**001、试列出至少三种目前流行的大型数据库的名称:\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_,其中您最熟悉的是\_\_\_\_\_\_\_\_\_\_,从\_\_\_\_\_\_\_\_\_\_年开始使用。**

Oracle，Mysql，SQLServer Oracle根据自己情况

**002、有表List，并有字段A、B、C，类型都是整数。表中有如下几条记录：**

A B C

2 7 9

5 6 4

3 11 9

现在对该表一次完成以下操作：

查询出B和C列的值，要求按B列升序排列

写出一条新的记录，值为{7,9,8}

查询C列，要求消除重复的值，按降序排列

写出完成完成以上操作的标准的SQL语句，并且写出操作3的结果。

create table List(A int,B int,C int)

Select B,C from List order by B

Insert into List values(7,9,8)

Select distinct(C) from List order by desc;

984

**003、请简要说明视图的作用？**

1.数据库视图隐藏了数据的复杂性。

2.数据库视图有利于控制用户对表中某些列的访问。

3.数据库视图使用户查询变得简单。

**004、列举您使用过的python网络爬虫所用到的网络数据包（最熟悉的在前）：**

requests、urllib、urllib2、httplib2

1. 请求

requests（第三方模块）

2. 解析：

bs4（即beautifulsoup，第三方模块）

3. 储存：

pymongo（第三方模块）：

　　把数据写入MongoDB

MySQL-python（第三方模块）：

　　把数据写入MySQL里面。

协程：gevent（第三方模块）

二、Python数据分析&科学计算

numpy（第三方模块，C拓展）：

　　Copy了MATLAB的数据结构。很多数据分析和科学计算库的底层模块。提供了良好的数组数据结构和C拓展接口。

pandas（第三方模块，C拓展）：

　　Copy了R的data frame的数据结构。

**005、列举您使用过的python网络爬虫所用到的解析数据包（最熟悉的在前）：**

BeautifulSoup、pyquery、Xpath、lxml

**006、列举您使用过的python中的编码方式（最熟悉的在前）：**

UTF-8，ASCII，gbk

**007、python3.5语言中enumerate的意思是？**

对于一个可迭代的（iterable）/可遍历的对象（如列表、字符串），enumerate将其组成一个索引序列，利用它可以同时获得索引和值

enumerate多用于在for循环中得到计数

**008、99的八进制表示是：143**

**009、请举出三种常用的排序算法？**

冒泡、选择、快速

**010、列出比较熟悉的爬虫框架？**

Scrapy

**011、用4、9、2、7四个数字，可以使用+、-、\* 和 /，每个数字使用一次，使表达式的结果为24？**

表达式是（9+7-4）\* 2

**012、您最熟悉的Unix环境是？Unix下查询环境变量的命令是？查询脚本定时任务的命令是？**

1AIX，env, crontab

**013、写出在网络爬虫爬取数据的过程中，遇到的防爬虫问题的解决方案**

通过headers反爬虫：解决策略，伪造headers

基于用户行为反爬虫：动态变化去爬取数据，模拟普通用户的行为

基于动态页面的反爬虫：跟踪服务器发送的ajax请求，模拟ajax请求

**014、阅读以下Python程序？**

for i in range(5,0,-1):

print(i)

请在下面写出打印结果：54321

**015、简述 requests模块的作用及基本使用？**

# 作用：

使用requests可以模拟浏览器的请求

# 常用参数：

url、headers、cookies、data、json、params、proxy

# 常用返回值：

content

iter\_content

text

encoding="utf-8"

cookie.get\_dict()

**016、简述 Beautifulsoup模块的作用及基本使用？**

#BeautifulSoup

用于从HTML或XML文件中提取、过滤想要的数据形式

#常用方法

解析：html.parser 或者 lxml（需要下载安装） 、find、find\_all、text、attrs、get

<https://www.cnblogs.com/wcwnina/p/8093987.html>

**018、Scrapy框架中各组件的工作流程？**

Scrapy 使用了 Twisted 异步非阻塞网络库来处理网络通讯，整体架构大致如下（绿线是数据流向）：



Scrapy主要包括了以下组件：

* **引擎(Scrapy)**
* *用来处理整个系统的数据流处理, 触发事务(框架核心)*
* **调度器(Scheduler)**
* *用来接受引擎发过来的请求, 压入队列中, 并在引擎再次请求的时候返回. 可以想像成一个URL（抓取网页的网址或者说是链接）的优先队列, 由它来决定下一个要抓取的网址是什么, 同时去除重复的网址*
* **下载器(Downloader)**
* *用于下载网页内容, 并将网页内容返回给蜘蛛(Scrapy下载器是建立在twisted这个高效的异步模型上的)*
* **爬虫(Spiders)**
* *爬虫是主要干活的, 用于从特定的网页中提取自己需要的信息, 即所谓的实体(Item)。用户也可以从中提取出链接,让Scrapy继续抓取下一个页面*
* **项目管道(Pipeline)**
* *负责处理爬虫从网页中抽取的实体，主要的功能是持久化实体、验证实体的有效性、清除不需要的信息。当页面被爬虫解析后，将被发送到项目管道，并经过几个特定的次序处理数据。*
* **下载器中间件(Downloader Middlewares)**
* *介于Scrapy引擎和下载器之间的中间件，主要是处理Scrapy引擎与下载器之间的请求及响应。*
* **爬虫中间件(Spider Middlewares)**
* *介于Scrapy引擎和爬虫之间的中间件，主要工作是处理蜘蛛的响应输入和请求输出。*
* **调度中间件(Scheduler Middewares)**
* *介于Scrapy引擎和调度之间的中间件，从Scrapy引擎发送到调度的请求和响应。*

Scrapy运行流程大概如下：

1.引擎：Hi！Spider, 你要处理哪一个网站？

2.Spider：老大要我处理xxxx.com（初始URL）。

3.引擎：你把第一个需要处理的URL给我吧。

4.Spider：给你，第一个URL是xxxxxxx.com。

5.引擎：Hi！调度器，我这有request请求你帮我排序入队一下。

6.调度器：好的，正在处理你等一下。

7.引擎：Hi！调度器，把你处理好的request请求给我。

8.调度器：给你，这是我处理好的request

9.引擎：Hi！下载器，你按照老大的下载中间件的设置帮我下载一下这个request请求。

10.下载器：好的！给你，这是下载好的东西。（如果失败：sorry，这个request下载失败了。然后引擎告诉调度器，这个request下载失败了，你记录一下，我们待会儿再下载）

11.引擎：Hi！Spider，这是下载好的东西，并且已经按照老大的下载中间件处理过了，你自己处理一下（注意！这儿responses默认是交给def parse()这个函数处理的）

12.Spider：（处理完毕数据之后对于需要跟进的URL），Hi！引擎，我这里有两个结果，这个是我需要跟进的URL，还有这个是我获取到的Item数据。

13.引擎：Hi ！管道 我这儿有个item你帮我处理一下！调度器！这是需要跟进URL你帮我处理下。然后从第四步开始循环，直到获取完老大需要全部信息。

14.管道、调度器：好的，现在就做！

<https://blog.csdn.net/qq_37143745/article/details/80996707>

**030、项目发布的过程（上传-发布），上线，发布，测试，谁发布测试。**

**031、pandas库有哪些函数，介绍pandas函数的作用：**

前言：本博文摘抄自中国慕课大学上的课程《Python数据分析与展示》，推荐刚入门的同学去学习，这是非常好的入门视频。

继续一个新的库，Pandas库。Pandas库围绕Series类型和DataFrame类型这两种数据结构，提供了一种高效便捷的数据处理方式。

一、常用功能及函数简介

包导入

一般我们需要做如下导入，numpy和pandas一般需要联合使用：

import pandas as pd

import numpy as np

本文采用如下缩写：

df：Pandas DataFrame对象

s：  Pandas Series对象

数据导入

pd.read\_csv(filename)：从CSV文件导入数据

pd.read\_table(filename)：从限定分隔符的文本文件导入数据

pd.read\_excel(filename)：从Excel文件导入数据

pd.read\_sql(query, connection\_object)：从SQL表/库导入数据

pd.read\_json(json\_string)：从JSON格式的字符串导入数据

pd.read\_html(url)：解析URL、字符串或者HTML文件

pd.read\_clipboard()：从粘贴板获取内容

pd.DataFrame(dict)：从字典对象导入数据

数据导出

df.to\_csv(filename)：导出数据到CSV文件

df.to\_excel(filename)：导出数据到Excel文件

df.to\_sql(table\_name, connection\_object)：导出数据到SQL表

df.to\_json(filename)：以Json格式导出数据到文本文件

创建对象

pd.DataFrame(np.random.rand(20,5))：创建20行5列的随机数组成的DataFrame对象

pd.Series(my\_list)：从可迭代对象my\_list创建一个Series对象

df.index = pd.date\_range('1900/1/30', periods=df.shape[0])：增加一个日期索引

index和reindex联合使用很有用处，index可作为索引并且元素乱排序之后，所以跟着元素保持不变，因此，当重拍元素时，只需要对index进行才重排即可:reindex。

