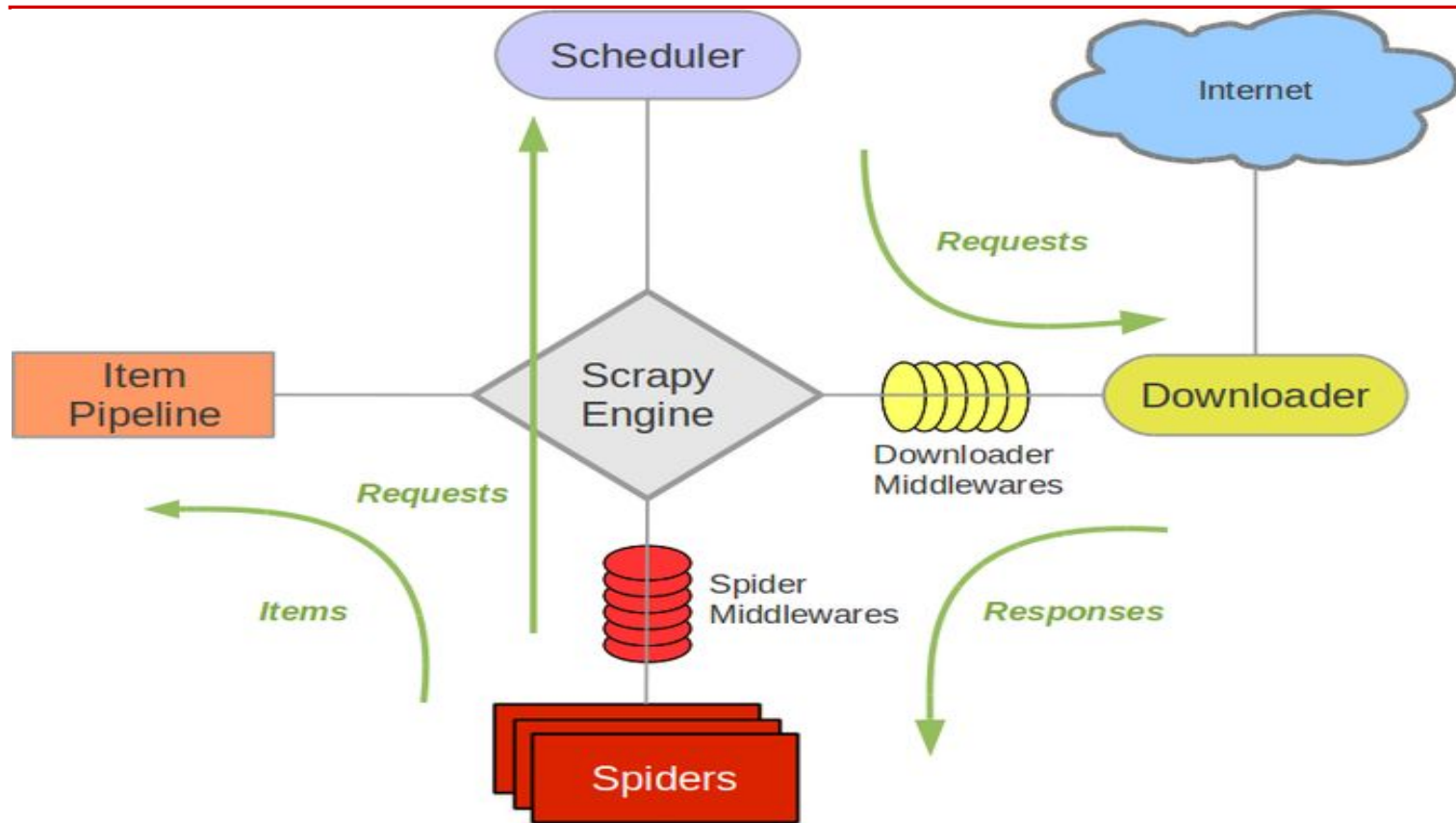
拥有丰富的面试经验

- Scrapy回顾
- <http://weixin.sogou.com/>
- 设计抓取微信公众号文章
 - 获取公众号信息
 - 获取最近的文章
- 爬虫利器Beautiful Soup的用法
- 补充知识：
 - Python的多线程实现方式
 - Python的Queue模块

