****

**人工智能程序设计(python)**

**实验报告**

**实验题目： 设计百度百科人物遍历爬取工具**

**专 业： 人工智能**

**姓 名： 马承乾**

**实验日期**：  **2023.4.20**

**指导教师： 曹冬林**

1. **实验目的**

设计百度百科人物遍历爬取工具。

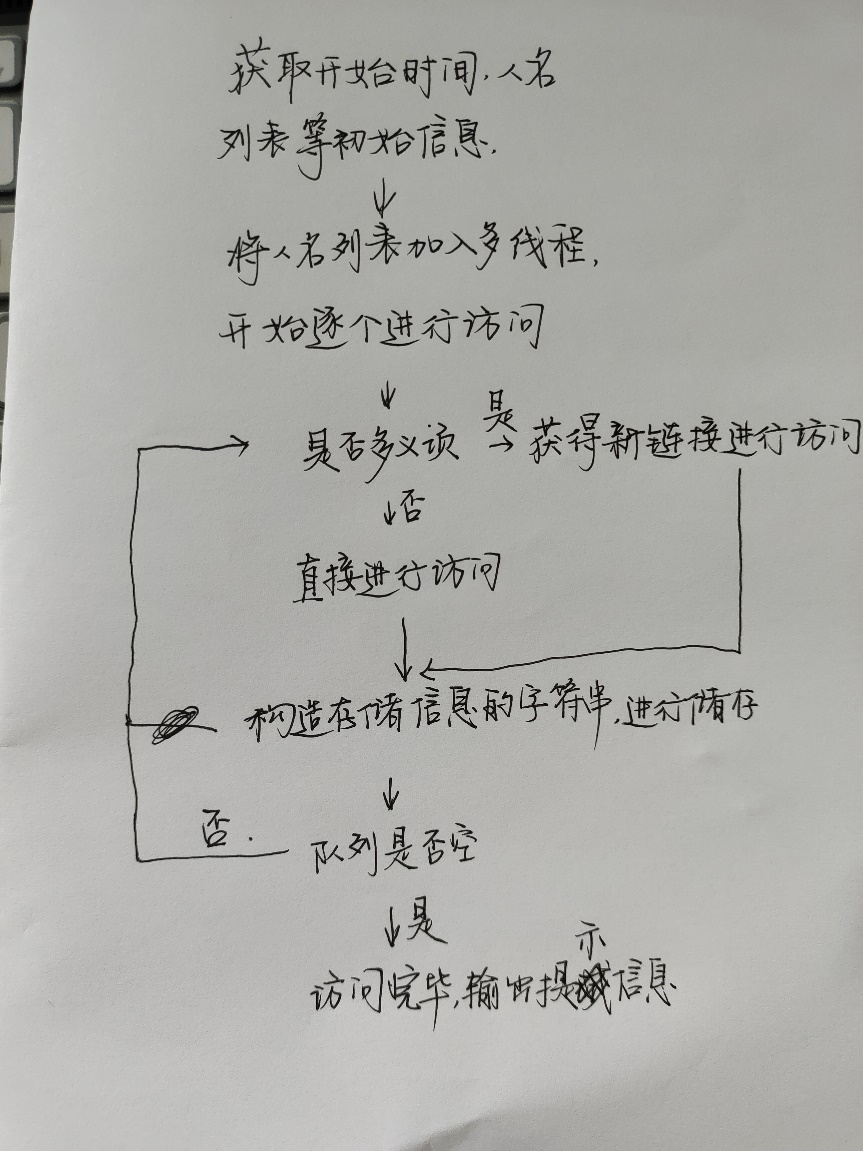
1. **实验内容**

遍历百度百科的人物信息

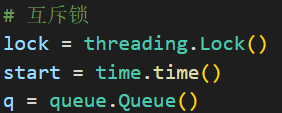
人物数不少于2000个

结合实验1， 抽取人物信息

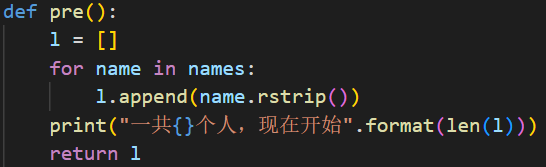
1. **实验步骤以及结果（完整代码见附件）**
2. 流程总览:

