Web Scraping Guide (for absolute novices)

06 Aug 2018

In this document, I provide a step-by-step guide to web scraping. I started from scratch with no understanding of either Python or web scraping, so this manual is for true beginners.

The document is organized as follows:

1. **Overview of the scraping process**: What is web scraping? What are we setting off to do?
2. **What to download**: Links and descriptions of all the tools that we will need to download
3. **Working example**: I had to learn how to scrape websites for a project for my professors Claudia Goldin and Claudia Olivetti. The data is publicly accessible (and no lawyers have called me yet!) so I will walk you through the process of building my dataset.

Please note: I am not an expert at any of the things I describe in the rest of this document. I am sure much of what follows can be done faster or better. This is just how I learned to scrape in the past 3 days. If I find better ways to do things, I will update this document or write a new one!

Overview of the scraping process

1. "Inspect" websites – in order to understand what we are about to embark on, go ahead and open a new Chrome window. I prefer to use Chrome but obviously you can use other browsers. The rest of this documentation assumes you are using Chrome, but I believe any other browser will work similarly. You can do the following steps with any website, but use this url so we can work through an example together

2. When you right-click “inspect,” your Chrome window will split into two. What you see is HTML code (with some Javascript, but we will deal with that later).

3. At the very top, you will see $<head>…</head>$. This is an example of an HTML tag. You will also see several other tags: $<div>…</div>$, $<body>…</body>$, etc. All the stuff in a web page is essentially written in between such tags. Whenever you find a tag such as $<div>$, you will also find its ending partner $</div>$ with a bunch of text in between.

4. HTML is basically made up of such tags nested in each other.

5. Tags can have attributes. Very commonly, the div tag will either have class or id as an attribute like the following $<div$ $class = “this”>$. We will use these attributes to access specific tags. Play around! Click on the small grey triangles on the left of these tags to open and collapse them and see what text there is between them.

6. For a more detailed (but novice-friendly) introduction to HTML, go here.

The task in three steps:

1. Get the HTML code from a website’s URL.
2. Inspect the HTML code to find where the information that we want to scrape is hidden.
3. Using Python, scrape the information we need and export it to an excel sheet to build your dataset.

What to download

1. Python: Download Anaconda
   - Anaconda lets you download the latest version of Python and other packages that you will need later. I downloaded Python 3.6.
   - When you download Anaconda, you also download a bunch of other *things* (packages and such) that you will need while working with Python. Fun fact: I am writing this documentation in *Jupyter*, which was downloaded automatically with Anaconda.
   - Later, if you want to check whether you have a package, click on Environments from the menu on the left. On your right, you will see a giant scrollable list. In the search bar that says “search packages” type the name of your package to check if it was automatically installed.
   - Link: https://www.anaconda.com/
2. Text editor: Download Atom
   - Since you will be writing a bunch of code, you need a text editor that lets you save all the

code. (For STATA users, Atom lets you make a "do file.") You can choose to use any, of course, but my friend Kyle recommended this one to me when I embarked upon learning Python and it's the one I have been using ever since!
    - Link: https://atom.io/
    - Various python packages
3. Have you ever used Latex? If you have, you know how you have to install packages for specific things you need to do, like include figures or write math? In Python, you install packages from which you import functions.
    - **BeautifulSoup**: This is the most important Python package we will use. BeautifulSoup accesses a URL, saves its HTML code, and lets you find "tags" and other elements in the HTML code
    - We will discuss how to download these packages later

Using the Terminal

Note: I use a Mac and the rest of this documentation applies to using the Mac Terminal.

So, what is the Terminal and how do you access it? Go ahead and type "Terminal" in your spotlight. Terminal is just an application on your computer so don't let it intimidate you. The terminal allows you to write commands and interact with the operating system of the computer directly.

Let's try out a few things! Type "pwd" and hit return/ Enter. The terminal will return the file path that you are in. Now type "ls". This command will list all the folders in your current destination. Do you want to go somewhere else? Type "cd" followed by the name of one of the folders that came up when you typed "ls". For a more thorough description, go here.

I stored all the files related to this project in the path /Users/namratanarain/Dropbox/Goldin/WebScraping. So I type "cd" /Users/namratanarain/Dropbox/Goldin/WebScraping. I type "ls" and see all the files in this folder. Great.

Python

Check what version of Python you have on your computer (it should be 3.6 if you followed the Anaconda step above.) You can do this by simply typing "Python" in your terminal. My terminal returns "Python 3.6.5 |Anaconda, Inc.|". Good!

Now, anything you type will be interpreted as Python code. If you want to exit and go back to the terminal, hit contrl-D.

