> 同济大学数据采集与集成实验报告06

《多线程采集同济贴吧图像文件》

> - 学院：电子信息工程学院

> - 专业：数据科学与大数据专业
> - 学号：1851804
> - 姓名：苗成林
> - 指导教师：郭玉臣
> - 时间：2021.12.12

实验报告正文

实验目的及要求

1. 掌握非结构化数据采集
2. 掌握 Python 的多线程编程方法
3. xpath解析应用

实验原理

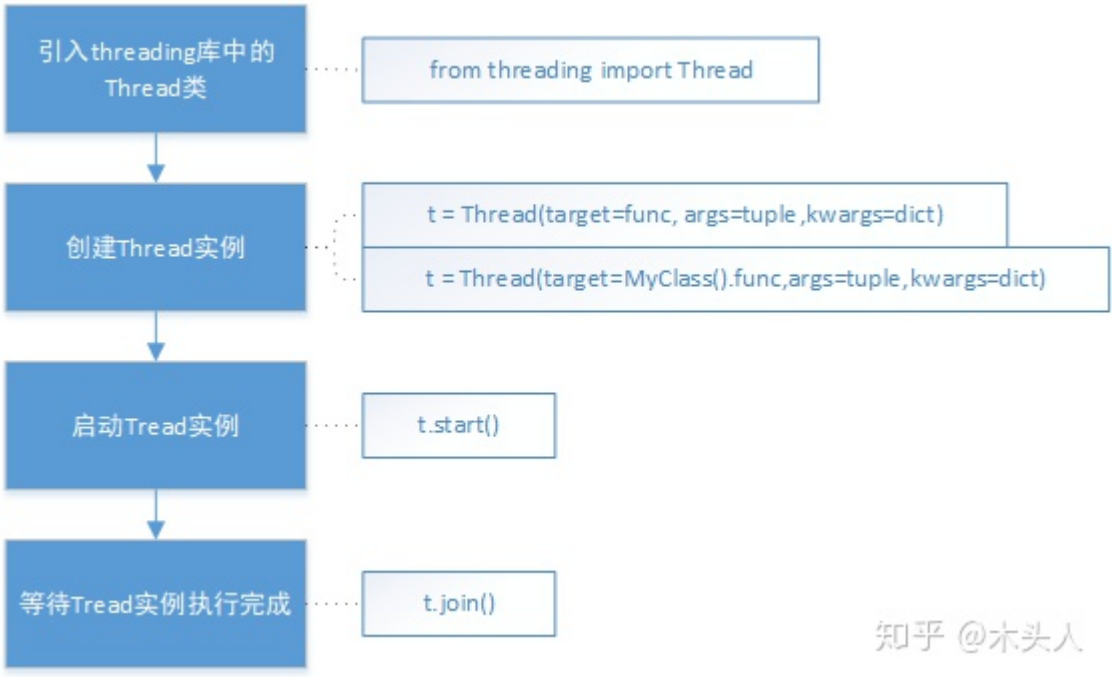
Threading 模块

threading 模块的Thread 类是主要的执行对象。使用Thread 类，可以有很多方法来创建线程。最常用的有下面三种：

> 创建Thread 的实例，传给它一个可调用对象（函数或者类的实例方法）。
> 派生Thread 的子类，并创建子类的实例。

可调用对象（函数，类的实例方法）使用多线程

步骤如下：



示例：创建Thread实例，传递给他一个函数

```
from threading import Thread
from time import sleep, ctime

def func(name, sec):
    print('---开始---', name, '时间', ctime())
    sleep(sec)
    print('***结束***', name, '时间', ctime())

# 创建 Thread 实例
t1 = Thread(target=func, args=('第一个线程', 1))
t2 = Thread(target=func, args=('第二个线程', 2))

# 启动线程运行
t1.start()
t2.start()

# 等待所有线程执行完毕
t1.join()  # join() 等待线程终止，要不然一直挂起
t2.join()
```