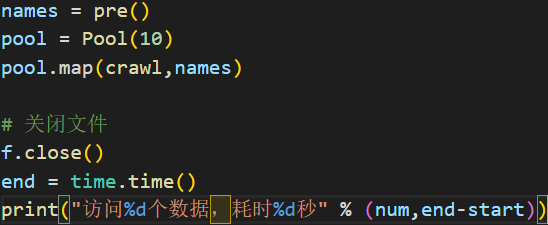


1. 第一步，获取程序开始时间，第二步获取一个互斥锁，第三步，创建一个空队列，第四步，从人名文件中获取已经准备好的1万个人名，将其储存为列表的形式。

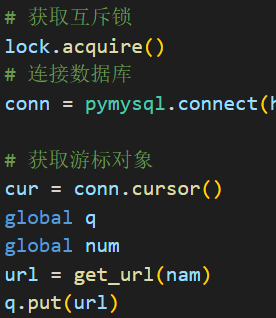


1. 随后用字符串方法去除人名，尾部的空格等信息等空白信息，输出总的人物信息，向用户说明一共有多少个人名需要被爬取后正式开始爬取，设置十个线程，将列表中的人名作为参数，交给Crawl函数进行爬取，爬取结束以后，关闭存储人名的文件获得结束时间，并向用户输出。

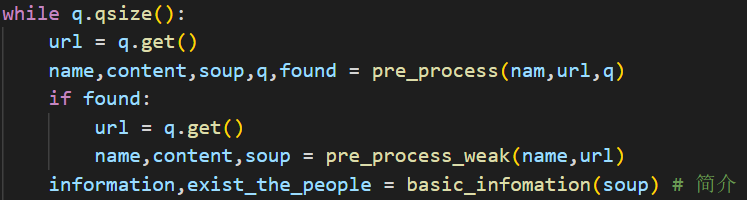




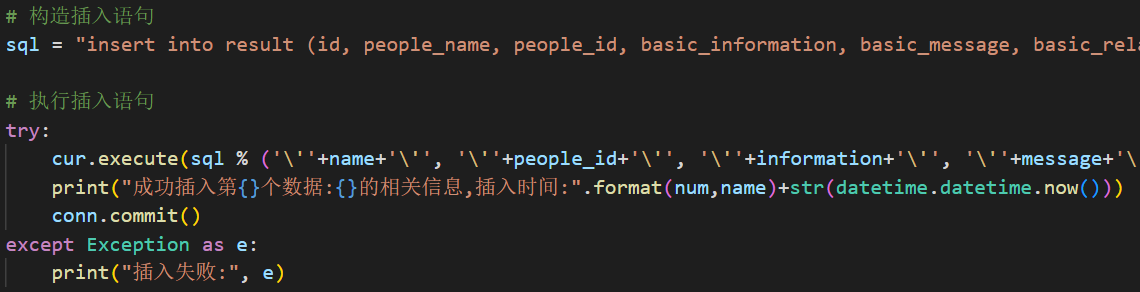
1. 具体的爬取一个人物信息的方法如下，首先，第一步获取一个互斥锁，第二步连接到数据库，并获取数据库中的游标对象，第三步，获取全局变量队列和总的人数，第四步，使用当前的人名，构成一个需要访问的网址。第五步，将该网址放入队列中，随后开始访问。



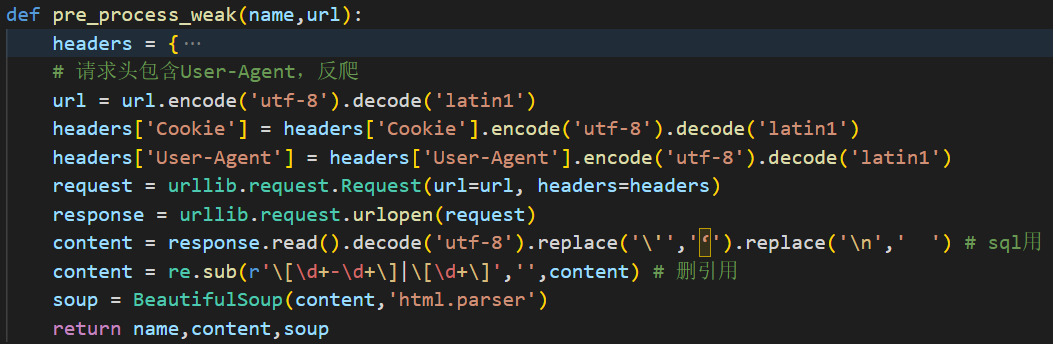
1. 首先从队列中取出一个需要被访问的人物信息的网址，对其进行访问，首先对该网页进行判断，如果该网页存储的是一个多义项人名，则在pro\_process函数中，将这些多义项对应的人物链接存入到队列q中，并返回标记是否该人名为多义项的标志变量found，如果该项是多义项，则当前队列中存储着多个网址，这些网址是该多义项对应的真正需要被访问的网址，found=1提示用户，这是一个多义项的网址，因此，从队列中取出一个元素，该元素就是真正需要被访问的网址。