另外， reindex时，还可以增加新的标为NaN的元素。

数据查看

df.head(n)：查看DataFrame对象的前n行

df.tail(n)：查看DataFrame对象的最后n行

df.shape()：查看行数和列数

df.info()：查看索引、数据类型和内存信息

df.describe()：查看数值型列的汇总统计

s.value\_counts(dropna=False)：查看Series对象的唯一值和计数

df.apply(pd.Series.value\_counts)：查看DataFrame对象中每一列的唯一值和计数

apply的用处很多，比如可以通过跟lambda函数联合，完成很多功能：将包含某个部分的元素挑出来等等。

cities['Is wide and has saint name'] = (cities['Area square miles'] > 50) & cities['City name'].apply(lambda name: name.startswith('San'))

数据选取

df[col]：根据列名，并以Series的形式返回列

df[[col1, col2]]：以DataFrame形式返回多列

s.iloc[0]：按位置选取数据

s.loc['index\_one']：按索引选取数据

df.iloc[0,:]：返回第一行

数据清洗

df.columns = ['a','b','c']：重命名列名

pd.isnull()：检查DataFrame对象中的空值，并返回一个Boolean数组

pd.notnull()：检查DataFrame对象中的非空值，并返回一个Boolean数组

df.dropna()：删除所有包含空值的行

df.fillna(x)：用x替换DataFrame对象中所有的空值

s.astype(float)：将Series中的数据类型更改为float类型

s.replace(1,'one')：用‘one’代替所有等于1的值

df.rename(columns=lambda x: x + 1)：批量更改列名

df.set\_index('column\_one')：更改索引列

数据处理：Filter, Sort, GroupBy

df[df[col] > 0.5]：选择col列的值大于0.5的行

df.sort\_values(col1)：按照列col1排序数据，默认升序排列

df.groupby(col)：返回一个按列col进行分组的Groupby对象

df.groupby(col1).agg(np.mean)：返回按列col1分组的所有列的均值

df.pivot\_table(index=col1, values=[col2,col3], aggfunc=max)：创建一个按列col1进行分组，并计算col2和col3的最大值的数据透视表

data.apply(np.mean)：对DataFrame中的每一列应用函数np.mean

数据合并

df1.append(df2)：将df2中的行添加到df1的尾部

df.concat([df1, df2],axis=1)：将df2中的列添加到df1的尾部

df1.join(df2,on=col1,how='inner')：对df1的列和df2的列执行SQL形式的join

数据统计

df.describe()：查看数据值列的汇总统计

df.mean()：返回所有列的均值

df.corr()：返回列与列之间的相关系数

df.count()：返回每一列中的非空值的个数

df.max()：返回每一列的最大值

df.min()：返回每一列的最小值

df.median()：返回每一列的中位数

df.std()：返回每一列的标准差

Pandas支持的数据类型

int 整型

float 浮点型

bool 布尔类型

object 字符串类型

category 种类

datetime 时间类型

补充：

df.astypes: 数据格式转换

df.value\_counts:相同数值的个数统计

df.hist(): 画直方图

df.get\_dummies: one-hot编码，将类型格式的属性转换成矩阵型的属性。比如：三种颜色RGB，红色编码为[1 0 0]

**160.列举您使用过的Python网络爬虫所用到的网络数据包?**

requests, urllib,urllib2, httplib2

**161.爬取数据后使用哪个数据库存储数据的，为什么？**

## 162、你用过的爬虫框架或者模块有哪些？谈谈他们的区别或者优缺点？

Python自带：urllib，urllib2

第 三 方：requests

框    架：Scrapy

urllib和urllib2模块都做与请求URL相关的操作，但他们提供不同的功能。

urllib2.：urllib2.urlopen可以接受一个Request对象或者url，（在接受Request对象时候，并以此可以来设置一个URL 的headers），urllib.urlopen只接收一个url

urllib 有urlencode,urllib2没有，因此总是urllib，urllib2常会一起使用的原因

scrapy是封装起来的框架，他包含了下载器，解析器，日志及异常处理，基于多线程， twisted的方式处理，对于固定单个网站的爬取开发，有优势，但是对于多网站爬取 100个网站，并发及分布式处理方面，不够灵活，不便调整与括展。

request 是一个HTTP库， 它只是用来，进行请求，对于HTTP请求，他是一个强大的库，下载，解析全部自己处理，灵活性更高，高并发与分布式部署也非常灵活，对于功能可以更好实现.

Scrapy优缺点：

优点：scrapy 是异步的

采取可读性更强的xpath代替正则

强大的统计和log系统

同时在不同的url上爬行

支持shell方式，方便独立调试

写middleware,方便写一些统一的过滤器

通过管道的方式存入数据库

缺点：基于python的爬虫框架，扩展性比较差

基于twisted框架，运行中的exception是不会干掉reactor，并且异步框架出错后是不会停掉其他任务的，数据出错后难以察觉。

## 007、写爬虫是用多进程好？还是多线程好？为什么？

IO密集型代码(文件处理、网络爬虫等)，多线程能够有效提升效率(单线程下有IO操作会进行IO等待，造成不必要的时间浪费，而开启多线程能在线程A等待时，自动切换到线程B，可以不浪费CPU的资源，从而能提升程序执行效率)。在实际的数据采集过程中，既考虑网速和响应的问题，也需要考虑自身机器的硬件情况，来设置多进程或多线程

## 008、常见的反爬虫和应对方法？

1）.通过Headers反爬虫

从用户请求的Headers反爬虫是最常见的反爬虫策略。很多网站都会对Headers的User-Agent进行检测，还有一部分网站会对Referer进行检测（一些资源网站的防盗链就是检测Referer）。如果遇到了这类反爬虫机制，可以直接在爬虫中添加Headers，将浏览器的User-Agent复制到爬虫的Headers中；或者将Referer值修改为目标网站域名。对于检测Headers的反爬虫，在爬虫中修改或者添加Headers就能很好的绕过。

2）.基于用户行为反爬虫

还有一部分网站是通过检测用户行为，例如同一IP短时间内多次访问同一页面，或者同一账户短时间内多次进行相同操作。

大多数网站都是前一种情况，对于这种情况，使用IP代理就可以解决。可以专门写一个爬虫，爬取网上公开的代理ip，检测后全部保存起来。这样的代理ip爬虫经常会用到，最好自己准备一个。有了大量代理ip后可以每请求几次更换一个ip，这在requests或者urllib2中很容易做到，这样就能很容易的绕过第一种反爬虫。

对于第二种情况，可以在每次请求后随机间隔几秒再进行下一次请求。有些有逻辑漏洞的网站，可以通过请求几次，退出登录，重新登录，继续请求来绕过同一账号短时间内不能多次进行相同请求的限制。

3）.动态页面的反爬虫

上述的几种情况大多都是出现在静态页面，还有一部分网站，我们需要爬取的数据是通过ajax请求得到，或者通过JavaScript生成的。首先用Fiddler对网络请求进行分析。如果能够找到ajax请求，也能分析出具体的参数和响应的具体含义，我们就能采用上面的方法，直接利用requests或者urllib2模拟ajax请求，对响应的json进行分析得到需要的数据。

能够直接模拟ajax请求获取数据固然是极好的，但是有些网站把ajax请求的所有参数全部加密了。我们根本没办法构造自己所需要的数据的请求。这种情况下就用selenium+phantomJS，调用浏览器内核，并利用phantomJS执行js来模拟人为操作以及触发页面中的js脚本。从填写表单到点击按钮再到滚动页面，全部都可以模拟，不考虑具体的请求和响应过程，只是完完整整的把人浏览页面获取数据的过程模拟一遍。

用这套框架几乎能绕过大多数的反爬虫，因为它不是在伪装成浏览器来获取数据（上述的通过添加 Headers一定程度上就是为了伪装成浏览器），它本身就是浏览器，phantomJS就是一个没有界面的浏览器，只是操控这个浏览器的不是人。利selenium+phantomJS能干很多事情，例如识别点触式（12306）或者滑动式的验证码，对页面表单进行暴力破解等。

**165.解析网页的解析器使用最多的是哪几个?**

**166.需要登录的网页，如何解决同时限制ip，cookie,session**

## 010、爬虫过程中验证码怎么处理？

1.scrapy自带（但是成功率不高）；2.付费接口（若快http://www.ruokuai.com/home/pricetype）

**168.使用最多的数据库，对他们的理解？**

**169.编写过哪些爬虫中间件？**

**170.“极验”滑动验证码如何破解？**

**171.爬虫多久爬一次，爬下来的数据是怎么存储？**

**172.cookie过期的处理问题？**

**173.动态加载又对及时性要求很高怎么处理？**

**174.HTTPS有什么优点和缺点？**

**175.HTTPS是如何实现安全传输数据的？**

**176.TTL，MSL，RTT各是什么？**

**177.谈一谈你对Selenium和PhantomJS了解**

**178.平常怎么使用代理的 ？**

**179.存放在数据库(redis、mysql等)。**

**180.怎么监控爬虫的状态?**

**181.描述下scrapy框架运行的机制？**

答：从 start\_urls 里获取第一批 url 并发送请求，请求由引擎交给调度器入请求队列，获取完毕后，调度器将请求队列里的请求交给下载器去获取请求对应的响应资源，并将响应交给自己编写的解析方法做提取处理：1. 如果提取出需要的数据，则交给管道文件处理；2. 如果提取出url，则继续执行之前的步骤（发送url请求，并由引擎将请求交给调度器入队列...)，直到请求队列里没有请求，程序结束。