示例：创建Thread实例，传递给他一个类的实例方法

```
from threading import Thread
from time import sleep, ctime


class MyClass(object):

    def func(self,name,sec):
        print('---开始---', name, '时间', ctime())
        sleep(sec)
```

```
        print('***结束***', name, '时间', ctime())

def main():
    # 创建 Thread 实例
    t1 = Thread(target=MyClass().func, args=(1, 1))
    t2 = Thread(target=MyClass().func, args=(2, 2))

    # 启动线程运行
    t1.start()
    t2.start()

    # 等待所有线程执行完毕
    t1.join()  # join() 等待线程终止，要不然一直挂起
    t2.join()++-

if __name__=="__main__":
    main()
```

运行结果：

> ---开始--- 一 时间 Fri Nov 29 11:34:31 2019
> ---开始--- 二 时间 Fri Nov 29 11:34:31 2019
> ***结束*** 一 时间 Fri Nov 29 11:34:32 2019
> ***结束*** 二 时间 Fri Nov 29 11:34:33 2019

程序总共运行两秒，如果程序按照线性运行需要3秒，节约1秒钟。

Thread 实例化时需要接收 target，args（kwargs）两个参数。

target 用于接收需要使用多线程调用的对象。

args 或 kwargs 用于接收调用对象的需要用到的参数，args接收tuple，kwargs接收dict。对args与kwargs你理解的看 **木头人：Python入门 函数 提高篇** 中的**Python函数的可变参数(*args, **kwargs)** 部分。

start() 是方法用来启动线程的执行。

join() 方法是一种自旋锁，它用来等待线程终止。也可以提供超时的时间，当线程运行达到超时时间后结束线程，如join(500)，500毫秒后结束线程运行。

> 注意：如果当你的主线程还有其他事情要做，而不是等待这些线程完成，就可以不调用join()。join()方法只有在你需要等待线程完成然后在做其他事情的时候才是有用的。

派生Thread 的子类，并创建子类的实例。

我们可以通过继承Thread类，派生出一个子类，使用子类来创建多线程。

示例：派生Thread 的子类，传递给他一个可调用对象

```
from threading import Thread
from time import sleep, ctime


# 创建 Thread 的子类
class MyThread(Thread):
    def __init__(self, func, args):
        '''
        :param func: 可调用的对象
        :param args: 可调用对象的参数
        '''
```

```
        Thread.__init__(self)   # 不要忘记调用Thread的初始化方法
        self.func = func
        self.args = args

    def run(self):
        self.func(*self.args)


def func(name, sec):
    print('---开始---', name, '时间', ctime())
    sleep(sec)
    print('***结束***', name, '时间', ctime())

def main():
    # 创建 Thread 实例
    t1 = MyThread(func, (1, 1))
    t2 = MyThread(func, (2, 2))
    # 启动线程运行
    t1.start()
    t2.start()
    # 等待所有线程执行完毕
    t1.join()
    t2.join()

if __name__ == '__main__':
    main()
```

注意：不要忘记在子类中初始化父类的方法Thread.**init**(self) 。需要重构 run() 方法来执行多线程的程序。

拓展：获取可调用对象的返回值

在多线程中运行的程序时与主线程分开，我们没法直接获取线程中程序的返回值。这时就可以使用派生Thread的子类，将给过保存的实例属性中，通过一个新方法获取运行结果。

示例: 获取多线程中程序运行的结果

```
from threading import Thread
from time import sleep, ctime

# 创建 Thread 的子类
class MyThread(Thread):
    def __init__(self, func, args):
        '''
        :param func: 可调用的对象
        :param args: 可调用对象的参数
        '''
        Thread.__init__(self)
        self.func = func
        self.args = args
        self.result = None

    def run(self):
        self.result = self.func(*self.args)

    def getResult(self):
```

```
        return self.result


def func(name, sec):
    print('---开始---', name, '时间', ctime())
    sleep(sec)
    print('***结束***', name, '时间', ctime())
    return sec


def main():
    # 创建 Thread 实例
    t1 = MyThread(func, (1, 1))
    t2 = MyThread(func, (2, 2))
    # 启动线程运行
    t1.start()
    t2.start()
    # 等待所有线程执行完毕
    t1.join()
    t2.join()
    # 或线程中程序的运行结果
    print(t1.getResult())
    print(t2.getResult())


if __name__ == '__main__':
    main()
```