Spider介绍



九章算法



1. 编写函数parse, 该函数名不能改变, 因为Scrapy源码中默认callback函数的函数名就是parse
2. `def parse(self, response):` 参数是response
3. parse出来新的url可以进行新的request请求, 也可以设置新的Callback
- 4.

```
yield Request(profile_link, callback=self.parse_profile)
```

- <http://weixin.sogou.com/> 通过sogou获取到公众号的信息和最近的历史10条记录
- 通过多线程的方式抓取这10篇微信文章

搜狗 | 微信

搜文章

搜公众号

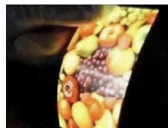


热门 要闻 段子手 养生堂 私房话 八卦精 科技咖 财经 汽车 生活 更多 ▾

可折叠!可弯曲!能“卷起来”的国产手机快来了!

近日，我国首条柔性显示屏生产线在成都实现量产，打破了国外企业在这一领域的垄断，加速我国移动电子产品进入柔性显示时代，卷起来的手机离我们不再遥远。实现量产的柔性屏生产线，是我国企业京东方完全自主设计...

新华网 2小时前



去年万圣节扮无脸男爆红的台湾小朋友，今年的造型也非常成功!



搜索热词

热度

- 1 杨利伟荣获奖章
- 2 尼泊尔客车坠河
- 3 上海无人馆叫停
- 4 清理僵尸粉被转账
- 5 送婆婆法院告自己
- 6 在澳留学生被围殴
- 7 女孩遭萨摩耶袭击
- 8 昆明现女性停车位
- 9 桂三高速全线开通
- 10 麦当劳改名金拱门

编辑精选

超级赞! 中国又一产业异军突起, 逆袭韩国走向世界!

人民网

10月27日



It girl脱下了小白鞋 又换了双



在网页上搜索“九章算法”



https://weixin.sogou.com/weixin?type=2&query=%E4%B9%9D%E7%AB%A0%E7%AE%97%E6%B3%95&ie=utf8&s_from=input&sug=n&sug_type=&w=01019900&sut=1507&sst0=1544406952236&lkt=1%2C1544406952134%2C1544406952134

如何解读这个URL ？

？后面是URL的参数，每个参数是key=value的形式存在，然后用&符号串联起来

在网页上搜索“九章算法”



九章算法



搜狗 微信

新闻

网页

微信

知乎

图片

视频

明医

英文

问问

学术

更多>>

九章算法



搜文章

搜公众号

以下内容来自微信公众号



九章算法



微信号: ninechapter



功能介绍: 专业的北美IT求职经验分享、技术交流社区,帮助你找到好的IT工作.由硅谷顶尖的IT企业工程师授课,提供专业的算法培训/面试咨询.官网:www.jiuzhang.com

微信认证: 九章算法(杭州)科技有限公司

最近文章: 人工智能集训营 | 无限次免费重听,早报名早学习

2天前

- 生成对应的URL

```
query = OrderedDict()  
query['type'] = _search_gzh  
query['page'] = page  
query['ie'] = 'utf8'  
query['query'] = keyword  
  
return 'http://weixin.sogou.com/weixin?%s' % parse.urlencode(query)
```

- 使用requests抓起对应网页内容
- 解析抓回来的网页内容

从公众号的Profile URL链接中, 通过访问, 我们可以获取到公众号最近发布的10篇文章

[http://mp.weixin.qq.com/profile?src=3×tamp=1544406982&ver=1&signature=7A*
nf*aRFC-IXoLwWiJVxJOeW0rDQzPqX5W8jchJY869Mex0aWd6feVLNagKtW6uzieGK
bpmeZJL3n*57gUkbw==](http://mp.weixin.qq.com/profile?src=3×tamp=1544406982&ver=1&signature=7A*
nf*aRFC-IXoLwWiJVxJOeW0rDQzPqX5W8jchJY869Mex0aWd6feVLNagKtW6uzieGK
bpmeZJL3n*57gUkbw==)

