Lecture7

红色标注的语句，为重点。
蓝色下划线标注的语句，说明给出了参考阅读链接，可依兴趣阅读。
紫色加粗，表示参看附件。

写在Lecture7之前

在Lecture2中，我们已经对Web端的爬虫有了一个基本认知：网页的内容由HTML文件设定，并遵从一定的语法规则。因此，在设法获取HTML内容后，我们就可以按照这样的语法规则筛选网页中我们需要的元素，存储到本地。

在Lecture2-附件4中，我们了解了如何使用requests库直接获取网页内容，并使用lxml库，依照元素的定位（xpath地址）从“杂乱”的HTML文件中筛选元素。

在Lecture2-附件5中，我们介绍了另一种抓取的思路，使用selenium控制浏览器的行为，打开我们需要的网页，执行自动化操作，再使用xpath来定位到我们所需的元素。这样的好处是，我们可以控制浏览器和网页进行真正的交互，如点击按钮、输入文本等，对于一些需要经过点击才能展现内容（使用了Ajax技术）的网站，尤为有效。

在Lecture7中，我们将按照这样的逻辑进行介绍：首先，我们通过一个例子再次回顾对selenium的使用；其次，我们结合实际操作中会遇到的问题，对该代码进行优化；接着，我们介绍将代码部署至云服务器的方法；最后，给出进一步学习的爬虫技术路线。

1.selenium爬虫回顾

——目标描述：

在这次的任务中，我们希望能够通过Bing搜索引擎，查找Linkedin上符合我们要求的信息，并将有效搜索结果的网址存储到本地，以便进一步爬取。

——目标拆解：

可以将目标拆解成如下几个步骤：1.操纵浏览器打开Bing。2.在Bing搜索框里搜索我们需要的关键字，并点击搜索按钮。3.定位每一页的搜索结果，在其中辨别我们需要的结果。4.点击“下一页”按钮，直到达到100页上限或者最后一页。5.循环1-4步。

——环境配置：

lecture2-challenge中已经提到，使用selenium进行爬虫要配置一定的环境，主要为selenium模块的安装，chromedriver的配置（一定要注意⚠️，下载对应自己Chrome版本的chromedriver）。

这里仅补充一点，大家可以将chromedriver视为一个驱动Chrome的钥匙，在配置时，你可以将存放chromedriver的地址放入环境变量中，让程序有需求时自动去这里寻找；也可以随便把chromedriver放到哪里，在代码中告诉程序这一次去哪个路径找chromedriver。具体使用哪一种方法，看大家个人的喜好。

——代码：

见Lecture7-附件1-Version1，提供了较为详细的注释。如果对为什么“这样定位”到想要的元素不了解，非常建议自己动手，观察下Bing搜索结果网页的结构，看看不同元素之间的层级关系。理解不同场景下，应该如何定位到我们所要的元素。

或者参照：https://blog.csdn.net/weixin_36279318/article/details/79475388

2.selenium爬虫代码优化

如果运气不错，Version1的代码应该能顺利运行。但在实际操作中，“能运行”不代表“能使用”。在一个较大型、耗时久的爬虫任务中，使用仅仅“能运行”的代码来完成任务，会让我们十分痛苦：被报错意外中断程序、电脑上不停弹出浏览器窗口干扰日常工作、突然断网导致程序进度中断、长达一两周的爬虫占用了手头为数不多的设备的大部分工作时长……

因此，在完成代码的基本逻辑编写后，我们往往要花相当长的时间对其进行优化，做好异常处理等工作。

代码见Lecture7-附件1-Version2.

——关闭不停弹出的浏览器：

selenium每建立一个浏览器对象，都会弹出一个Chrome窗口，对于刚刚学习selenium的同学来说，看浏览器在自己的代码指令下执行各种自动化操作，无疑是有意思的。但如果在正式爬取项目时，爬取时间往往很长，这段时间一直看着浏览器不停弹出的话，自己的电脑就几乎不可用了。

我们可以对每次新建的Chrome对象添加一些参数，来让Chrome不再弹出：

```
from selenium.webdriver.chrome.options import Options

chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
chrome_options.add_argument('--remote-debugging-port=9222')

driver = webdriver.Chrome(chrome_driver,options=chrome_options)
```

如图，新建一个selenium.webdriver.chrome.options.Options对象，在新建一个Chrome对象的时候作为参数传入，即可让Chrome具有我们指定的特征，如设置代理等。