广度优先算法介绍

整个的广度优先爬虫过程就是从一系列的种子节点开始，把这些网页中的"子节点"(也就是超链接)提取出来，放入队列中依次进行抓取。被处理过的链接需要放 入一张表(通常称为Visited表)中。每次新处理一个链接之前，需要查看这个链接是否已经存在于Visited表中。如果存在，证明链接已经处理过， 跳过，不做处理，否则进行下一步处理。

初始的URL地址是爬虫系统中提供的种子URL(一般在系统的配置文件中指定)。当解析这些种子URL所表示的网页时，会产生新的URL(比如从页面中的<a href= "http://www.admin.com "中提取出 http://www.admin.com 这个链接)。然后，进行以下工作：

1. 把解析出的链接和Visited表中的链接进行比较，若Visited表中不存在此链接，表示其未被访问过。
2. 把链接放入TODO表中。
3. 处理完毕后，再次从TODO表中取得一条链接，直接放入Visited表中。
4. 针对这个链接所表示的网页，继续上述过程。如此循环往复。

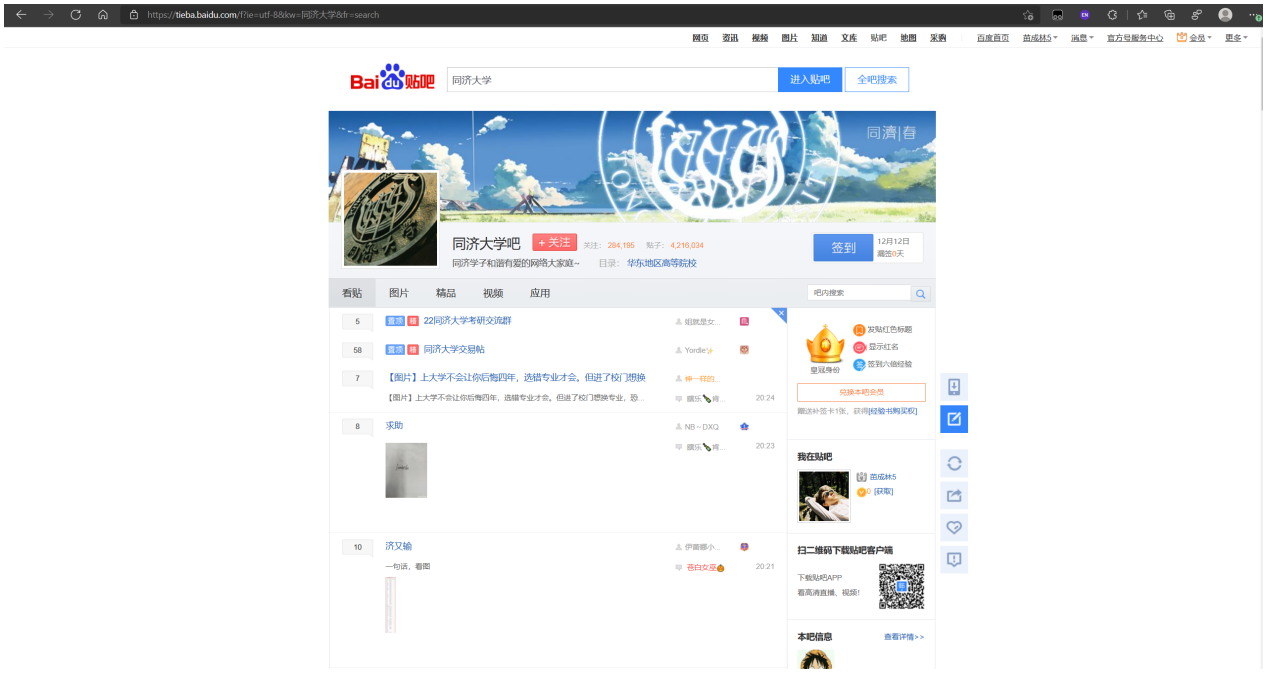
广度优先遍历是爬虫中使用最广泛的一种爬虫策略，之所以使用广度优先搜索策略，主要原因有三点：