1. 对其进行访问后，获取我们需要得知的name, soup等内容，使用实验一中的方法对其进行信息提取，随后构造一个数据库插入语句，并将其插入循环此过程，直到将队列中所有网址都进行访问并获取信息为止，循环结束后断开与数据库的连接，并释放互斥锁。



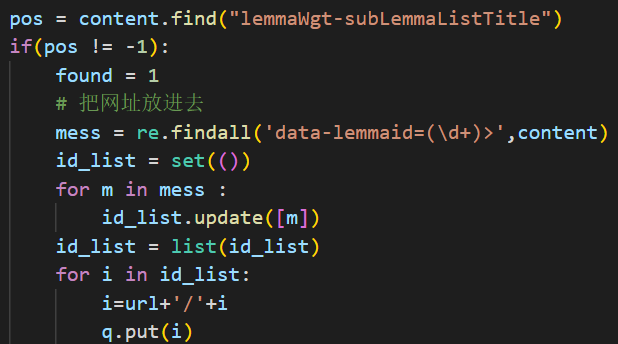
1. 函数pro\_process是对一个网址进行第一步处理的函数，如果某个网址对应的网页不是多义项，则该函数可以返回该网页中我们需要的人物信息，如果并且告诉用户，该网页并不储存着一个多义项，如果该网页存储的是一个多义项，则用户需要根据返回的标志变量来使用pre\_process\_weak函数重新对人物信息进行提取。
2. Pre\_process\_weak函数是对函数pre\_process的阉割，其中内容仅仅只是删去了对该网页是否是一个多义项网页的判断。



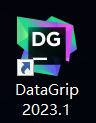
1. 注意，在构造插入语句时，数据库的插入语句需要使用字符串方法来构造，且根据数据库插入方法的格式，必须在需要被插入的信息中两旁加上引号，因此，为了避免加在信息两旁的引号与信息当中的引号匹配，我们需要对网页中的信息进行预处理，将网页中所有单引号全部使用字符串方法替换为中文单引号来解决单引号的匹配问题。



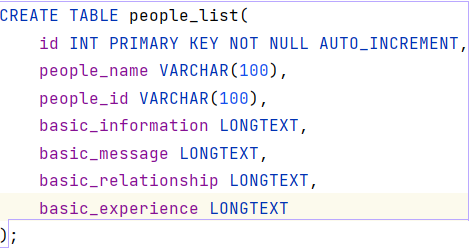
1. 对一个网址对应的网页是否是多意向的判断方法是，我们注意到，在一个网页中，如果是一个多义项的话，该网页的源代码，会有一串如下的特殊信息，并且在该特殊信息中，我们可以找到对应的几个多义项所储存的人物的ID，这个ID我猜测猜测，是百度用来区分词条的，我们得到这些多义项的ID之后就可以使用这些ID来构成，我们需要访问的人物的网址，从而进行下一次访问。



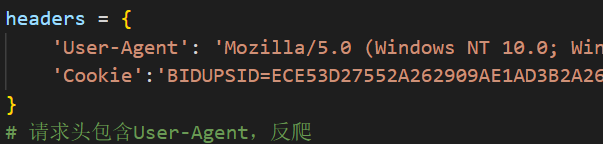
1. 为了更好，更方便的储存数据，以及方便实验三中对数据进行查询，我们使用了数据库来存储信息，管理数据库的软件是以下这个。



1. 存储的信息有以下六项，分别是人物的姓名，人物在百度网页中对应的ID号、人物的基本信息、简介、人物关系以及人物生平，对于一些不知名的人物，百度对其并没有进行收录，我们在插入这种数据时，将该人物的人物信息等内容全部存储为空字符串，对于一些缺少的信息，例如某个人物可能没有人物关系，某个人物可能并没有被百度收录其人物关系，我们也将其插入为空字符串，人物姓名等4项存储的形式都是数据库语言中的长文本形式，该类型最多可以存储4GB的文件，4GB的内容足够存储百度百科中任何一个人物的人物信息。