传入’—headless’参数，即可让Chrome在后台运行，每次新建时就不会弹出了。

对于上图中的其他参数，均为为了方便后续在服务器端的部署添加的，在此不多赘述，感兴趣的朋友可以自己搜索了解。

——做好异常处理

对于一个想要稳健运行的程序，必须要做好对异常情况的处理，保证程序能顺利运行下去。

以该爬虫程序为例，程序的每一个环节都有可能出错：例如，Bing网站的加载可能太过缓慢，超出了程序的等待时间；某个按钮加载出现问题，迟迟无法点击；chromedriver遭遇了崩溃，一时间无法打开下一个Chrome；内存超出限制，程序被迫中断……每一个环节的报错都会导致程序终止。大家可以想象等待了一夜，早上起来收取结果，却发现程序已经在凌晨停止了运行是何种心情，为了避免这种情况的发生，我们使用Python的异常处理功能。

Python中异常处理主要由try……catch……except……语句进行。

具体参照：https://www.runoob.com/python/python-exceptions.html

我们可以对该程序中的如下部分做try语句处理：

```
try:
    pagebar = driver.find_element_by_class_name('b_pag')
except:
    break
try:
    next_page = pagebar.find_elements_by_tag_name('a')[-1]
    next_page.click()
except:
    break
```

这一部分代码主要做了两件事：1.定位到网页下方的序号栏（即第一页、第二页……）。2.点击“下一页”按钮。

在实际操作中，如果到最后一页了，很可能无法定位到可点击的按钮，或者由于网络原因暂时无法加载出序号栏，当程序无法点击时会直接报错，导致代码中断。

使用try语句，可以让try下方的代码块即使报错，也可以不中断程序，转而运行except语句下的代码块。在上图中，我们指定报错后跳出当前循环，意为不用再循环点击“下一页”了，默认当前搜索结果已至最后一页，进入下一个关键词的搜索。

——避免把信息存储在内存里

我们常常会把抓取到的信息存储在某个List列表里，等爬取完毕后一并写入本地文件。但这样的行为是非常危险的，List中的信息是写在内存里的，一旦代码中断，就会立刻丢失，假设我们要爬取100000条信息，并存储在了一个List中，但在99999条的时候网络断了，代码停止，我们就只能重新再来一遍。

所以实际操作中，建议大家每爬取一部分信息，就固定将其存储到本地，避免“一着不慎，满盘皆输”的情况出现。

3.将爬虫代码部署到服务器端

然而，在更大的爬虫任务中，我们会遇到更多棘手的问题：数据规模太大，即便在一台机器上多进程爬取，也需要几个月的时间；所处网络环境不稳定，例如用自己的笔记本爬取，但宿舍总是断网；只有一台主力计算设备，如果爬数据要一周，那一周都不能用这台电脑……

在这种环境下爬数据是异常痛苦的（亲身经历），为了不让大家经历和我一样的痛苦，强烈建议大家掌握将代码部署至服务器的方法。

将代码部署至服务器有如下几个优点：1.服务器的运行环境非常稳定，不用担心网络等故障。2.不会占用自己的主力设备，电脑可以随时关机，需要时连接到服务器查看运行结果即可。3.可以利用分布式部署多台服务器同时进行爬虫，效率高。

当然，部署到服务器端也有以下缺点：1.学习成本较高，需要掌握Linux、Vim基本操作。2.不是学生优惠的话，贵。

在这一部分中，大家可以通过自己动手部署上一部分的代码至服务器，了解Linux的基本知识，掌握Linux爬虫环境的搭建，学习Linux的实用指令。

建议大家自己动手，将附件的代码**bingSearch.py**部署到服务器并运行。

——*Step1: ECS*服务器的购买

目前市场上主流的ECS服务器供应商为阿里云、腾讯云，这两者都针对学生推出了优惠服务器，约10元/月/台，建议大家先买一个月熟悉下操作！

无论是阿里云还是腾讯云，我们都选择Linux的CentOS发行版（注意⚠️，请勿购买成Windows Server），因为CentOS对nginx等部署支持较好，大家买来的服务器不想做爬虫了也可以用来搭建自己的网站。

购买服务器教程如下：

阿里云：https://www.jianshu.com/p/8520b17c317f

腾讯云：https://cloud.tencent.com/developer/article/1469642

关键词：CentOS！ 一个月！请勿弄错！

——*Step2:* 远程连接至云服务器

买来的服务器，如果不能远程连接上并操纵，无异于白买。

阿里云和腾讯云在控制台中，都提供了类似“一键直连”的选项：

https://cloud.tencent.com/document/product/213/5436?fromSource=gwzcw.1293314.1293314.1293314&cps_key=806a34e58199d2e0ccdf9a10ef0ba6ac