(2018-12-09)

如何从网页中提取URL(提取10篇文章)?

pip3 install beautifulsoup4 安装bs4

```
from bs4 import BeautifulSoup
import requests

html = requests.get("http://weixin.sogou....A0%E7%AE%97%E6%B3%95").text
print(html)
```

Beautiful Soup可以方便的帮助我们
提取网页中的标签内容

读取网页后, 我们要尝试加载
进 BeautifulSoup

```
from bs4 import BeautifulSoup
import requests

html = requests.get("https://www.jiuzhang.com").text

soup = BeautifulSoup(html, features='lxml')
print(soup.script)
```

我们用lxml的形式来解析这个网页，如果网页内有元素script，我们可以直接使用soup.script来获取信息。

但是script标签有多个，如何是好？

Beautiful Soup的用法



九章算法

```
from bs4 import BeautifulSoup
import requests
```

```
html = requests.get("https://www.jiuzhang.com").text
```

```
soup = BeautifulSoup(html, features='lxml')
soup.find_all('script')
```

利用find_all返回所有tag是script的元素

find_all利用正则表达式

```
from bs4 import BeautifulSoup
import requests
import re

html = requests.get("https://www.jiuzhang.com").text

soup = BeautifulSoup(html, features='lxml')
soup.find_all('script', {'src': re.compile('.*min.js')})
```

找到所有script边间中, 属性src结尾是min.js的

令 html文本如下：

```
html = ""
<html><head><title>The crawler course</title></head>
<body>
<p class="title" name="crawler"><b>crawler content</b></p>
<p class="course">Week4.Session1 Python Data Capture & Crawler II V2.0
<a href="http://example.com/spider1" class="spider" id="link1"><!-- spider1 --></a>,
<a href="http://example.com/spider2" class="spider" id="link2">spider2</a>,
<a href="http://example.com/spider3" class="spider" id="link3">spider3</a>,
Hahahaha </p>
<p class="course">...</p>
""
```

利用Beautiful Soup格式化输出内容

```
soup = BeautifulSoup(html)
print(soup.prettify())
```

```
<html>
<head>
  <title>
    The crawler  course
  </title>
</head>
<body>
  <p class="title" name="crawler">
    <b>
      crawler content
    </b>
  </p>
  <p class="course">
```

利用Beautiful Soup对标签tag内容的提取

- soup.head
- soup.title
- 查看对象的雷霆 `type(soup.a)`

Beautiful Soup标签tag两大重要属性attrs和name

- soup.p.name
- soup.p.attrs # 范围: {'class': ['title'], 'name': 'crawler'}
- 如何提取class的内容?
 - soup.p['class'] 或者 soup.p.get('class')

Select通过css元素

- 标签名不加任何修饰
- 类名前加点
- id名前加 #

Select通过css元素

- 标签名不加任何修饰
- 通过标签名查找: `soup.select('title')`

Select通过css元素

- 类名前加点
- 通过类名查找: `soup.select('.title')`

Select通过css元素

- id名前加 #
- 通过id名查找:soup.select('#name')

Select通过css元素

- 组合查找
- `soup.select('.title #name')`

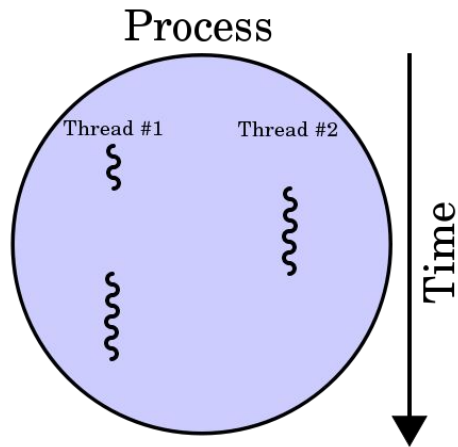
Select属性查找：

```
from bs4 import BeautifulSoup
import requests
import re

html = requests.get("https://www.jiuzhang.com").text

soup = BeautifulSoup(html, features='lxml')
print(soup.body)
print(soup.select('div[class~="text-center"]'))
```