We will be going back and forth between working in the Python environment and hitting ctrl-D to work out of the environment.

When you want to execute (run) a file with an extension .py, type "python filename.py" in your terminal. Make sure you are in the right folder first!

If you want to run a Python command on its own, go into the Python environment. Remember that each time you go into the Python environment, you will have to import packages all over again. Just copy paste from your helpful Atom editor. (I do not yet know how to execute commands line-by-line directly from Atom.)

Scraping

- Open Atom. Go to File > New File. Save this file in your folder as scraping.py. The extension .py is important because it tells Atom that the code in this file should be treated like Python code.
- Go ahead and type "pip install selenium" in your terminal (*outside* the Python environment.) Selenium is a Python package that we will need later.
- Copy paste the following lines of code. Each line of code imports a package.

```
from bs4 import BeautifulSoup, SoupStrainer
import re
import requests
from pprint import pprint
from selenium import webdriver
import xlsxwriter
from openpyxl import load_workbook
import time

start_time = time.time()
```

- Note that I use the "time" module to see what time it is when the code starts running – we will use this to calculate the length of time it takes for the code to execute. This is useful if you are running a for loop and want to know how long the whole program will take to run!
- Great, now let's pass in the url that we want to scrape.

```
url = "http://reviews.greatplacetowork.com/salesforce"
```

- The following two lines of code first (1) request the url, and then (2) download the html content of the page. This content is now saved in the variable soup.
- For fun, type each line of code so far in the Python environment to see what each command is doing. Typing soup will print a jumble of html gibberish on your screen. soup.prettify() will format it to make it more readable!

```
page = requests.get(url)
soup = BeautifulSoup(page.content,'html.parser')
print(soup.prettify())
```

- So, when I did the above^, I got stuck: some of the things I wanted to scrape, like the percent of

female executives, were presented as graphs!
- These figures are rendered in JavaScript – I had to look for a first step to convert the JavaScript into HTML (this is not exactly what happens, but is close enough.)
- This is where the webdriver comes in handy! Go to this stackexchange page to read more.
- You will need selenium (which you've already installed!) to run the webdriver.
- When you run this, a new Chrome window will pop up with your url.

```
browser = webdriver.Chrome("/Users/namratanarain/Dropbox/Goldin/WebScraping/chromedriver.exe")
browser.get(url)
html = browser.page_source
soup = BeautifulSoup(html,"lxml")
browser.quit()
```

- We are basically all done!
- The only commands you really need to know are .find and .get_text
- soup.find searches the output stored in soup to find the tags given in the parentheses. In the following command, we look for the title tag – $$ – in the soup. I had inspected the webpage to see that this is where the firm name was hidden.
- get_text gets you the text within the tags $$.
- Sometimes, you'll get lots of spaces like /t or blank lines /n in your text. Typing strip=True in parentheses strips the text of these unnecessary characters

```
name = soup.find("title").get_text()
name = soup.find("title").get_text(strip=True)
```

- Important: I was scraping several websites that were *similar* but not exactly the same. Concretely, some had the "industry" information but others did not. Python lets you use a try-except code to try a code, and if it doesn't work, do something "else".
- I use try-except to assign empty values to variables since I will be later passing on everything to Excel

```
# preliminary information
try:
    name = soup.find("title").get_text()
except Exception:
    name = ""
```

- Okay, one last point before I let you read through the rest of the code on your own: go here to learn the basics of BeautifulSoup. Seriously, it is SO helpful. Play around with it!
- Remember that .find returns the first element that matches your criterion. On the other hands, .find_all returns the entire LIST of all elements that match your criterion. As in all lists, you can use indexing to access a particular element. For example, the following code will return the first tag of the type $<h3>…</h3>$ from the several that exist on this website.

```
soup.find_all("h3")[0]
```