但这种方法不如ssh连接来的方便。对于Windows用户，想要使用ssh连接到服务器，需要先安装openSSH工具。Mac用户直接使用Terminal即可。

Lecture7

Windows下安装openSSH可见：

https://blog.csdn.net/wm609972715/article/details/83759114

配置完毕后，Windows在cmd中，Mac在terminal中，执行命令：

ssh root@xxx.xx.xx.xxx

其中，ssh命令表示远程连接服务器，root为服务器中你的用户名（虽不是很建议，但大家用超级用户root即可），xxx.xx.xx.xxx为你的服务器的公网ip。

回车后，服务器会询问你root用户对应的密码，敲入并回车即可。（你没法看到你键入了密码，这是Linux的安全特性，但大胆键入就行）

但密码从何而来呢?

对于阿里云，你需要在登陆服务器之前，自行修改登录密码：

https://www.cnblogs.com/e0yu/p/11811670.html

对于腾讯云，密码会在购买完成后，以站内信的形式发送给你：

云服务器创建成功

尊敬的腾讯云用户，您好！
您（账号ID：100009707601，昵称：Fowill）的云服务器（共1台）已创建成功（订单号：20201104601000438774691），感谢您对腾讯云的支持！
服务器操作系统为 CentOS 7.5 64位，默认账户为 root，初始密码为 6]w78zcf(YP@U

顺利登陆后，可以看到下图，我们顺利接入了阿里云服务器：

```
Last login: Fri Nov  6 15:17:01 on ttys003
(base) liangyuwu@wuliangyudeMacBook-Pro ~ % ssh root@139.196.105.175
root@139.196.105.175's password:

Welcome to Alibaba Cloud Elastic Compute Service !

Activate the web console with: systemctl enable --now cockpit.socket

Last failed login: Fri Nov  6 15:17:20 CST 2020 from 39.144.40.27 on ssh:notty
There were 2 failed login attempts since the last successful login.
Last login: Fri Nov  6 12:33:47 2020 from 39.144.40.27
[root@iZuf6b1fl0swvxr9gloib4Z ~]#
```

——*Step3*：配置*Python*开发环境

首先，我们使用passwd命令修改登录密码，因为腾讯云给的密码太难用了：

```
[root@iZuf6b1fl0swvxr9gloib4Z ~]# passwd
更改用户 root 的密码 。
新的 密码：
重新输入新的 密码：
passwd: 所有的身份验证令牌已经成功更新。
```

（Linux的操作方式就和古老的DOS一样，是以命令的形式操作的，大家从图形界面迁移过来，可能需要花一段时间适应，但习惯后会上瘾的～我也会在篇末总结常用的命令，大家看教程时有不了解的地方可以对应查找）

我们可以看到，CentOS是内置了Python的，但是为Python2，所以我们要安装Python3.

```
[root@izbp17dkl0049tgohpw8ksz ~]# python
Python 2.7.5 (default, Nov  6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
[root@izbp17dkl0049tgohpw8ksz ~]#
```

安装Python3的教程：

https://cloud.tencent.com/developer/article/1478216

安装了Python3和pip3后，我们使用pip3命令安装常用的模块，如selenium、pandas等。

至此，Python3环境在CentOS上的配置完成。

——Step4: Chrome和chromedriver的安装

由于本次要部署的代码涉及到selenium操纵浏览器，所以我们也需要安装Chrome和chromedriver。

安装Chrome的方法如下（只看一、二部分即可）：

https://blog.csdn.net/weixin_44322234/article/details/105800477

安装完毕Chrome后，使用google-chrome —version命令检查Chrome版本：

```
[root@izbp17dkl0049tgohpw8ksz ~]# google-chrome --version
Google Chrome 86.0.4240.111
```

对于chromedriver，我们不在服务器端下载，而是采用sftp上传的方式，把chromedriver从我们的电脑上传到服务器端。

如果你和上图的Chrome版本一致（概率较大），那么恭喜，我会在附件里提供一个chromedriver，你可以跳过寻找过程。

如果不是，我们要在 https://chromedriver.storage.googleapis.com/index.html 中找到符合自己服务器端Chrome版本的driver，注意⚠️，请下载linux版本，并解压好。