1. 其中人物姓名和人物ID使用100个字符的字符串来存储，其余四个信息使用长文本类型来存储。
2. 在实验过程中，我们遇见了被百度反爬虫的情况，为了解决这个问题，在访问百度时，要加上自己在百度网页的cookie信息来进行反爬操作。

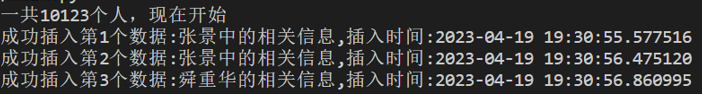


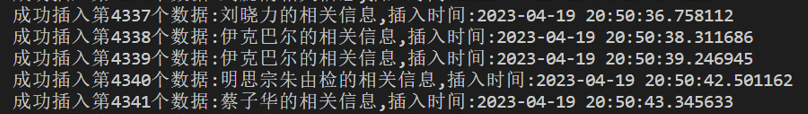
1. 与此同时，在实验过程中我发现urllib库对文本信息的解码方式是la tin-1，我们的字符串使用UTF8格式进行的编码，因此，要让urllib认识我们的字符串一定要用latin-1格式来解码，才能让urllib库对我们的网页进行正确的访问。

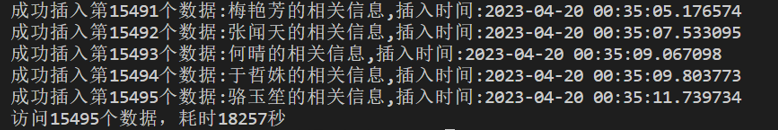


1. **实验结果与分析**

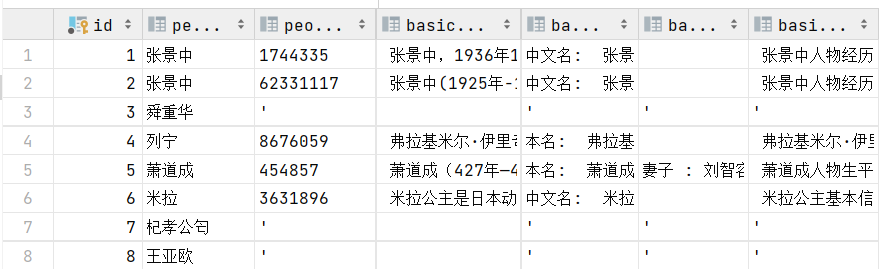
运行过程

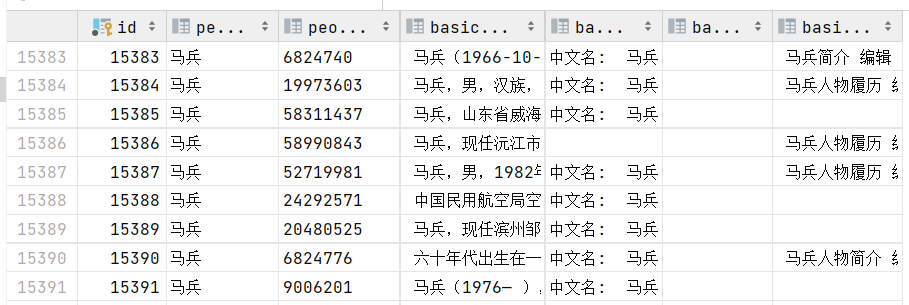






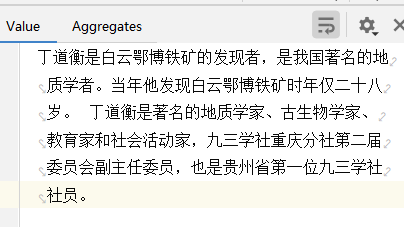
运行结果：

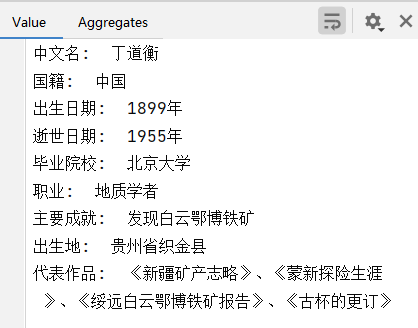




进行详细查看：







1. **实验总结**

通过本实验，我学会了使用网络爬取库urllib来访问网页，使用beau tiful soup, json, lxml, re等扩展库对网页进行爬取以及对网页信息进行解析。使用多线程方法，加快爬虫的速度。使用数据库存储信息，以加快插入所获得的信息的速度，使用互斥锁来保证多线程访问数据库时，多个线程之间不会相互冲突。通过本次实验，我提升了自己的代码能力和对python语言的理解。