- I will let you figure out the rest of the code. Google is your best friend – I learned all of this by repeatedly Googling my problems. Python and BeautifulSoup are extremely well documented!
- At the bottom of the code, I describe how to export these values to an Excel sheet.

```
# About This Company
try:
    searchtext = re.compile(r'about this company',re.IGNORECASE)
    foundtext = soup.find('h3',text=searchtext)
    table = foundtext.findNext('table')
except Exception:
    pass

# Industry
try:
    indtext = re.compile(r'industry',re.IGNORECASE)
    indfoundtext = table.find('span',text=indtext)
    industry = indfoundtext.next_sibling
except Exception:
    industry = ""

# Headquarters
try:
    hqtext = re.compile(r'headquarters',re.IGNORECASE)
```

```
    hqfoundtext = table.find('span',text=hqtext)
    headquarters = hqfoundtext.next_sibling
except Exception:
    headquarters = ""

# Founded
try:
    fdtext = re.compile(r'founded',re.IGNORECASE)
    fdfoundtext = table.find('span',text=fdtext)
    founded = fdfoundtext.next_sibling
except Exception:
    founded = ""

# WorldEmployees
try:
    wetext = re.compile(r'employees worldwide',re.IGNORECASE)
    wefoundtext = table.find('span',text=wetext)
    wdemployees = wefoundtext.next_sibling
except Exception:
    wdemployees = ""

# USEmployees
try:
    uetext = re.compile(r'us employees',re.IGNORECASE)
    uefoundtext = table.find('span',text=uetext)
    usemployees = uefoundtext.next_sibling
except Exception:
    usemployees = ""

# Stock Symbol
try:
    sttext = re.compile(r'stock symbol',re.IGNORECASE)
    stfoundtext = table.find('span',text=sttext)
    stocksymbol = stfoundtext.next_sibling
except Exception:
    stocksymbol = ""

# Revenues
try:
    revtext = re.compile(r'worldwide revenues',re.IGNORECASE)
    revfoundtext = table.find('span',text=revtext)
    revenue = revfoundtext.next_sibling
except Exception:
    revenue = ""

# Corporate Structure
try:
    corptext = re.compile(r'corporate structure',re.IGNORECASE)
    corpfoundtext = table.find('span',text=corptext)
    corpstructure = corpfoundtext.next_sibling
except Exception:
    corpstructure = ""

# Family Care by the Numbers
try:
    searchtext = re.compile(r'family care by the numbers',re.IGNORECASE)
    foundtext = soup.find('h3',text=searchtext)
    table = foundtext.findNext('table')
except Exception:
    pass

try:
    matleave = re.compile(r'protected maternity',re.IGNORECASE)
    #pprint(soup.find(text = matleave).__dict__)
    maternitydays = table.find(text = matleave).previous_element
except Exception:
    maternitydays = ""
try:
    matfull = re.compile(r'fully-paid maternity',re.IGNORECASE)
    matfulldays = table.find(text = matfull).previous_element
except Exception:
    matfulldays = ""

try:
    patleave = re.compile(r'protected paternity',re.IGNORECASE)
```

```
    paternitydays = table.find(text = patleave).previous_element
except Exception:
    paternitydays = ""

try:
    patfull = re.compile(r'fully-paid paternity',re.IGNORECASE)
    patfulldays = table.find(text = patfull).previous_element
except Exception:
    patfulldays = ""

try:
    adopt = re.compile(r'protected parental leave for adoptive parents',re.IGNORECASE)
    adoptdays = table.find(text=adopt).previous_element
except Exception:
    adoptdays = ""

try:
    adoptfull = re.compile(r'paid parental leave for adoptive parents',re.IGNORECASE)
    adoptfulldays = table.find(text=adoptfull).previous_element
except Exception:
    adoptfulldays = ""

try:
    adoptdoll = re.compile(r'adoption benefit',re.IGNORECASE)
    adoption = table.find(text=adoptdoll).previous_element
except Exception:
    adoption = ""

    # Averages
try:
    matavg = re.compile(r'length of maternity',re.IGNORECASE)
    matavgdays = table.find(text=matavg).previous_element
except Exception:
    matavgdays = ""

try:
    patavg = re.compile(r'length of paternity',re.IGNORECASE)
    patavgdays = table.find(text=patavg).previous_element
except Exception:
    patavgdays = ""

# Leadership
# get full time
try:
    ft=re.compile(r'full time',re.IGNORECASE)
    fulltime = soup.find(text=ft).previous_element.get_text()
except Exception:
    fulltime = ""

    # get female executives
try:
    searchtext = re.compile(r'female executives',re.IGNORECASE)
    foundtext = soup.find('text',text=searchtext) # Find the first <p> tag with the search text
    table = foundtext.findNext('table')

    pctwom_exec = table.find_all(text=re.compile(r"Women"))[1]
except Exception:
    pctwom_exec = ""

    # get mid-level managers
try:
    searchtext = re.compile(r'female mid-level managers',re.IGNORECASE)
    foundtext = soup.find('text',text=searchtext) # Find the first <p> tag with the search text
    table = foundtext.findNext('table')

    pctwom_mang = table.find_all(text=re.compile(r"Women"))[1]
except Exception:
    pctwom_mang = ""

    # frontline managers and supervisors
try:
    searchtext = re.compile(r'female frontline managers or supervisors',re.IGNORECASE)
    foundtext = soup.find('text',text=searchtext) # Find the first <p> tag with the search text
    table = foundtext.findNext('table')
```

```
    pctwom_sup = table.find_all(text=re.compile(r"Women"))[1]
except Exception:
    pctwom_sup = ""

    # workforce percentage
try:
    searchtext = re.compile(r'gender',re.IGNORECASE)
    foundtext = soup.find('text',text=searchtext) # Find the first <p> tag with the search text
    table = foundtext.findNext('table')

    pctwom_all = table.find_all(text=re.compile(r"Female"))[0]
except Exception:
    pctwom_all = ""

# List of Awards
try:
    soup.find("h2",text=re.compile(r'has been awarded'))
    soup.find("div",{"class" : "awards"})
    list = ""
    for i, tag in enumerate(soup.find("div",{"class" : "awards"}).find_all("a")):
        if i != len(soup.find("div",{"class" : "awards"}).find_all("a"))-1:
            award = tag.get_text()
            list = list + "," + award

    awardlist = list[1:]
except Exception:
    awardlist = ""

# Related Firms
try:
    list = ""
    for tag in soup.find_all("div", {"class" : "company_image"}):
        imgs = tag.find_all("img")
        related = imgs[0]['title']
        list = list + "," + related
    relatedfirms = list[1:]
except Exception:
    relatedfirms = ""

# Date of survey
def find_between( s, first, last ):
    try:
        start = s.index( first ) + len( first )
        end = s.index( last, start )
        return s[start:end]
    except ValueError:
        return ""

try:
    datesen = soup.find("div",{"class" : "review_text"}).get_text(strip=True)
    datesurvey = find_between(datesen,"was published on",".")
except Exception:
    datesurvey = ""

### Work Life Balance by the Numbers
try:
    wlbalance = re.compile(r'work-life balance by the numbers',re.IGNORECASE)
    ftxt = soup.find("h3",text=wlbalance)
    table = ftxt.findNext("table")
except Exception:
    table = ""

# paid time off - full time workers
try:
    stext = re.compile(r'paid time off after one year of full-time',re.IGNORECASE)
    paidofffull = table.find(text=stext).previous_element
except Exception:
    paidofffull = ""

# paid time off - part time workers
try:
    stext = re.compile(r'paid time off after one year of part-time',re.IGNORECASE)
    paidoffpart = table.find(text=stext).previous_element
except Exception:
    paidoffpart = ""
```

