Python网络爬虫笔记（二）：链接爬虫和下载限速

（一）代码1（link\_crawler()和get\_links()实现链接爬虫）

（二）delayed.py（实现下载限速的类）

1 import urllib.request as ure  
 2 import re  
 3 import urllib.parse  
 4 from delayed import WaitFor  
 5 #下载网页并返回HTML(动态加载的部分下载不了)  
 6 def download(url,user\_agent='Socrates',num=2):  
 7 print('下载:'+url)  
 8 #设置用户代理  
 9 headers = {'user\_agent':user\_agent}  
10 request = ure.Request(url,headers=headers)  
11 try:  
12 #下载网页  
13 html = ure.urlopen(request).read()  
14 except ure.URLError as e:  
15 print('下载失败'+e.reason)  
16 html=None  
17 if num>0:  
18 #遇到5XX错误时，递归调用自身重试下载，最多重复2次  
19 if hasattr(e,'code') and 500<=e.code<600:  
20 return download(url,num-1)  
21 return html  
22 #seed\_url传入一个url  
23 #link\_regex传入一个正则表达式  
24 #函数功能：提取和link\_regex匹配的所有网页链接并下载  
25 def link\_crawler(seed\_url, link\_regex):  
26 html = download(seed\_url)  
27 crawl\_queue = []  
28 #迭代get\_links（）返回的列表，将匹配正则表达式link\_regex的链接添加到列表中  
29 for link in get\_links(html):  
30 if re.match(link\_regex, link):  
31 #拼接https://www.cnblogs.com/ 和 /cate/...  
32 link = urllib.parse.urljoin(seed\_url, link)  
33 #不在列表中才添加  
34 if link not in crawl\_queue:  
35 crawl\_queue.append(link)  
36 #调用WaitFor的wait（）函数，下载限速，间隔小于2秒则等待，直到时间等于2秒才继续下载（大于则直接继续下载）  
37 waitFor = WaitFor(2)  
38 #下载crawl\_queue中的所有网页  
39 while crawl\_queue:  
40 #删除列表末尾的数据  
41 url = crawl\_queue.pop()  
42 waitFor.wait(url)  
43 download(url)  
44 #传入html对象，以列表形式返回所有链接  
45 def get\_links(html):  
46 #使用正则表达式提取html中所有网页链接  
47 webpage\_regex = re.compile('<a[^>]+href=["\'](.\*?)["\']',re.IGNORECASE)  
48 html = html.decode('utf-8')  
49 # 以列表形式返回所有网页链接  
50 return webpage\_regex.findall(html)  
51   
52 link\_crawler('https://www.cnblogs.com/','/cate/.\*')

1 import urllib.parse  
 2 import datetime  
 3 import time  
 4 class WaitFor():  
 5   
 6 def \_\_init\_\_(self,delay):  
 7 #delay：希望延迟多长时间（wait（）中的处理是以秒为单位）  
 8 self.delay = delay  
 9 #用来存放上次下载时间  
10 self.domains = dict()  
11   
12 def wait(self,url):  
13 #获取url netloc属性的值（即www.cnblogs.com，// 和第一个 /之间的内容）  
14 domain = urllib.parse.urlparse(url).netloc  
15 #存在键值为domain的数据返回value值，否则返回None  
16 last\_down = self.domains.get(domain)  
17 if self.delay >0 and last\_down is not None:  
18 # 希望延迟时间 - （当前时间-上次下载时间），seconds时间间隔以秒为单位显示  
19 sleep\_sec = self.delay-(datetime.datetime.now()-last\_down).seconds  
20 if sleep\_sec > 0:  
21 time.sleep(sleep\_sec)  
22 #将当前时间添加到domains中  
23 self.domains[domain] = datetime.datetime.now()