**182.谈谈你对Scrapy的理解？**

**183.怎么样让 scrapy 框架发送一个 post 请求（具体写出来）**

**184.怎么监控爬虫的状态 ？**

**185.怎么判断网站是否更新？**

**186.图片、视频爬取怎么绕过防盗连接**

**187.你爬出来的数据量大概有多大？大概多长时间爬一次？**

**188.用什么数据库存爬下来的数据？部署是你做的吗？怎么部署？**

**189.增量爬取**

**190.爬取下来的数据如何去重，说一下scrapy的具体的算法依据。**

**191.Scrapy的优缺点?**

**192.怎么设置爬取深度？**

**193.scrapy和scrapy-redis有什么区别？**

　答：scrapy 是一个 Python 爬虫框架，爬取效率极高，具有高度定制性，但是不支持分布式。而 scrapy-redis 一套基于 redis [数据库](https://www.boxuegu.com/news/494.html)、运行在 scrapy 框架之上的组件，可以让scrapy 支持分布式策略，Slaver 端共享 Master 端 redis 数据库里的 item 队列、请求队列和请求指纹集合。

**194.为什么选择redis数据库？**

因为redis支持主从同步，而且数据都是缓存在内存中的，所以基于redis的分布式爬虫，对请求和数据的高频读取效率非常高。

## 194.分布式爬虫主要解决什么问题？

1) ip

2) 带宽

3）cpu

4）io

**195.什么是分布式存储？**

**196.你所知道的分布式爬虫方案有哪些？**

**197.scrapy-redis，有做过其他的分布式爬虫吗？**

## 001、python爬虫scrapy项目（一）

### 爬取目标：腾讯招聘网站（起始url：https://hr.tencent.com/position.php?keywords=&tid=0&start）

### 爬取内容：职位；职位类型；招聘人数；工作地点；发布时间；招聘详细链接；工作职责；工作要求

### 反反爬措施：设置随机user-agent、设置请求延时操作、

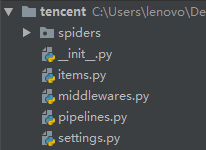
#### 1、开始创建项目

1 scrapy startproject tencent

#### 2、进入tencent文件夹，执行启动spider爬虫文件代码，编写爬虫文件。

1 scrapy genspider hr "tencent.com"

　　命令执行完，用Python最好的IDE---pycharm打开该文件目录，会在你的当前目录创建如下文件目录。



#### 3、编写该目录下的items.py文件，设置你需要爬取的字段。

1 class TencentItem(scrapy.Item):

2 # define the fields for your item here like:

3 # 职位

4 position = scrapy.Field()

5 # 职位类型

6 position\_type = scrapy.Field()

7 # 招聘人数

8 persons = scrapy.Field()

9 # 工作地点

10 place = scrapy.Field()

11 # 招聘发布时间

12 time = scrapy.Field()

13 # 职位详细链接

14 detail\_link = scrapy.Field()

15 # 工作职责

16 work\_duty = scrapy.Field()

17 # 工作要求

18 work\_request = scrapy.Field()

#### 4、进入spiders文件夹，打开hr.py文件,开始编写爬虫文件

1 # -\*- coding: utf-8 -\*-

2 import scrapy

3 import re

4 from items import TencentItem

5

6 class HrSpider(scrapy.Spider):

7 name = 'hr'

8 allowed\_domains = ['tencent.com']

9 offset = 0

10 original\_url = 'https://hr.tencent.com/position.php?keywords=&tid=0&start='

11 # 设置动态起始url

12 start\_urls = ['https://hr.tencent.com/position.php?keywords=&tid=0&start=' + str(offset)]

13

14 def parse(self, response):

15 # 编写xpath规则提取需要的数据，进行数据清洗。

16 trs = response.xpath("//table[@class='tablelist']//tr")[1:-1]

17 for tr in trs:

18 item = TencentItem()

19 item["position"] = tr.xpath("./td[1]/a/text()").extract()

20 item["position\_type"] = tr.xpath("./td[2]/text()").extract()

21 item["persons"] = tr.xpath("./td[3]/text()").extract()

22 item["place"] = tr.xpath("./td[4]/text()").extract()

23 item["time"] = tr.xpath("./td[5]/text()").extract()

24 link\_part = tr.xpath("./td[1]/a/@href").extract\_first()

25 # 分析网址结构，拼接正确的职位详细链接

26 url\_detail = item["detail\_link"] = 'https://hr.tencent.com/' + link\_part

27 # 将找到的详细链接yield 到scrapy的调度器，调度器进行入队列，依次发送请求。

28 yield scrapy.Request(url=url\_detail,

29 callback=self.parse\_next\_url,#编写处理链接的回调函数

30 meta = {"item":item},

31 )

32 # 进行翻页操作

33 if self.offset < 2870:

34 self.offset += 10

35 url\_send = self.original\_url + str(self.offset)

36 yield scrapy.Request(

37 url=url\_send,

38 callback=self.parse,

39 )

40 # 编写回调函数

41 def parse\_next\_url(self,response):

42 item = response.meta["item"]

43 item["work\_duty"] = response.xpath("//table[@class='tablelist textl']//tr[3]//ul//text()").extract()

44 item["work\_request"] = response.xpath("//table[@class='tablelist textl']//tr[4]//ul//text()").extract()

45 item["work\_duty"] = re.sub(r'(\xa0)','',str(item["work\_duty"]))

46 item["work\_request"] = re.sub(r'(\xa0)','',str(item["work\_request"]))

47 yield item

#### 5、编写pipeline.py文件，处理接收到的数据

1 import json

2

3

4 class TencentPipeline(object):

5 # 自定义一个打开文件，写入文件的方式存储数据

6 def \_\_init\_\_(self):

7 self.f = open("tencent.json","wb")

8

9 def process\_item(self, item, spider):

10 # 当item文件中有中文时，ensure默认是用ascii编码中文

11 content = json.dumps(dict(item),ensure\_ascii= False) + ", \n"

12 self.f.write(content.encode("utf-8"))

13 return item

14

15 def close\_file(self):

16 self.f.close()

#### 6、设置setting.py文件，配置scrapy运行的相关内容

DOWNLOADER\_MIDDLEWARES = {

'tencent.middlewares.RandomUA': 543,

}

ITEM\_PIPELINES = {

'tencent.pipelines.TencentPipeline': 300,

# 'scrapy\_redis.pipelines.RedisPipeline': 400,

}

USER\_AGENT = [

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/22.0.1207.1 Safari/537.1",

"Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1092.0 Safari/536.6",

"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6",

"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/19.77.34.5 Safari/537.1",

"Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.9 Safari/536.5",

"Mozilla/5.0 (Windows NT 6.0) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",

"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",

"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; SE 2.X MetaSr 1.0; SE 2.X MetaSr 1.0; .NET CLR 2.0.50727; SE 2.X MetaSr 1.0)",

"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",

"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; 360SE)",

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",

"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.0 Safari/536.3",

"Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/535.24 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24",

"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/535.24 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24"

]

# 设置请求延时操作

DOWNLOAD\_DELAY = 1

7、设置middlewares.py文件，对请求进行处理。

class RandomUA(object):

# 设置随机请求头

def process\_request(self,request,spider):

UA = random.choice(USER\_AGENT)

request.headers["user-agent"] = UA

8、设置爬虫的启动文件start.py

1 from scrapy.cmdline import execute

2 execute("scrapy crawl hr".split())

9、执行效果如下。（保存为json数据格式的字符串到本地）

1 {"position": ["25928-高级图形开发工程师（深圳总部）"], "position\_type": ["技术类"], "persons": ["3"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46479&keywords=&tid=0&lid=0", "work\_duty": "['负责游戏引擎图形相关特性的开发；', '负责渲染流程和算法的优化，以及相关工具的开发；', '负责图形兼容性分析以及疑难问题的分析定位工作。']", "work\_request": "['本科以上学历，精通C/C++，具备扎实的数据结构和算法基础，熟悉常用设计模式；', '具备计算机图形学知识，熟练掌握3D图形渲染技术，熟悉OpenGL以及Shader开发；', '熟练掌握3D游戏引擎架构， 熟悉3D引擎的接口和游戏制作流程；', '3年以上3D引擎（Unreal、Unity等）开发经验，一年以上渲染相关开发和优化经验；', '深刻理解客户端框架和其他核心模块的实现，有主导过核心模块的开发经验者优先；', '熟悉移动端GPU/CPU架构，有移动端渲染开发经验者优先；', '责任心强，善于沟通，对游戏前沿技术应用抱有热情。']"},