1. **代码附录：**

main.py

import pymysql

from work import \*

import queue

import time

from multiprocessing.dummy import Pool

import threading

import datetime

# 互斥锁

lock = threading.Lock()

start = time.time()

q = queue.Queue()

# 爬取个数

num = 0

f = open('res.txt','r',encoding='utf8')

names = f.readlines()

def pre():

    l = []

    for name in names:

        l.append(name.rstrip())

    print("一共{}个人，现在开始".format(len(l)))

    return l

# 查询函数

def crawl(nam):

    # 获取互斥锁

    lock.acquire()

    # 连接数据库

    conn = pymysql.connect(host="localhost", user="root", password="0000", db="baidu\_person", charset="utf8")

    # 获取游标对象

    cur = conn.cursor()

    global q

    global num

    url = get\_url(nam)

    q.put(url)

    while q.qsize():

        url = q.get()

        name,content,soup,q,found = pre\_process(nam,url,q)

        if found:

            url = q.get()

            name,content,soup = pre\_process\_weak(name,url)

        information,exist\_the\_people = basic\_infomation(soup) # 简介

        num += 1

        if exist\_the\_people:

            people\_id,message,relation,experience = get\_id(content),basic\_message(content),basic\_relationship(content,name),basic\_experience(soup) # 基本信息, 关系,履历

        else:

            people\_id,message,relation,experience = '\'\'','\'\'','\'\'','\'\''

        # 构造插入语句

        sql = "insert into result (id, people\_name, people\_id, basic\_information, basic\_message, basic\_relationship, basic\_experience) values (0, %s, %s, %s, %s, %s, %s);"

        # 执行插入语句

        try:

            cur.execute(sql % ('\''+name+'\'', '\''+people\_id+'\'', '\''+information+'\'', '\''+message+'\'', '\''+relation+'\'', '\''+experience+'\''))

            print("成功插入第{}个数据:{}的相关信息,插入时间:".format(num,name)+str(datetime.datetime.now()))

            conn.commit()

        except Exception as e:

            print("插入失败:", e)

    cur.close()

    conn.close()

    # 释放互斥锁

    lock.release()

names = pre()

pool = Pool(10)

pool.map(crawl,names)

# 关闭文件

f.close()

end = time.time()

print("访问%d个数据，耗时%d秒" % (num,end-start))

work.py

from bs4 import BeautifulSoup

import re

import urllib.parse

import urllib.request

import json

import jsonpath

from lxml import etree

def get\_url(name):

    return 'https://baike.baidu.com/item/' + urllib.parse.quote(name)

def pre\_process(name,url,q):