我们要先在服务器上建立对应的文件夹，方便文件的存储，这一步建议大家先断开并重连服务器，按照以下步骤操作：

```
[root@iZuf6b1fl0swvxr9gloib4Z ~]# ls
bin  get-pip.py  python  python-3.8  python3.8  Python-3.8.0a3.tar.xz
[root@iZuf6b1fl0swvxr9gloib4Z ~]# cd ../
[root@iZuf6b1fl0swvxr9gloib4Z /]# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
[root@iZuf6b1fl0swvxr9gloib4Z /]# mkdir chromedriver
[root@iZuf6b1fl0swvxr9gloib4Z /]# ls
bin   chromedriver  etc   lib    media  opt   root  sbin  sys  usr
boot  dev           home  lib64  mnt    proc  run   srv   tmp  var
[root@iZuf6b1fl0swvxr9gloib4Z /]# mkdir projects-linkedin
[root@iZuf6b1fl0swvxr9gloib4Z /]# ls
bin           dev   lib    mnt   projects-linkedin  sbin  tmp
boot          etc   lib64  opt   root               srv   usr
chromedriver  home  media  proc  run                sys   var
[root@iZuf6b1fl0swvxr9gloib4Z /]#
```

如上图，重新连接后，我们先执行cd ../命令，返回到当前目录的上级目录(cd命令代表切换目录，../表示上级)，执行ls命令可以查看当前目录下的子目录和文件，事实上，当前的目录就是整个服务器的根目录（包含了bin、dev、sys等Linux系统的核心目录）。紧接着，我们使用mkdir命令分别新建了chromedriver文件夹和projects-linkedin文件夹。通过ls命令可以看到文件夹已经被顺利创建。

接下来使用sftp将文件传输到服务器上。

对于Windows用户，推荐使用XShell工具完成ssh连接或sftp连接：

https://cloud.tencent.com/developer/article/1678570

在sftp连接下，我们将准备好的chromedriver文件传入服务器的chromedriver文件夹中，将代码文件bingSearch.py传入服务器的projects-linkedin文件夹中。

对于Mac用户，使用Terminal自带的远程连接工具，使用put命令上传文件，如：

put /Users/liangyuwu/Documents/Projects/Linkedin/bingSearch.py /projects-linkedin

put后包括两个路径，第一个为本机想要上传的路径，另一个为服务器端接收路径。

至此，Chrome的配置和相关代码上传完成。

——*Step5:* 代码的运行

这时，代码已经可以在服务器端运行，我们使用cd命令切换到projects-linkedin目录下，执行 python3 bingSearch.py 命令即可让程序运行。

按Ctrl+C，可以中断代码运行。

虽然代码已经成功运行，但这样还是会有很多弊端，例如，断开和服务器的连接后，服务器上的代码就会终止运行。这是很蠢的一件事，因为这样，服务器就失去了它全部的意义了，如果合上笔记本，代码就终止，我还要服务器干什么。

为了解决这一问题，我们使用screen工具和nohup命令。

Lecture7

screen工具需要安装：http://linux.it.net.cn/CentOS/fast/2015/0705/16122.html

安装完毕后，我们使用cd命令切换到projects-linkedin目录下，执行screen命令，就进入了一个新的会话页面，我们可以使用exit命令退出会话，也可以用Ctrl+A 再按D保留该会话并退出，这样下次就可以继续访问。

在这个会话页面中，我们再执行如下命令：

nohup python3 -u bingSearch.py >Log1.out 2>&1 &

其中，nohup表示挂起至后台，>Log1.out表示将程序的输出全部导入Log1.out日志文件中，2>&1表示将错误和正常输出全放到同一个文件即Log1.out中，结尾的&表示静默执行，-u表示取消Python的缓冲机制，这样一有输出就会被立即更新至Log1.out。

```
[root@izbp17dkl0049tgohpw8ksz projects-linkedin]# nohup python3 -u bingSearch-1.
py >Log3.out 2>&1 &
[1] 23206
[root@izbp17dkl0049tgohpw8ksz projects-linkedin]#
```

执行完该命令后，用Ctrl+A 再按D保留该会话并退出。这样即使断开链接，Python程序也会不断运行下去了。

如何检查Python的运行情况呢?