2 {"position": ["25667-渠道销售经理（深圳）"], "position\_type": ["市场类"], "persons": ["2"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46485&keywords=&tid=0&lid=0", "work\_duty": "['担任腾讯云渠道经理，负责区域渠道体系建设及产品销售；', '定期拜访渠道合作伙伴，充分了解客户需求并积极跟进，制定合理方案，负责方案提示、谈判，追踪公司相关部门的工作，保证方案的有效实施；', '维持与现有合作伙伴的良好业务关系，及时更新公司产品信息，传达企业及品牌文化。']", "work\_request": "['本科及以上学历，计算机、电信、市场营销或其它相关专业；', '软件或互联网行业五年以上相关工作经验；', '具有丰富的渠道销售、区域管理及长尾中小企业客户覆盖经验；', '具有企业级应用软件销售经验，具有云计算及互联网行业渠道销售经验优先；', '能够有效通过渠道覆盖中长尾客户，承担区域销售业绩；', '能够建立区域渠道体系，有效处理渠道冲突与风险防范；', '能够主导制定各种服务与激励方式，持续提高渠道合作伙伴的满意度；', '具有出色的协调能力，良好的团队合作精神；为人诚信，工作敬业，有责任心。']"},

3 {"position": ["28481-医疗健康UI开发工程师(深圳)"], "position\_type": ["设计类"], "persons": ["1"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46476&keywords=&tid=0&lid=0", "work\_duty": "['负责腾讯觅影，智慧医院等相关医疗产品的前端组件的编写，web开发工作；', '根据产品与设计要求，不断优化前端架构，改善用户体验。参与相关UI组件体系的建立、维护等。']", "work\_request": "['网页重构或web前端开发工作2年以上； ', '精通HTML5，CSS3，JavaScript构建高性能web应用；掌握React或Vue并有相关实战经验，掌握主流前端构建工具grunt，gulp，webpack；', '精通UI组件化开发、动效开发、响应式、多终端适配、无障碍有一定开发经验；', '有node.js/vue/react开发经验者优先，有前端性能、工具研发方面的实践经验优先。', '对Web性能、安全相关有一定的了解； ', '有创新精神并能积极学习业界新技术，顺畅的沟通合作能力。']"},

4 {"position": ["SA-腾讯社交广告高级系统测试工程师（研发中心 北京）"], "position\_type": ["技术类"], "persons": ["1"], "place": ["北京"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46486&keywords=&tid=0&lid=0", "work\_duty": "['参与互联网软件产品测试的全流程，包括参与需求分析、设计评审，制定测试计划，设计和执行测试用例，进行缺陷跟踪和软件质量分析等；', '制定测试计划，构建测试环境，执行集成测试，回归测试等； ', '保证被测系统的质量，并通过测试流程和方法创新，努力提升研发的质量和效率。']", "work\_request": "['工科、计算机或其他相关专业本科以上学历；', '熟悉C/C++/Java等至少一种编程语言,有Shell或Ruby/PHP/Perl/Python等使用经验者优先；', '至少1年以上软件开发、自动化测试工作经验；', '有性能、安全、白盒测试等专业测试领域经验者优先；', '具备互联网广告、搜索、大数据处理、分布式系统、数据库和网络等业务领域测试经验者优先; ', '熟悉Linux或Unix操作系统；', '精通测试流程和测试用例设计方法,能主动进行技术钻研；', '解决复杂问题和编写自动测试工具和系统的能力；', '很强的逻辑思维能力,谈判的能力和冲突管理的能力；', '善于团队合作,理解和适应变化,以结果和行动为准则,努力追求成功。']"},

5 {"position": ["25664-政府行业交付项目经理"], "position\_type": ["产品/项目类"], "persons": ["2"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46484&keywords=&tid=0&lid=0", "work\_duty": "['1、负责腾讯云政府行业的项目交付管理工作；', '2、负责项目资源的组织与协调，确保项目团队各干系人及内外部合作团队的协同工作； ', '3、负责项目计划的制定、跟踪与维护，确保项目按计划完成，并解决交付中的各类问题；', '4、协助收集客户需求和用户反馈，驱动研发团队完善产品，确保项目顺利通过验收。']", "work\_request": "['1、全日制统招本科及以上学历，5年以上政府行业经验，至少深入参与5个政府行业大中型项目；', '2、有在大型企业工作的经历，管理过20人以上的项目团队，有丰富的跨部门、跨组织沟通协调经验，能够应对复杂的项目环境；', '3、熟悉研发过程，包括产品设计、需求分析、架构设计、开发、测试、运维等，熟悉敏捷开发过程；', '4、有出色的沟通能力和技巧，能够想方设法推动项目的顺利进行，有强烈的结果导向意识；', '5、具备良好的项目管理、客户关系维护能力，和优秀的沟通技巧，能妥善协调好客户、合作伙伴、内部团队的合作关系；', '6、有非常强的事业心、责任感和担当精神，有较强的抗压能力，能并行处理多个项目工作，能承受一定程度的出差或驻场工作；', '7、有PMP、ITIL证书者优先，信息产业部系统集成项目经理证书者优先。']"},

6 {"position": ["PCG14-应用宝数据挖掘算法工程师（深圳）"], "position\_type": ["技术类"], "persons": ["1"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46483&keywords=&tid=0&lid=0", "work\_duty": "['负责提供合适的推荐算法模型；', '负责研究业内领先的技术，结合腾讯各个业务平台的数据，根据应用中心和社交渠道这两个场景给出具体的实验数据，并且评估结果；', '负责根据不同的算法模型，上报业务需要的统计数据，协助各种算法的实施；', '研究已有算法的瓶颈，提出合理的改进措施和解决方案。']", "work\_request": "['计算机、应用数学、人工智能、模式识别、统计、自控等专业的硕士或者博士优先；', '2年以上相关工作经验；', '对机器学习、数据挖掘算法及其在互联网上的应用有比较深入的理解；', '熟悉掌握C/C++语言；', '有大规模分布式计算平台的使用和并行算法开发经验；', '严密的数学思维、突出的分析和归纳能力、优秀的沟通表达能力；', '有互联网广告，电商，搜索等方面推荐经验优先。']"},

7 {"position": ["19867-游戏后台开发工程师（深圳）"], "position\_type": ["技术类"], "persons": ["1"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46482&keywords=&tid=0&lid=0", "work\_duty": "['负责游戏后台架构设计；', '负责游戏后台系统模块以及新特性开发；', '负责服务器性能优化和体验优化。']", "work\_request": "['2年以上游戏服务器后台工作经验，有完整的项目经验；', '扎实的编程基础，对高在线大并发游戏后台架构有一定认识；', '熟悉Unix/Linux操作系统下的C/C++开发；', '熟悉TCP/IP协议相关知识，熟悉网络编程，熟悉数据库；', '了解游戏服务器架构及性能优化方法；', '具备良好的分析解决问题能力，能独立承担后台逻辑系统开发工作；', '高度的责任心、良好的沟通能力和团队合作精神。']"},

8 {"position": ["TME-全民K歌项目经理（深圳）"], "position\_type": ["产品/项目类"], "persons": ["1"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46478&keywords=&tid=0&lid=0", "work\_duty": "['负责全民K歌版本计划制定，风险监控，过程跟踪，保障版本目标的实现；', '负责FT内目标确认，目标拆解，资源分配以及目标达成情况跟进，推动各角色协同工作；', '发现、总结、跟踪过程问题，推动团队各环节持续改进，提高效率。']", "work\_request": "['本科以上学历，计算机或相关专业；', '3年以上软件项目管理经验，有互联网、软件领域技术开发经验优先；', '精通软件项目过程管理，对敏捷项目管理有实际应用经验及深刻理解；', '具有良好的执行力和责任心，能推动项目团队朝目标前进；', '具有丰富的与人沟通、交流和组织能力，出色的团队合作精神；']"},

9 {"position": ["25928-高级语音算法工程师（上海）"], "position\_type": ["技术类"], "persons": ["1"], "place": ["上海"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46477&keywords=&tid=0&lid=0", "work\_duty": "['负责游戏语音算法优化；', '负责语音前沿技术研究；', '负责游戏语音现网版本算法维护升级。']", "work\_request": "['本科及以上学历；', '熟悉PC/Android/iOS SDK任一平台C/C++开发，性能优化； ', '熟悉数字信号处理，数学功底扎实，熟悉MATLAB仿真； ', '熟悉语音前处理算法AEC,AGC,VAD,NS,CNG,JitterBuffer,Mix等算法；', '熟悉常见Codec,Opus/AAC/Speex等;', '有AI语音前处理经验优先；', '熟悉WebRTC,Speex,Opus等开源代码。']"},

10 {"position": ["HY-游戏发行/运营培训生（深圳）"], "position\_type": ["产品/项目类"], "persons": ["1"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46481&keywords=&tid=0&lid=0", "work\_duty": "['游戏发行/运营培训生项目致力于培养高潜力的游戏运营人才，以满足快速增长游戏业务需求；', '项目采用定制化的培养方式，通过专班学习，名师辅导和项目实战，提升学员的产品sense、运营能力及通用素质，使学员成为优秀的游戏发行/运营人才。', '协助项目制作人与研发商共同制定运营目标和工作计划，约定的各阶段游戏优化、运营开发和运营支持工作；', '推动游戏研发商的日常沟通，密切关注研发和运营筹备进度同时提供必要协助；', '指导并支持项目组内不同职能员工的日常工作，推动合作部门的目标和工作计划制定；', '根据项目需求，制定并推广项目流程规范，确保项目有序推进；', '及时发现并跟踪解决项目问题，有效管理项目风险。']", "work\_request": "['热爱并乐于体验游戏，对研发与运营有一定的了解，保持强烈的好奇心和求知欲；', '优秀的中英文读写能力；', '积极主动，能够承受高压的工作。']"},