    found = 0 # 是否找到

    headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36 Edg/112.0.1722.48',

        'Cookie':'BIDUPSID=ECE53D27552A262909AE1AD3B2A261A8; PSTM=1673444977; \_\_bid\_n=18669da4a119648a23c495; ZFY=L54qHzUJb25R:A7D6yy4WApVCa3hw1EEgBH7rgMe2Y4E:C; newlogin=1; BAIDUID=CB3D294E78B867CA647893E7FEA39C1D:FG=1; BAIDUID\_BFESS=CB3D294E78B867CA647893E7FEA39C1D:FG=1; BDUSS=lNqV25heDVEZ2p5dHlxM3Q0VkhremU1QnFpajdPRWxEMEhJdURmczBDRjg5VnBrRVFBQUFBJCQAAAAAAAAAAAEAAADmZDCRztK1xMPOsKHO0tf21vcAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHxoM2R8aDNkN; BDUSS\_BFESS=lNqV25heDVEZ2p5dHlxM3Q0VkhremU1QnFpajdPRWxEMEhJdURmczBDRjg5VnBrRVFBQUFBJCQAAAAAAAAAAAEAAADmZDCRztK1xMPOsKHO0tf21vcAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHxoM2R8aDNkN; MCITY=-:; BA\_HECTOR=a0a0058l8l01800l01852ke91i3tot61m; BDORZ=B490B5EBF6F3CD402E515D22BCDA1598; channel=bing; BK\_SEARCHLOG={"key":["李亚轩","龙文章","龙文章我的团","沈从文","李白","人物","体育人物","历史人物体育人物","硪维度大发给","剑"]}; X\_ST\_FLOW=0; Hm\_lvt\_55b574651fcae74b0a9f1cf9c8d7c93a=1681843122,1681843636,1681869634,1681887405; Hm\_lpvt\_55b574651fcae74b0a9f1cf9c8d7c93a=1681887412; H\_PS\_PSSID=38516\_36559\_38470\_38345\_38468\_38377\_38485\_37930\_38414\_38387\_38356\_26350\_38186; zhishiTopicRequestTime=1681887662692; ab\_sr=1.0.1\_MmQ1MGRmNzcyN2RjNmNjMDNiMThjNzg5ZGMwNzNhZmM5OWMzMWM2ZmI0NDE5OWEyNzU2NTlmNzIzZTlhZjA0N2Y4ZjA2MmI5ODMwZjA5MmQ1MTQ1ZmM4MzQ2OTRiN2NiOTdjYjFjNzBhNDgzZjhjYmM1NmJhOTVhNzNkYjJjNWRjMzAzN2JiZDU4MTM5NmNjMGE4NzYxMWUxN2YyMzhlZWQ2ODU1MTFiNjMzYzNhMDQ4NjkzOTczZDAwNGE5YTFl; RT="z=1&dm=baidu.com&si=8fb893d2-3fb5-413a-b287-e9199821df6b&ss=lgnbea3y&sl=o&tt=63l&bcn=https://fclog.baidu.com/log/weirwood?type=perf"; baikeVisitId=044fe712-1133-4fb5-8305-ce4d6f8a7d44'

    }

    # 请求头包含User-Agent，反爬

    url = url.encode('utf-8').decode('latin1')

    headers['Cookie'] = headers['Cookie'].encode('utf-8').decode('latin1')

    headers['User-Agent'] = headers['User-Agent'].encode('utf-8').decode('latin1')

    request = urllib.request.Request(url=url, headers=headers)

    response = urllib.request.urlopen(request)

    content = response.read().decode('utf-8').replace('\'','‘').replace('\n','  ') # sql用

    content = re.sub(r'\[\d+-\d+\]|\[\d+\]','',content) # 删引用

    soup = BeautifulSoup(content,'html.parser')

    pos = content.find("lemmaWgt-subLemmaListTitle")

    if(pos != -1):

        found = 1

        # 把网址放进去

        mess = re.findall('data-lemmaid=(\d+)>',content)

        id\_list = set(())

        for m in mess :

            id\_list.update([m])

        id\_list = list(id\_list)

        for i in id\_list:

            i=url+'/'+i

            q.put(i)

    return name,content,soup,q,found

def pre\_process\_weak(name,url):

    headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36 Edg/112.0.1722.48',

        'Cookie':'BIDUPSID=ECE53D27552A262909AE1AD3B2A261A8; PSTM=1673444977; \_\_bid\_n=18669da4a119648a23c495; ZFY=L54qHzUJb25R:A7D6yy4WApVCa3hw1EEgBH7rgMe2Y4E:C; newlogin=1; BAIDUID=CB3D294E78B867CA647893E7FEA39C1D:FG=1; BAIDUID\_BFESS=CB3D294E78B867CA647893E7FEA39C1D:FG=1; BDUSS=lNqV25heDVEZ2p5dHlxM3Q0VkhremU1QnFpajdPRWxEMEhJdURmczBDRjg5VnBrRVFBQUFBJCQAAAAAAAAAAAEAAADmZDCRztK1xMPOsKHO0tf21vcAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHxoM2R8aDNkN; BDUSS\_BFESS=lNqV25heDVEZ2p5dHlxM3Q0VkhremU1QnFpajdPRWxEMEhJdURmczBDRjg5VnBrRVFBQUFBJCQAAAAAAAAAAAEAAADmZDCRztK1xMPOsKHO0tf21vcAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHxoM2R8aDNkN; MCITY=-:; BA\_HECTOR=a0a0058l8l01800l01852ke91i3tot61m; BDORZ=B490B5EBF6F3CD402E515D22BCDA1598; channel=bing; BK\_SEARCHLOG={"key":["李亚轩","龙文章","龙文章我的团","沈从文","李白","人物","体育人物","历史人物体育人物","硪维度大发给","剑"]}; X\_ST\_FLOW=0; Hm\_lvt\_55b574651fcae74b0a9f1cf9c8d7c93a=1681843122,1681843636,1681869634,1681887405; Hm\_lpvt\_55b574651fcae74b0a9f1cf9c8d7c93a=1681887412; H\_PS\_PSSID=38516\_36559\_38470\_38345\_38468\_38377\_38485\_37930\_38414\_38387\_38356\_26350\_38186; zhishiTopicRequestTime=1681887662692; ab\_sr=1.0.1\_MmQ1MGRmNzcyN2RjNmNjMDNiMThjNzg5ZGMwNzNhZmM5OWMzMWM2ZmI0NDE5OWEyNzU2NTlmNzIzZTlhZjA0N2Y4ZjA2MmI5ODMwZjA5MmQ1MTQ1ZmM4MzQ2OTRiN2NiOTdjYjFjNzBhNDgzZjhjYmM1NmJhOTVhNzNkYjJjNWRjMzAzN2JiZDU4MTM5NmNjMGE4NzYxMWUxN2YyMzhlZWQ2ODU1MTFiNjMzYzNhMDQ4NjkzOTczZDAwNGE5YTFl; RT="z=1&dm=baidu.com&si=8fb893d2-3fb5-413a-b287-e9199821df6b&ss=lgnbea3y&sl=o&tt=63l&bcn=https://fclog.baidu.com/log/weirwood?type=perf"; baikeVisitId=044fe712-1133-4fb5-8305-ce4d6f8a7d44'