- 多线程好比同时执行多个不同程序片段，优点如下：
 - 一些任务可以放置在后台运行，面向用户的程序可以在前台运行，可以有更好的用户体验
 - 程序的运行速度可能加快，多线程并发



- 函数
 - `_thread`模块中的`start_new_thread()`函数来产生新线程（一般不建议使用）
- 类来包装线程对象
 - `threading`模块中继承父类`threading.Thread`

原来Python2中`thread` 模块已被废弃。推荐直接使用`threading`，为了兼容性，Python3 将 `thread` 重命名为 "`_thread`"。

- `_thread`和`threading`模块都可以用来创建和管理
- `_thread`模块提供了基本的线程和锁支持
- `threading`提供的是更高级的完全的线程管理
- 低级别的`_thread`模块是推荐给高手用
- 普通应用程序推荐使用更高级的`threading`模块即可

_thread模块中的start_new_thread()函数



```
import _thread
import time

def func(name, delay):
    count = 0
    while count < 10:
        time.sleep(delay)
        count += 1
        print(name, time.ctime(time.time()))

_thread.start_new_thread(func, ('thread-1', 2))
_thread.start_new_thread(func, ('thread-2', 3))

while True:
    pass
```

- 参数
 - function - 线程函数名
 - 比如Func
 - args - 传递给线程函数的参数, 必须是个tuple
 - 比如('Thread-1', 3)

Threading模块提供了Thread类来处理线程

- 这里的Thread类提供了以下方法:
 - run() - 相当于定义一个该线程运行的函数
 - start() - 开始启动线程
 - join() - 等待线程中止
 - isAlive() - 返回线程是否还在活动

```
import threading
import time
```

```
class TestThread(threading.Thread):

    def __init__(self, name, delay):
        threading.Thread.__init__(self)
        self.name = name
        self.delay = delay

    def run(self):
        func(self.name, self.delay)|
```

1. 继承自threading.Thread
2. run方法内写入需要该线程运行的方法

需要定义func函数和创建2个TestThread的对象

```
def func(name, delay):  
    count = 0  
    while count < 3:  
        time.sleep(delay)  
        count += 1  
        print(name, time.ctime(time.time()))
```

```
thread1 = TestThread('thread-1', 2)  
thread2 = TestThread('thread-2', 3)
```

```
thread1.start()  
thread2.start()
```

```
thread1.join()  
thread2.join()
```

使用Threading模块创建线程：

1. 直接从threading.Thread继承
2. 重写__init__和run方法
3. 创建新生成的线程类的对象
4. start 开始运行线程
5. join 等待线程结束并退出

- 什么是Queue(队列)？
-
- 队列是线程间最常用的交换数据的方式
- Queue模块是一个提供了队列操作的模块

创建 : `queue = Queue.Queue()`

获取元素 : `queue.get()`

放入元素 : `queue.put(element)`

当前新增 :

`join & task_done`

queue.join()

- 阻塞调用线程，一直等待，直到队列中的全部任务被处理。
只要有数据被加入队列既put(element)被调用，未完成任务数量就会增加。当有线程调用task_done()，那么就是有线程**取得任务并完成任务**，未完成任务数就会减少
- 当未完成任务数降到0，join()解除阻塞
- 因此使用Queue我们**不需要自己去实现统计**当前是否还有未完成任务