11 {"position": ["25928-高级图形开发工程师（上海）"], "position\_type": ["技术类"], "persons": ["3"], "place": ["上海"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46480&keywords=&tid=0&lid=0", "work\_duty": "['负责游戏引擎图形相关特性的开发；', '负责渲染流程和算法的优化，以及相关工具的开发；', '负责图形兼容性分析以及疑难问题的分析定位工作。']", "work\_request": "['本科以上学历，精通C/C++，具备扎实的数据结构和算法基础，熟悉常用设计模式；', '具备计算机图形学知识，熟练掌握3D图形渲染技术，熟悉OpenGL以及Shader开发；', '熟练掌握3D游戏引擎架构， 熟悉3D引擎的接口和游戏制作流程；', '3年以上3D引擎（Unreal、Unity等）开发经验，一年以上渲染相关开发和优化经验；', '深刻理解客户端框架和其他核心模块的实现，有主导过核心模块的开发经验者优先；', '熟悉移动端GPU/CPU架构，有移动端渲染开发经验者优先；', '责任心强，善于沟通，对游戏前沿技术应用抱有热情。']"},

12 {"position": ["28481-健康保险行业合作高级经理（深圳）"], "position\_type": ["市场类"], "persons": ["1"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46474&keywords=&tid=0&lid=0", "work\_duty": "['1、负责商业保险行业客户资源拓展（包括不限于保险公司、创新保险平台、行业协会等专业领域）；', '2、拓展相关行业合作伙伴以及合作机构，整合公司已有产品和资源，形成场景化的创新解决方案；', '3、搜集整理健康保险行业的市场动态、政策变动等行业信息，解读反馈助推业务策略制定；', '4、整合资源，设计制定并推进商业合作方案落地，有效撬动行业资源合作。']", "work\_request": "['1、全日制本科及以上学历，三年以上健康保险领域工作经验；', '2、熟悉保险行业，有健康险创新产品运营经验或创新平台运营经验； ', '3、具备良好的沟通表达能力，清晰的思维逻辑，敏锐的洞察力，较强的自驱力和执行力； ', '4、对工作有高度的责任心和激情，注重团队合作，适应频繁差旅需求。']"},

13 {"position": ["25928-前端测试开发工程师（深圳）"], "position\_type": ["技术类"], "persons": ["1"], "place": ["深圳"], "time": ["2018-12-19"], "detail\_link": "https://hr.tencent.com/position\_detail.php?id=46473&keywords=&tid=0&lid=0", "work\_duty": "['负责平台类软件的测试开发工作；', '负责平台组件的接口测试、单元测试工作；', '能够在关键技术上给予团队技术指引和支持；', '按时完成安排的移动端开发任务；', '负责与项目组之间的协调，推动工作，帮助项目组推动整个项目质量的提升。']", "work\_request": "['本科及以上学历，计算机相关专业，开发或者测试开发出身，有软件开发的基础；', '2-3年以上软件行业或者互联网行业经验，熟悉Windows编程或Android/iOS编程；', '熟悉软件开发流程，熟悉Android/iOS环境下自动化测试技术；', '扎实的测试用例设计能力，熟悉主流自动化方法；', '具备扎实的C++、C或Object-C程序设计基础。', '有以下任意一项经验者优先：', '较强的DEBUG能力；', '有Android、iOS产品自动化测试经验。']"},

......

[返回主页](https://www.cnblogs.com/xuechaojun/)

## 002、python爬虫+数据可视化项目（一）

### 爬取目标：中国天气网（起始url：http://www.weather.com.cn/textFC/hb.shtml#）

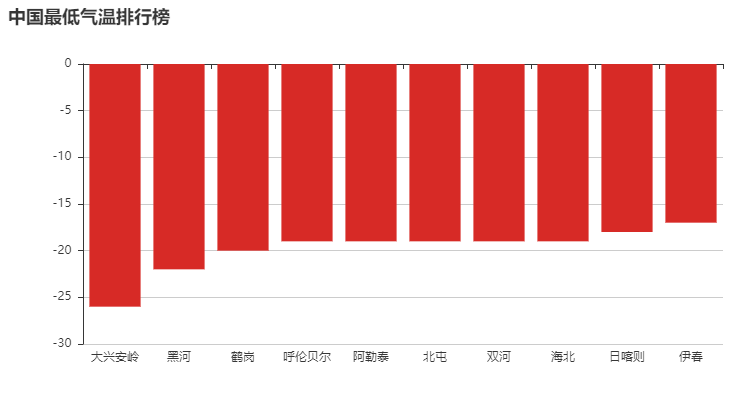
### 爬取内容：全国实时温度最低的十个城市气温排行榜

### 使用工具：requests库实现发送请求、获取响应。

### beautifulsoup实现数据解析、提取和清洗

### pyechart模块实现数据可视化

### 爬取结果：柱状图可视化展示：



## 直接放代码（详细说明在注释里，欢迎同行相互交流、学习~）：

1 import requests

2 from bs4 import BeautifulSoup

3 from pyecharts import Bar

4

5 ALL\_DATA = []

6 def send\_parse\_urls(start\_urls):

7 headers = {

8 "User-Agent": "Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)"

9 }

10 for start\_url in start\_urls:

11 response = requests.get(start\_url,headers=headers)

12 # 编码问题的解决

13 response = response.text.encode("raw\_unicode\_escape").decode("utf-8")

14 soup = BeautifulSoup(response,"html5lib") #lxml解析器：性能比较好，html5lib：适合页面结构比较混乱的

15 div\_tatall = soup.find("div",class\_="conMidtab") #find() 找符合要求的第一个元素

16 tables = div\_tatall.find\_all("table") #find\_all() 找到符合要求的所有元素的列表

17 for table in tables:

18 trs = table.find\_all("tr")

19 info\_trs = trs[2:]

20 for index,info\_tr in enumerate(info\_trs): # 枚举函数，可以获得索引

21 # print(index,info\_tr)

22 # print("="\*30)

23 city\_td = info\_tr.find\_all("td")[0]

24 temp\_td = info\_tr.find\_all("td")[6]

25 # if的判断的index的特殊情况应该在一般情况的后面，把之前的数据覆盖

26 if index==0:

27 city\_td = info\_tr.find\_all("td")[1]

28 temp\_td = info\_tr.find\_all("td")[7]

29 city=list(city\_td.stripped\_strings)[0]

30 temp=list(temp\_td.stripped\_strings)[0]

31 ALL\_DATA.append({"city":city,"temp":temp})

32 return ALL\_DATA

33

34 def get\_start\_urls():

35 start\_urls = [

36 "http://www.weather.com.cn/textFC/hb.shtml",

37 "http://www.weather.com.cn/textFC/db.shtml",

38 "http://www.weather.com.cn/textFC/hd.shtml",

39 "http://www.weather.com.cn/textFC/hz.shtml",

40 "http://www.weather.com.cn/textFC/hn.shtml",

41 "http://www.weather.com.cn/textFC/xb.shtml",

42 "http://www.weather.com.cn/textFC/xn.shtml",

43 "http://www.weather.com.cn/textFC/gat.shtml",

44 ]

45 return start\_urls

46

47 def main():

48 """

49 主程序逻辑

50 展示全国实时温度最低的十个城市气温排行榜的柱状图

51 """

52 # 1 获取所有起始url

53 start\_urls = get\_start\_urls()

54 # 2 发送请求获取响应、解析页面

55 data = send\_parse\_urls(start\_urls)

56 # print(data)

57 # 4 数据可视化

58 #1排序

59 data.sort(key=lambda data:int(data["temp"]))

60 #2切片，选择出温度最低的十个城市和温度值

61 show\_data = data[:10]

62 #3分出城市和温度

63 city = list(map(lambda data:data["city"],show\_data))

64 temp = list(map(lambda data:int(data["temp"]),show\_data))

65 #4创建柱状图、生成目标图

66 chart = Bar("中国最低气温排行榜") #需要安装pyechart模块

67 chart.add("",city,temp)

68 chart.render("tempture.html")

69

70 if \_\_name\_\_ == '\_\_main\_\_':

71 main()

[返回主页](https://www.cnblogs.com/xuechaojun/)

## 003、[python多线程爬虫+批量下载斗图啦图片项目（关注、持续更新）](https://www.cnblogs.com/xuechaojun/p/10158873.html)

## python多线程爬虫项目（）

### 爬取目标：斗图啦（起始url：http://www.doutula.com/photo/list/?page=1）

### 爬取内容：斗图啦全网图片

### 使用工具：requests库实现发送请求、获取响应。

### xpath实现数据解析、提取和清洗

### threading模块实现多线程爬虫

### 爬取结果：

### 

### 思路：由于该爬虫存在网络密集IO和磁盘密集IO，存在大量等待时间，遂采用多线程方式爬取。

### 设计：本文采用多为结构化代码的面向对象封装设计思路，使用生产消费者模型，完成多线程的调度、爬取。

## 直接放代码（详细说明在注释里，欢迎同行相互交流、学习~）：

1 import os

2 import threading

3 import re

4 from queue import Queue

5 import requests

6 from urllib import request

7 from lxml import etree

8

9 # 定义一个全局变量，存储请求头headers数据

10 headers = {

11 "User-Agent": "Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)"