- Now that we have stored all the variables we need, let's export them to Excel
- I have already initialized an Excel file (the .py code for that file is in the main directory)

```
# Export information to Excel
wb = load_workbook("/Users/namratanarain/Dropbox/Goldin/WebScraping/gptw1.xlsx")
ws = wb.worksheets[0]

row = [name,founded,headquarters,industry,usemployees,wdemployees,
    maternitydays,matfulldays,matavgdays,paternitydays,patfulldays,pataavgdays,
    adoptdays,adoptfulldays,adoption,corpstructure,revenue,pctwom_exec,
    pctwom_mang,pctwom_sup,pctwom_all,awardlist,relatedfirms,stocksymbol,
    datesurvey,paidofffull,paidoffpart]
print(row)
ws.append(row)
wb.save("/Users/namratanarain/Dropbox/Goldin/WebScraping/gptw1.xlsx")
time.sleep(2)
```

- I print the number of seconds it took me to run this code so I know how much time it will take me to scrape *n* websites.

```
print("--- %s seconds ---" % round((time.time()-start_time)))
```

Building the dataset

I wanted to scrape all this information for all the companies on this website. Kyle told me that all (?) websites have a site map (map of the "backend") where I can access all the websites. url: http://reviews.greatplacetowork.com/index.php?option=com_gptw&view=sitemap&format=xml.

Inspect the site carefully! All the urls I need are stored in $$ tags. I made a list of all the $$ tags on this site and looped through each. How would you do this?

```
page = requests.get("http://reviews.greatplacetowork.com/index.php?option=com_gptw&view=sitemap&format=x
soup = BeautifulSoup(page.content,'html.parser')
firms = soup.find_all('loc')

# for each firm, extract data - 761 firms
firmslist = list()
for allfirms in soup.find_all('loc'):
    firmslist.append(allfirms.get_text())

    url = allfirms.get_text()

    ### DO THE REST OF THE CODE IN THIS LOOP
```

- The list firmslist will now contain the list of all urls.
- You can count the number of firms by simply typing len(firmslist).

I hope this helped! Good luck!

Please direct all queries to nnarains@gmail.com.

---

email **twitter** **github**