我们可以使用top命令检查当前的cpu使用情况，看看有没有chrome进程占用cpu，可以用Ctrl+C中断top命令：

```
top - 16:32:54 up 8 days, 15:25,  4 users,  load average: 0.58, 0.51, 0.72
Tasks: 291 total,   2 running, 280 sleeping,   0 stopped,   9 zombie
%Cpu(s): 12.0 us, 18.9 sy,  0.0 ni, 66.4 id,  2.3 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem :  1883724 total,    76972 free,  1688284 used,   118468 buff/cache
KiB Swap:        0 total,        0 free,        0 used.    40708 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
   25 root      20   0       0      0      0 S 12.3  0.0  74:20.18 kswapd0
 1285 root      10 -10  134228   5968      0 S  5.3  0.3  98:51.13 AliYunDun
11545 root      20   0  200808   5340   3496 S  3.3  0.3   0:00.17 chromedriver
11573 root      20   0  200808   5200   3440 S  2.6  0.3   0:00.10 chromedriver
11610 root      20   0  683668  38140  26192 S  2.0  2.0   0:00.10 chrome
11635 root      20   0  579872  23572   6460 S  1.3  1.3   0:00.04 chrome
11638 root      20   0  574820  32516  21980 S  1.3  1.7   0:00.04 chrome
11640 root      20   0 4812344  28736  15432 S  1.0  1.5   0:00.03 chrome
 7135 root      20   0  677728  17192    952 S  0.7  0.9   1:51.90 chrome
    9 root      20   0       0      0      0 R  0.3  0.0   1:13.93 rcu_sched
11546 root      20   0  157840   1620    804 R  0.3  0.1   0:00.02 top
11619 root      20   0  525516  29408  19352 S  0.3  1.6   0:00.03 chrome
    1 root      20   0   43116   1480    348 S  0.0  0.1   0:10.08 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.03 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:05.83 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
```

我们也可以用 ps -ef | grep python 命令查看所有名字包含python的进程是否存活：

```
[root@izbp17dkl0049tgohpw8ksz ~]# ps -ef | grep python
root      5393  5382  0 11:02 pts/5    00:00:14 python3 -u bingSearch-1.py
root      5493  5478  0 11:02 pts/6    00:00:13 python3 -u bingSearch-2.py
root     11784 11529  0 16:34 pts/9    00:00:00 grep --color=auto python
[root@izbp17dkl0049tgohpw8ksz ~]#
```

o

由于我们运行时还指定了Log1.out作为日志文件，因此我们还可以查看日志来监视运行情况。执行 vim Log1.out 命令即可。

Vim是一款集成于Linux上的文件编辑软件。我们可以使用i键进入编辑模式，使用esc返回到阅览模式。阅览模式下按:可以进入底命令行模式，例如:q表示退出，:wq表示保存后退出。

其他基本操作参考：

https://www.runoob.com/linux/linux-vim.html

至此，我们完成了代码在Linux上的部署，以后就可以随时通过ssh登录服务器，查看我们爬虫的进展了。

——常见*Linux*命令对照表

前几部分突然涉及了大量*Linux*命令，可能会给大家带来困扰，这里单独列出如下，供查询和参考：

命令名	作用	例子	解释
cd	切换路径	cd ../python	切换到当前父目录下的python文件夹
ls	列出当前路径下的文件夹/文件		
mkdir	新建文件夹	mkdir python	新建一个python文件夹
touch	新建文件夹	touch abc	新建一个文件abc
rm	删除文件（删库跑路）	rm python	删除python文件
ssh	远程连接	ssh root@xxxxx	以root用户连接到xxx服务器
yum	CentOS自带的安装软件的工具，类似pip与conda	yum install screen	安装screen软件
wget	下载文件的工具	wget https://xxxxxxx	从对应网站下载文件
tar	解压	tar xxxx.tgz	解压xxxx.tgz文件
screen	新建一个会话		
top	查看cpu使用情况		
ps	查看所有进程	ps -ef | grep python	查看名字里带有python的进程
vim	使用vim查看和编辑文件	vim bingSearch.py	查看和编辑这个Python脚本

当然，我的解释肯定不如菜鸟教程权威：

https://www.runoob.com/linux/linux-command-manual.html

4.爬虫的进阶路线

最后给出一些爬虫方面继续学习的建议。

关于知识结构，爬虫主要包括Web端爬虫、App端抓包爬虫这两类，为了成功进行这两类爬虫，我们需要掌握代理技术、验证码破解技术、身份混淆技术、逆向工程技术，为了高效进行这两类爬虫，我们需要掌握建立在服务器上的分布式爬虫技术。目前我们仅仅对Web端爬虫和服务器爬虫做了一些了解，感兴趣的同学可以探索上面提到的其他方向。

关于学习材料，我强烈推荐崔庆才的《Python3网络爬虫开发实战》，这本书包含了绝大多数常见的爬虫场景（逆向工程除外，这也决定了这本书不能解决所有问题），但内容还是很完善的。

关于学习建议，推荐大家评估自己的需求后，再根据实际需要学习。因为几个金融App我尝试过爬取，但其实难度还是挺高的，常规的抓包都不管用，基本都需要上逆向，这个学习曲线太过陡峭，我们专业所需的数据很多时候都不太好通过爬虫获得。