12 }

13

14 class Producter(threading.Thread):

15 """

16 生产者模型：负责从起始url队列中提取url，进行解析，将得到的图片地址放入img图片队列中

17 """

18 def \_\_init\_\_(self, page\_queue, img\_queue, \*args, \*\*kwargs):

19 # 改写父类threading.Thread的\_\_init\_\_方法，添加默认值

20 super(Producter, self).\_\_init\_\_(\*args, \*\*kwargs)

21 # 添加对象属性

22 self.page\_queue = page\_queue

23 self.img\_queue = img\_queue

24

25 def run(self):

26 """

27 实现消费者模型的主要业务逻辑

28 """

29 while True:

30 # 当请求队列为空，生产者停止生产

31 if self.page\_queue.empty():

32 break

33 # 获取起始url队列的对象，进行页面解析

34 url = self.page\_queue.get()

35 self.parse\_url(url)

36

37 def parse\_url(self, url):

38 """

39 实现解析指定页面的功能

40 :param url: 传入待处理的页面url

41 """

42 response = requests.get(url=url, headers=headers)

43 html = etree.HTML(response.text)

44 # 使用lxml库里HTML解析器进行数据解析，利用xpath语法解析得到指定数据，返回一个element对象列表

45 url\_gifs = html.xpath("//div[@class='page-content text-center']//img[@class!='gif']")

46 for url\_gif in url\_gifs:

47 # element.get(属性名)可以获取属性值

48 url\_name = url\_gif.get("alt")

49 # 正则表达式替换非法字符

50 url\_name = re.sub(r"[\!！\.\?？]", "", url\_name).strip()

51 url\_link = url\_gif.get("data-original")

52 # os模块中os.path.splitext()可以获取url的后缀名

53 url\_suffix = os.path.splitext(url\_link)[1]

54 filename = url\_name + url\_suffix

55 # 队列的put（）里面传的是元组或者列表

56 self.img\_queue.put((url\_link, filename))

57

58 class Consumer(threading.Thread):

59 """

60 消费者模型的主要业务逻辑

61 """

62

63 def \_\_init\_\_(self, page\_queue, img\_queue, \*args, \*\*kwargs):

64 super(Consumer, self).\_\_init\_\_(\*args, \*\*kwargs)

65 self.page\_queue = page\_queue

66 self.img\_queue = img\_queue

67

68 def run(self):

69 """

70 实现读取图片url内容的功能

71 """

72 while True:

73 if self.page\_queue.empty() and self.img\_queue.empty():

74 break

75 url, filename = self.img\_queue.get()

76 # urllib库里面的request模块可以读取图片url内容

77 request.urlretrieve(url, "GIF/" + filename)

78 # 控制台输出提示信息

79 print(filename + "-------下载完成！")

80

81 def main():

82 # 创建page队列，存放请求的起始url;创建img队列，存放图片data的url

83 page\_queue = Queue(100) # 设置队列的最大存储数量

84 img\_queue = Queue(1000) # 设置队列的最大存储数量

85 for i in range(100):

86 start\_url\_format = "http://www.doutula.com/photo/list/?page={}".format(i)

87 # print(start\_url\_format) #调试代码用

88 page\_queue.put(start\_url\_format) #将获取的起始url放入队列中

89 # 生成多线程对象（多个生产者、消费者）。实现发送请求，获取响应，解析页面，获取数据

90 for i in range(10):

91 t = Producter(page\_queue, img\_queue)

92 t.start()

93 for i in range(10):

94 t = Consumer(page\_queue, img\_queue)

95 t.start()

96

97 if \_\_name\_\_ == '\_\_main\_\_':

98 main()

[返回主页](https://www.cnblogs.com/xuechaojun/)

## 004、[python爬虫项目(scrapy-redis分布式爬取房天下租房信息)](https://www.cnblogs.com/xuechaojun/p/10164939.html)

## python爬虫scrapy项目（二）

### 爬取目标：房天下全国租房信息网站（起始url：http://zu.fang.com/cities.aspx）

### 爬取内容：城市；名字；出租方式；价格；户型；面积；地址；交通

### 反反爬措施：设置随机user-agent、设置请求延时操作、

#### 1、开始创建项目

1 scrapy startproject fang

#### 2、进入fang文件夹，执行启动spider爬虫文件代码，编写爬虫文件。

1 scrapy genspider zufang "zu.fang.com"

　　命令执行完，用Python最好的IDE---pycharm打开该文件目录

#### 3、编写该目录下的items.py文件，设置你需要爬取的字段。

1 import scrapy

2

3

4 class HomeproItem(scrapy.Item):

5 # define the fields for your item here like:

6 # name = scrapy.Field()

7

8 city = scrapy.Field() #城市

9 title = scrapy.Field() # 名字

10 rentway = scrapy.Field() # 出租方式

11 price = scrapy.Field() #价格

12 housetype = scrapy.Field() # 户型

13 area = scrapy.Field() # 面积

14 address = scrapy.Field() # 地址

15 traffic = scrapy.Field() # 交通

#### 4、进入spiders文件夹，打开hr.py文件,开始编写爬虫文件

1 # -\*- coding: utf-8 -\*-

2 import scrapy

3 from homepro.items import HomeproItem

4 from scrapy\_redis.spiders import RedisCrawlSpider

5 # scrapy.Spider

6 class HomeSpider(RedisCrawlSpider):

7 name = 'home'

8 allowed\_domains = ['zu.fang.com']

9 # start\_urls = ['http://zu.fang.com/cities.aspx']

10

11 redis\_key = 'homespider:start\_urls'

12 def parse(self, response):

13 hrefs = response.xpath('//div[@class="onCont"]/ul/li/a/@href').extract()

14 for href in hrefs:

15 href = 'http:'+ href

16 yield scrapy.Request(url=href,callback=self.parse\_city,dont\_filter=True)

17

18

19 def parse\_city(self, response):

20 page\_num = response.xpath('//div[@id="rentid\_D10\_01"]/span[@class="txt"]/text()').extract()[0].strip('共页')

21 # print('\*' \* 100)

22 # print(page\_num)

23 # print(response.url)

24

25 for page in range(1, int(page\_num)):

26 if page == 1:

27 url = response.url

28 else:

29 url = response.url + 'house/i%d' % (page + 30)

30 print('\*' \* 100)

31 print(url)

32 yield scrapy.Request(url=url, callback=self.parse\_houseinfo, dont\_filter=True)

33

34 def parse\_houseinfo(self, response):

35 divs = response.xpath('//dd[@class="info rel"]')

36 for info in divs:

37 city = info.xpath('//div[@class="guide rel"]/a[2]/text()').extract()[0].rstrip("租房")

38 title = info.xpath('.//p[@class="title"]/a/text()').extract()[0]

39 rentway = info.xpath('.//p[@class="font15 mt12 bold"]/text()')[0].extract().replace(" ", '').lstrip('\r\n')

40 housetype = info.xpath('.//p[@class="font15 mt12 bold"]/text()')[1].extract().replace(" ", '')

41 area = info.xpath('.//p[@class="font15 mt12 bold"]/text()')[2].extract().replace(" ", '')

42 addresses = info.xpath('.//p[@class ="gray6 mt12"]//span/text()').extract()

43 address = '-'.join(i for i in addresses)

44 try:

45 des = info.xpath('.//p[@class ="mt12"]//span/text()').extract()

46 traffic = '-'.join(i for i in des)

47 except Exception as e:

48 traffic = "暂无详细信息"

49

50 p\_name = info.xpath('.//div[@class ="moreInfo"]/p/text()').extract()[0]

51 p\_price = info.xpath('.//div[@class ="moreInfo"]/p/span/text()').extract()[0]

52 price = p\_price + p\_name

53

54 item = HomeproItem()