`queue.task_done()`

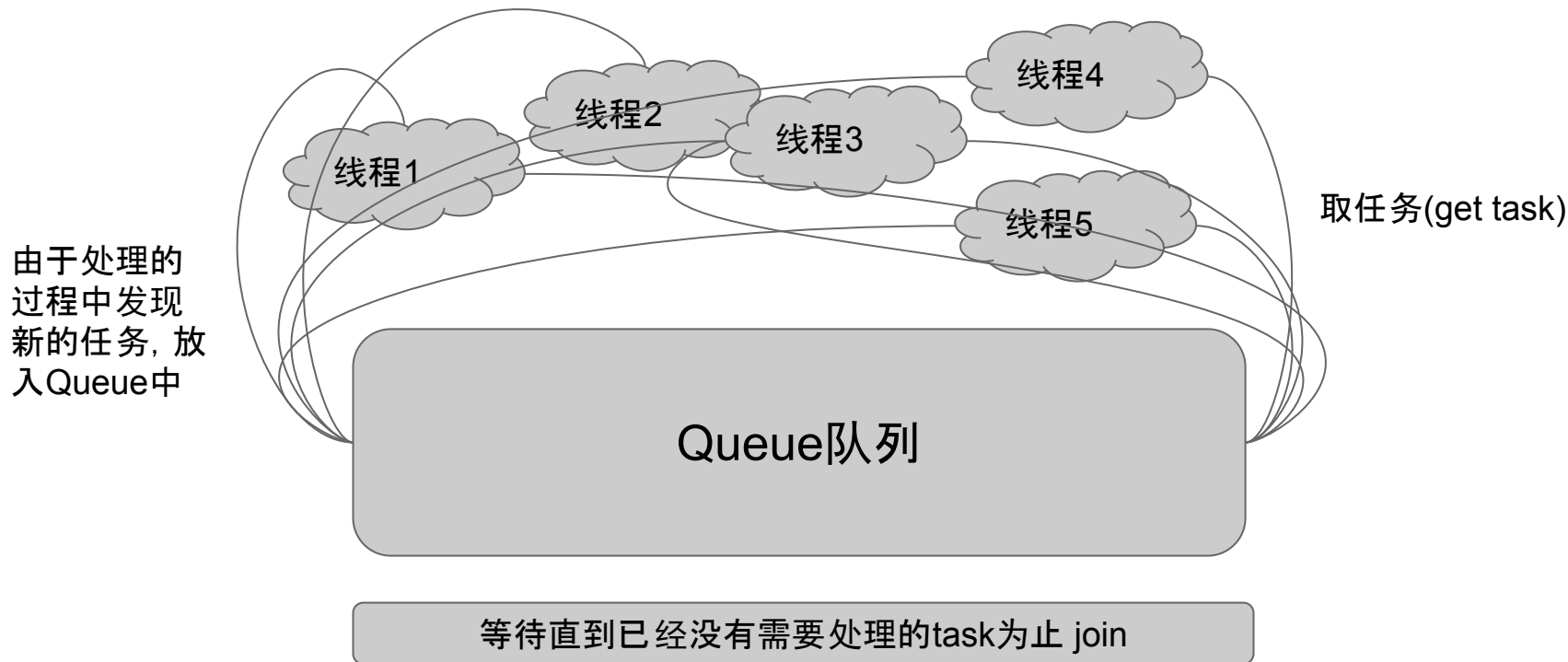
- 当我们从Queue中获取一个任务即`get()`得到一个任务，在这个任务执行完成之后，我们可以调用`task_done()`来告诉队列该任务已经处理完毕，使得需要执行的任务数量减少一
- 如果该方法被调用的次数多于被放入队列中的任务的个数，会有异常抛出。

1. 创建一个Queue
2. 创建多个Thread
3. 多个Thread共享一个Queue
4. 线程从Queue中取出任务(get task), 操作完成之后调用task_done
5. 执行task的时候发现新的任务, 使用put将这个任务塞入Queue
6. 使用join, 直到所有task都执行完毕

基于Queue实现多线程crawler



九章算法



更多阅读材料: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/https://doc.scrapy.org/en/latest/intro/tutorial.html>



谢谢大家