    }

    # 请求头包含User-Agent，反爬

    url = url.encode('utf-8').decode('latin1')

    headers['Cookie'] = headers['Cookie'].encode('utf-8').decode('latin1')

    headers['User-Agent'] = headers['User-Agent'].encode('utf-8').decode('latin1')

    request = urllib.request.Request(url=url, headers=headers)

    response = urllib.request.urlopen(request)

    content = response.read().decode('utf-8').replace('\'','‘').replace('\n','  ') # sql用

    content = re.sub(r'\[\d+-\d+\]|\[\d+\]','',content) # 删引用

    soup = BeautifulSoup(content,'html.parser')

    return name,content,soup

def get\_id(content):     #获取每个人物对应的唯一id

    html\_tree=etree.HTML(content)   #利用xpath解析

    id=html\_tree.xpath('//div[@style="display:none;"]//@data-lemmaid')   #找到合适的xpath路径，返回一个列表

    return str(id[0])

# 人物简介

def basic\_infomation(soup):

    info = soup.find(class\_ = 'lemma-summary J-summary')

    information = ''

    if info == None:

        exist\_the\_people = 0

    else:

        exist\_the\_people = 1

    if exist\_the\_people == 0:

        return information,exist\_the\_people

    for i in info:

        information = information + i.get\_text()

    return information,exist\_the\_people

# 人物基本信息

def basic\_message(content):

    item = re.findall('<dt class="basicInfo-item name" id="basic-name">(.\*?)</dt>',content)

    temp = []

    for i in item:

        i = i.replace('&nbsp','').replace(';','')

        temp.append(i)

    item = temp

    mess = re.findall('<dd class="basicInfo-item value">(.\*?)</dd>',content)

    temp = []

    for i in mess:

        i = re.sub('<a(.\*?)>','',i)

        i = re.sub('</a>','',i)

        i = re.sub('<br/>(.\*?)<br/>','',i)

        i = re.sub('<sup(.\*?)</sup>&nbsp;',',',i)

        i = re.sub('<em(.\*?)em>',',',i)

        i = re.sub('<div(.\*?)收起',',',i)

        i = re.sub('展开','',i)

        temp.append(i)

    mess = temp

    message = ''

    for i,j in zip(item,mess):

        t = i + ':' + j + '\n'

        message = message + t

    return message

# 人物生平

def basic\_experience(soup):

    mess = soup.find(class\_ = 'anchor-list')

    experience = ''

    if mess == None:

        return experience

    mess = mess.next\_siblings

    for i in mess:

        text = i.get\_text()

        text\_pos = text.find("参考资料")

        if(text\_pos != -1):

            text = text[0:text\_pos]

            break

        text\_pos = text.find("词条图册")

        if(text\_pos != -1):

            text = text[0:text\_pos]

            break

        experience = experience + text

    return experience

def basic\_relationship(content,name):