55 item['city'] = city

56 item['title'] = title

57 item['rentway'] = rentway

58 item['price'] = price

59 item['housetype'] = housetype

60 item['area'] = area

61 item['address'] = address

62 item['traffic'] = traffic

63 yield item

#### 5、设置setting.py文件，配置scrapy运行的相关内容

1 # 指定使用scrapy-redis的调度器

2 SCHEDULER = "scrapy\_redis.scheduler.Scheduler"

3

4 # 指定使用scrapy-redis的去重

5 DUPEFILTER\_CLASS = 'scrapy\_redis.dupefilter.RFPDupeFilter'

6

7 # 指定排序爬取地址时使用的队列，

8 # 默认的 按优先级排序(Scrapy默认)，由sorted set实现的一种非FIFO、LIFO方式。

9 SCHEDULER\_QUEUE\_CLASS = 'scrapy\_redis.queue.SpiderPriorityQueue'

10

11 REDIS\_HOST = '10.8.153.73'

12 REDIS\_PORT = 6379

13 # 是否在关闭时候保留原来的调度器和去重记录，True=保留，False=清空

14 SCHEDULER\_PERSIST = True

#### 6、然后把代码发给其他附属机器,分别启动.子程序redis链接主服务器redis。

1 redis-cli -h 主服务器ip

#### 7、主服务器先启动redis-server，再启动redis-cli

1 lpush homespider:start\_urls 起始的url

## 188、你常用的mysql引擎有哪些？各引擎间有什么区别？

主要 MyISAM 与 InnoDB 两个引擎，其主要区别如下：

一、InnoDB 支持事务，MyISAM 不支持，这一点是非常之重要。事务是一种高

级的处理方式，如在一些列增删改中只要哪个出错还可以回滚还原，而 MyISAM

就不可以了；

二、MyISAM 适合查询以及插入为主的应用，InnoDB 适合频繁修改以及涉及到

安全性较高的应用；

三、InnoDB 支持外键，MyISAM 不支持；

四、MyISAM 是默认引擎，InnoDB 需要指定；

五、InnoDB 不支持 FULLTEXT 类型的索引；

六、InnoDB 中不保存表的行数，如 select count(\*) from table 时，InnoDB；需要

扫描一遍整个表来计算有多少行，但是 MyISAM 只要简单的读出保存好的行数即

可。注意的是，当 count(\*)语句包含 where 条件时 MyISAM 也需要扫描整个表；

七、对于自增长的字段，InnoDB 中必须包含只有该字段的索引，但是在 MyISAM

表中可以和其他字段一起建立联合索引；

八、清空整个表时，InnoDB 是一行一行的删除，效率非常慢。MyISAM 则会重

建表；

九、InnoDB 支持行锁（某些情况下还是锁整表，如 update table set a=1 where

user like '%lee%'

## 189、什么是关联查询，有哪些？

将多个表联合起来进行查询，主要有内连接、左连接、右连接、全连接（外连接）

## 190、数据库的优化？

1. 优化索引、SQL 语句、分析慢查询；

2. 设计表的时候严格根据数据库的设计范式来设计数据库；

3. 使用缓存，把经常访问到的数据而且不需要经常变化的数据放在缓存中，能

节约磁盘IO；

4. 优化硬件；采用SSD，使用磁盘队列技术(RAID0,RAID1,RDID5)等；

5. 采用MySQL 内部自带的表分区技术，把数据分层不同的文件，能够提高磁

盘的读取效率；

6. 垂直分表；把一些不经常读的数据放在一张表里，节约磁盘I/O；

7. 主从分离读写；采用主从复制把数据库的读操作和写入操作分离开来；

8. 分库分表分机器（数据量特别大），主要的的原理就是数据路由；

9. 选择合适的表引擎，参数上的优化；

10. 进行架构级别的缓存，静态化和分布式；

11. 不采用全文索引；

12. 采用更快的存储方式，例如 NoSQL存储经常访问的数据

## 191、简述 RabbitMQ、Kafka、ZeroMQ的区别？

https://blog.csdn.net/zhailihua/article/details/7899006

## 192、[RabbitMQ](http://www.cnblogs.com/yuanchenqi/articles/8507109.html)如何在消费者获取任务后未处理完前就挂掉时，保证数据不丢失？

为了预防消息丢失，rabbitmq提供了ack

即工作进程在收到消息并处理后，发送ack给rabbitmq，告知rabbitmq这时候可以把该消息从队列中删除了。

如果工作进程挂掉 了，rabbitmq没有收到ack，那么会把该消息 重新分发给其他工作进程。

不需要设置timeout，即使该任务需要很长时间也可以处理。

ack默认是开启的，工作进程显示指定了no\_ack=True

## 193、RabbitMQ如何对消息做持久化？

1、创建队列和发送消息时将设置durable=Ture，如果在接收到消息还没有存储时，消息也有可能丢失，就必须配置publisher confirm

channel.queue\_declare(queue='task\_queue', durable=True)

2、返回一个ack，进程收到消息并处理完任务后，发给rabbitmq一个ack表示任务已经完成，可以删除该任务

3、镜像队列：将queue镜像到cluster中其他的节点之上。

在该实现下，如果集群中的一个节点失效了，queue能自动地切换到镜像中的另一个节点以保证服务的可用性

## 194、RabbitMQ如何控制消息被消费的顺序？

默认消息队列里的数据是按照顺序被消费者拿走，

例如：消费者1 去队列中获取奇数序列的任务，消费者2去队列中获取偶数序列的任务。

channel.basic\_qos(prefetch\_count=1)

表示谁来谁取，不再按照奇偶数排列（同时也保证了公平的消费分发）

## 195、以下RabbitMQ的exchange type分别代表什么意思？如：fanout、direct、topic。

amqp协议中的核心思想就是生产者和消费者隔离，生产者从不直接将消息发送给队列。

生产者通常不知道是否一个消息会被发送到队列中，只是将消息发送到一个交换机。

先由Exchange来接收，然后Exchange按照特定的策略转发到Queue进行存储。

同理，消费者也是如此。Exchange 就类似于一个交换机，转发各个消息分发到相应的队列中。

--------------------------------------------------

type=fanout 类似发布者订阅者模式，会为每一个订阅者创建一个队列，而发布者发布消息时，会将消息放置在所有相关队列中

type=direct 队列绑定关键字，发送者将数据根据关键字发送到消息exchange，exchange根据 关键字 判定应该将数据发送至指定队列。

type=topic 队列绑定几个模糊的关键字，之后发送者将数据发送到exchange，exchange将传入”路由值“和 ”关键字“进行匹配，匹配成功，

则将数据发送到指定队列。

---------------------------------------------------

发送者路由值 队列中

old.boy.python old.\* -- 不匹配 \*表示匹配一个

old.boy.python old.# -- 匹配 #表示匹配0个或多个

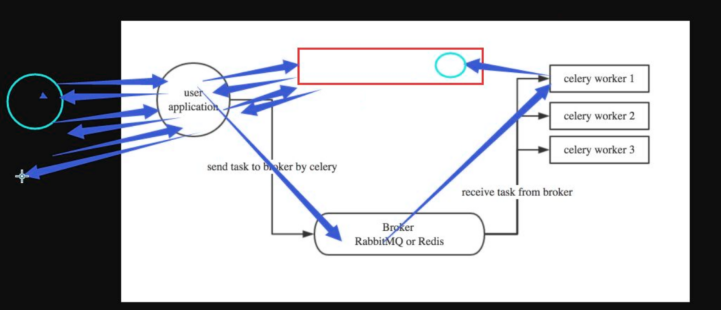
## 196、简述 celery 是什么以及应用场景？

# Celery是由Python开发的一个简单、灵活、可靠的处理大量任务的分发系统，

# 它不仅支持实时处理也支持任务调度。

# http://www.cnblogs.com/wupeiqi/articles/8796552.html

## 197、简述celery运行机制。



## 198、celery如何实现定时任务？

# celery实现定时任务

启用Celery的定时任务需要设置CELERYBEAT\_SCHEDULE 。

CELERYBEAT\_SCHEDULE='djcelery.schedulers.DatabaseScheduler'#定时任务

'创建定时任务'

# 通过配置CELERYBEAT\_SCHEDULE：

#每30秒调用task.add

from datetime import timedelta

CELERYBEAT\_SCHEDULE = {

'add-every-30-seconds': {

'task': 'tasks.add',

'schedule': timedelta(seconds=30),

'args': (16, 16)

},

}

## 199、简述 celery多任务结构目录

pro\_cel

├── celery\_tasks # celery相关文件夹

│ ├── celery.py # celery连接和配置相关文件

│ └── tasks.py # 所有任务函数

├── check\_result.py # 检查结果

└── send\_task.py # 触发任务

## 200、celery中装饰器 @app.task 和 @shared\_task的区别？

# 一般情况使用的是从celeryapp中引入的app作为的装饰器：@app.task

# django那种在app中定义的task则需要使用@shared\_task

## 201、简述 seleninu模块的作用及基本使用?

Selenium是一个用于Web应用程序测试的工具，

他的测试直接运行在浏览器上，模拟真实用户，按照代码做出点击、输入、打开等操作

爬虫中使用他是为了解决requests无法解决javascript动态问题

## 202、scrapy框架中各组件的工作流程？

#1、生成初始的Requests来爬取第一个URLS，并且标识一个回调函数

第一个请求定义在start\_requests()方法内默认从start\_urls列表中获得url地址来生成Request请求，

默认的回调函数是parse方法。回调函数在下载完成返回response时自动触发

#2、在回调函数中，解析response并且返回值

返回值可以4种：

a、包含解析数据的字典

b、Item对象

c、新的Request对象（新的Requests也需要指定一个回调函数）

d、或者是可迭代对象（包含Items或Request）

#3、在回调函数中解析页面内容

通常使用Scrapy自带的Selectors，但很明显你也可以使用Beutifulsoup，lxml或其他你爱用啥用啥。

#4、最后，针对返回的Items对象将会被持久化到数据库

通过Item Pipeline组件存到数据库

或者导出到不同的文件（通过Feed exports）

http://www.cnblogs.com/wupeiqi/articles/6229292.html

## 203、在scrapy框架中如何设置代理（两种方法）？

方式一：内置添加代理功能

# -\*- coding: utf-8 -\*-

import os

import scrapy

from scrapy.http import Request

class ChoutiSpider(scrapy.Spider):

name = 'chouti'

allowed\_domains = ['chouti.com']

start\_urls = ['https://dig.chouti.com/']

def start\_requests(self):

os.environ['HTTP\_PROXY'] = "http://192.168.11.11"

for url in self.start\_urls:

yield Request(url=url,callback=self.parse)

def parse(self, response):

print(response)