- 重要的网页往往离种子比较近，例如我们打开新闻网站的时候往往是最热门的新闻，随着不断的深入冲浪，所看到的网页的重要性越来越低。
- 万维网的实际深度最多能达到17层，但到达某个网页总存在一条很短的路径。而广度优先遍历会以最快的速度到达这个网页。
- 广度优先有利于多爬虫的合作抓取，多爬虫合作通常先抓取站内链接，抓取的封闭性很强。

实验过程

网页展示

百度贴吧同济大学主页



获取图片

```
def get_pic(html_tiezi,counter1):

    selector = etree.HTML(html_tiezi) #可xpath对象
    #先尝试爬楼主
    #firstpic=selector.xpath('//div[@class="l_post j_l_post l_post_bright noborder
"]/div[@class="d_post_content_main d_post_content_firstfloor
"]/div/cc/div[@class="d_post_content j_d_post_content  clearfix"]/img')
    firstpic = selector.xpath('//img[@class="BDE_Image"]/@src')

    for i,j in zip(firstpic,range(counter1,counter1+len(firstpic))):
        print("存储图片：",j, "......")
        response = requests.get(i)
        img = response.content
        # 将他拷贝到本地文件 w 写  b 二进制  wb代表写入二进制文本
        with open( r'.\picOf11\{}.jpg'.format(j),'wb' ) as f:
            f.write(img)
    return counter1+len(firstpic)
```

获取url队列

```
def get_more_url_fromtibeba(html_tieba):
    link_list=[]
    selector = etree.HTML(html_tieba)
    # print(html_tieba)
    # with open("11.html",'w',encoding="utf-8") as f:
    #     f.write(html_tieba)
    for i in selector.xpath('//div[@class="threadlist_lz clearfix"]/div/a/@href'):
        link_list.append('https://tieba.baidu.com/'+i+'?pn=1')  #帖子的首页
    return link_list
```

创建Threading线程

```
def thread1(list_url,list_stop):
    global counter
    #while(list_url):
    # for i in range(1000):
    # print("休眠中")
    # time.sleep(1)
    # print("恢复")
    print("thread1")

    lock.acquire()
    url = list_url.pop(0)
    list_stop.add(url)
    lock.release()

    if re.match(r'.*/p/.*',url): ##是帖子
        html_tiezi,strr=get_html_tiezi(url)
        lock.acquire()
        counter =  get_pic(html_tiezi,counter)
        lock.release()
        # 获取不同页数url
        for i in get_more_url_fromtiezi(html_tiezi):
            if i not in list_stop:
                lock.acquire()
                list_url.append(i)
                lock.release()

    else:                           ##是贴吧目录
        html_tieba,strr=get_html_tieba(url)
        #获取目录下帖子
        for i in get_more_url_fromtibeba(html_tieba):
            if i not in list_stop:
                lock.acquire()
                list_url.append(i)
                lock.release()
        #贴吧目录图片可以不爬
        #贴吧目录下一页要存
        if url.replace(re.findall(r'.*&pn=(.*)',url)[0], str(int(re.findall(r'.*&pn=
(.*)',url)[0])+50)) not in list_stop:
            lock.acquire()
            list_url.append(url.replace(re.findall(r'.*&pn=(.*)',url)[0],
str(int(re.findall(r'.*&pn=(.*)',url)[0])+50)))
            lock.release()
```

附录

百度贴吧同济大学主页相关的网页爬取的图片展示：