    url = 'https://baike.baidu.com/starmap/api/gethumanrelationcard?'  #基础的url，无具体参数

    # id 和 name

    data = {

        'lemmaId': get\_id(content),

        'lemmaTitle': name

    }

    url = url + urllib.parse.urlencode(data)  #利用urlencode拼接合成完整的url

    headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36 Edg/112.0.1722.48',

        'Cookie':'BIDUPSID=ECE53D27552A262909AE1AD3B2A261A8; PSTM=1673444977; \_\_bid\_n=18669da4a119648a23c495; ZFY=L54qHzUJb25R:A7D6yy4WApVCa3hw1EEgBH7rgMe2Y4E:C; newlogin=1; BAIDUID=CB3D294E78B867CA647893E7FEA39C1D:FG=1; BAIDUID\_BFESS=CB3D294E78B867CA647893E7FEA39C1D:FG=1; BDUSS=lNqV25heDVEZ2p5dHlxM3Q0VkhremU1QnFpajdPRWxEMEhJdURmczBDRjg5VnBrRVFBQUFBJCQAAAAAAAAAAAEAAADmZDCRztK1xMPOsKHO0tf21vcAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHxoM2R8aDNkN; BDUSS\_BFESS=lNqV25heDVEZ2p5dHlxM3Q0VkhremU1QnFpajdPRWxEMEhJdURmczBDRjg5VnBrRVFBQUFBJCQAAAAAAAAAAAEAAADmZDCRztK1xMPOsKHO0tf21vcAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAHxoM2R8aDNkN; MCITY=-:; BA\_HECTOR=a0a0058l8l01800l01852ke91i3tot61m; BDORZ=B490B5EBF6F3CD402E515D22BCDA1598; channel=bing; BK\_SEARCHLOG={"key":["李亚轩","龙文章","龙文章我的团","沈从文","李白","人物","体育人物","历史人物体育人物","硪维度大发给","剑"]}; X\_ST\_FLOW=0; Hm\_lvt\_55b574651fcae74b0a9f1cf9c8d7c93a=1681843122,1681843636,1681869634,1681887405; Hm\_lpvt\_55b574651fcae74b0a9f1cf9c8d7c93a=1681887412; H\_PS\_PSSID=38516\_36559\_38470\_38345\_38468\_38377\_38485\_37930\_38414\_38387\_38356\_26350\_38186; zhishiTopicRequestTime=1681887662692; ab\_sr=1.0.1\_MmQ1MGRmNzcyN2RjNmNjMDNiMThjNzg5ZGMwNzNhZmM5OWMzMWM2ZmI0NDE5OWEyNzU2NTlmNzIzZTlhZjA0N2Y4ZjA2MmI5ODMwZjA5MmQ1MTQ1ZmM4MzQ2OTRiN2NiOTdjYjFjNzBhNDgzZjhjYmM1NmJhOTVhNzNkYjJjNWRjMzAzN2JiZDU4MTM5NmNjMGE4NzYxMWUxN2YyMzhlZWQ2ODU1MTFiNjMzYzNhMDQ4NjkzOTczZDAwNGE5YTFl; RT="z=1&dm=baidu.com&si=8fb893d2-3fb5-413a-b287-e9199821df6b&ss=lgnbea3y&sl=o&tt=63l&bcn=https://fclog.baidu.com/log/weirwood?type=perf"; baikeVisitId=044fe712-1133-4fb5-8305-ce4d6f8a7d44'

    }

    url = url.encode('utf-8').decode('latin1')

    headers['Cookie'] = headers['Cookie'].encode('utf-8').decode('latin1')

    headers['User-Agent'] = headers['User-Agent'].encode('utf-8').decode('latin1')

    request = urllib.request.Request(url=url, headers=headers)   #定制请求对象

    content = urllib.request.urlopen(request).read().decode('utf-8')  #获取服务器返回的数据并解码，获取文本内容

    with open('temp\_relation.json', 'w', encoding='utf-8') as fs: #将爬取到的内容存入本地文件中

        fs.write(content)

    obj = json.load(open('temp\_relation.json', 'r', encoding='utf-8'))  #采用Jsonpath解析方式

    relation\_name = jsonpath.jsonpath(obj, '$..relationName')   #确定合适的Jsonpath解析路径

    hisname = jsonpath.jsonpath(obj, '$..lemmaTitle')

    res=''

    if relation\_name == 0:# 为空，提前结束

        return ''

    for i, j in zip(relation\_name, hisname):#按对应的顺序输出两个列表中的值

        res = res + i + ' : ' + j + ', '

    return res.strip(', ')