方式二：自定义下载中间件

import random

import base64

import six

def to\_bytes(text, encoding=None, errors='strict'):

"""Return the binary representation of `text`. If `text`

is already a bytes object, return it as-is."""

if isinstance(text, bytes):

return text

if not isinstance(text, six.string\_types):

raise TypeError('to\_bytes must receive a unicode, str or bytes '

'object, got %s' % type(text).\_\_name\_\_)

if encoding is None:

encoding = 'utf-8'

return text.encode(encoding, errors)

class MyProxyDownloaderMiddleware(object):

def process\_request(self, request, spider):

proxy\_list = [

{'ip\_port': '111.11.228.75:80', 'user\_pass': 'xxx:123'},

{'ip\_port': '120.198.243.22:80', 'user\_pass': ''},

{'ip\_port': '111.8.60.9:8123', 'user\_pass': ''},

{'ip\_port': '101.71.27.120:80', 'user\_pass': ''},

{'ip\_port': '122.96.59.104:80', 'user\_pass': ''},

{'ip\_port': '122.224.249.122:8088', 'user\_pass': ''},

]

proxy = random.choice(proxy\_list)

if proxy['user\_pass'] is not None:

request.meta['proxy'] = to\_bytes("http://%s" % proxy['ip\_port'])

encoded\_user\_pass = base64.encodestring(to\_bytes(proxy['user\_pass']))

request.headers['Proxy-Authorization'] = to\_bytes('Basic ' + encoded\_user\_pass)

else:

request.meta['proxy'] = to\_bytes("http://%s" % proxy['ip\_port'])

配置：

DOWNLOADER\_MIDDLEWARES = {

# 'xiaohan.middlewares.MyProxyDownloaderMiddleware': 543,

}

## 204、scrapy框架中如何实现大文件的下载？

from twisted.web.client import Agent, getPage, ResponseDone, PotentialDataLoss

from twisted.internet import defer, reactor, protocol

from twisted.web.\_newclient import Response

from io import BytesIO

class \_ResponseReader(protocol.Protocol):

def \_\_init\_\_(self, finished, txresponse, file\_name):

self.\_finished = finished

self.\_txresponse = txresponse

self.\_bytes\_received = 0

self.f = open(file\_name, mode='wb')

def dataReceived(self, bodyBytes):

self.\_bytes\_received += len(bodyBytes)

# 一点一点的下载

self.f.write(bodyBytes)

self.f.flush()

def connectionLost(self, reason):

if self.\_finished.called:

return

if reason.check(ResponseDone):

# 下载完成

self.\_finished.callback((self.\_txresponse, 'success'))

elif reason.check(PotentialDataLoss):

# 下载部分

self.\_finished.callback((self.\_txresponse, 'partial'))

else:

# 下载异常

self.\_finished.errback(reason)

self.f.close()

## 205、scrapy中如何实现限速？

http://scrapy-chs.readthedocs.io/zh\_CN/1.0/topics/autothrottle.html

## 206、[scrapy中如何实现暂停爬虫？](https://scrapy-chs.readthedocs.io/zh_CN/1.0/topics/jobs.html?highlight=item)

# 有些情况下，例如爬取大的站点，我们希望能暂停爬取，之后再恢复运行。

# Scrapy通过如下工具支持这个功能:

一个把调度请求保存在磁盘的调度器

一个把访问请求保存在磁盘的副本过滤器[duplicates filter]

一个能持续保持爬虫状态(键/值对)的扩展

Job 路径

要启用持久化支持，你只需要通过 JOBDIR 设置 job directory 选项。

这个路径将会存储所有的请求数据来保持一个单独任务的状态(例如：一次spider爬取(a spider run))。

必须要注意的是，这个目录不允许被不同的spider 共享，甚至是同一个spider的不同jobs/runs也不行。

也就是说，这个目录就是存储一个 单独 job的状态信息。

## 020、scrapy中如何进行[自定制命令](http://www.cnblogs.com/wupeiqi/articles/6229292.html)？

在spiders同级创建任意目录，如：commands

在其中创建'crawlall.py'文件（此处文件名就是自定义的命令）

from scrapy.commands import ScrapyCommand

from scrapy.utils.project import get\_project\_settings

class Command(ScrapyCommand):

requires\_project = True

def syntax(self):

return '[options]'

def short\_desc(self):

return 'Runs all of the spiders'

def run(self, args, opts):

spider\_list = self.crawler\_process.spiders.list()

for name in spider\_list:

self.crawler\_process.crawl(name, \*\*opts.\_\_dict\_\_)

self.crawler\_process.start()

在'settings.py'中添加配置'COMMANDS\_MODULE = '项目名称.目录名称''

在项目目录执行命令：'scrapy crawlall'

## 208、scrapy中如何实现的记录爬虫的深度？

'DepthMiddleware'是一个用于追踪每个Request在被爬取的网站的深度的中间件。

其可以用来限制爬取深度的最大深度或类似的事情。

'DepthMiddleware'可以通过下列设置进行配置(更多内容请参考设置文档):

'DEPTH\_LIMIT':爬取所允许的最大深度，如果为0，则没有限制。

'DEPTH\_STATS':是否收集爬取状态。

'DEPTH\_PRIORITY':是否根据其深度对requet安排优先

## 209、scrapy中的pipelines工作原理？

Scrapy 提供了 pipeline 模块来执行保存数据的操作。

在创建的 Scrapy 项目中自动创建了一个 pipeline.py 文件，同时创建了一个默认的 Pipeline 类。

我们可以根据需要自定义 Pipeline 类，然后在 settings.py 文件中进行配置即可

## 210、scrapy的pipelines如何丢弃一个item对象？

通过raise DropItem**()**方法

## 211、简述scrapy中爬虫中间件和下载中间件的作用？

http://www.cnblogs.com/wupeiqi/articles/6229292.html

## 212、scrapy-redis组件的作用？

实现了分布式爬虫，url去重、调度器、数据持久化

'scheduler'调度器

'dupefilter'URL去重规则（被调度器使用）

'pipeline'数据持久化

## 213、scrapy-redis组件中如何实现的任务的去重？

a. 内部进行配置，连接Redis

b.去重规则通过redis的集合完成，集合的Key为：

   key = defaults.DUPEFILTER\_KEY % {'timestamp': int(time.time())}

   默认配置：

      DUPEFILTER\_KEY = 'dupefilter:%(timestamp)s'

c.去重规则中将url转换成唯一标示，然后在redis中检查是否已经在集合中存在

   from scrapy.utils import request

   from scrapy.http import Request

   req **=**Request**(**url**='http://www.cnblogs.com/wupeiqi.html')**

result **=**request.request\_fingerprint**(**req**)**

print**(**result**)**# 8ea4fd67887449313ccc12e5b6b92510cc53675c

scrapy和scrapy-redis的去重规则（源码）

1. scrapy中去重规则是如何实现？

class RFPDupeFilter(BaseDupeFilter):

"""Request Fingerprint duplicates filter"""

def \_\_init\_\_(self, path=None, debug=False):

self.fingerprints = set()

@classmethod

def from\_settings(cls, settings):

debug = settings.getbool('DUPEFILTER\_DEBUG')

return cls(job\_dir(settings), debug)

def request\_seen(self, request):

# 将request对象转换成唯一标识。

fp = self.request\_fingerprint(request)

# 判断在集合中是否存在，如果存在则返回True，表示已经访问过。

if fp in self.fingerprints:

return True

# 之前未访问过，将url添加到访问记录中。

self.fingerprints.add(fp)

def request\_fingerprint(self, request):

return request\_fingerprint(request)

2. scrapy-redis中去重规则是如何实现？

class RFPDupeFilter(BaseDupeFilter):

"""Redis-based request duplicates filter.

This class can also be used with default Scrapy's scheduler.

"""

logger = logger

def \_\_init\_\_(self, server, key, debug=False):

# self.server = redis连接

self.server = server

# self.key = dupefilter:123912873234

self.key = key

@classmethod

def from\_settings(cls, settings):

# 读取配置，连接redis

server = get\_redis\_from\_settings(settings)

# key = dupefilter:123912873234

key = defaults.DUPEFILTER\_KEY % {'timestamp': int(time.time())}

debug = settings.getbool('DUPEFILTER\_DEBUG')

return cls(server, key=key, debug=debug)

@classmethod

def from\_crawler(cls, crawler):

return cls.from\_settings(crawler.settings)

def request\_seen(self, request):

fp = self.request\_fingerprint(request)

# This returns the number of values added, zero if already exists.

# self.server=redis连接

# 添加到redis集合中：1，添加工程；0，已经存在

added = self.server.sadd(self.key, fp)

return added == 0

def request\_fingerprint(self, request):

return request\_fingerprint(request)

def close(self, reason=''):

self.clear()

def clear(self):

"""Clears fingerprints data."""

self.server.delete(self.key)

## 230、scrapy-redis的调度器如何实现任务的深度优先和广度优先？

　　https://www.cnblogs.com/pupilheart/articles/9851936.html

　　https://www.cnblogs.com/pupilheart/articles/9851964.html