

Web Scraping and Text Extracting Guide

Lucas Rosso
Universidad de Chile

July, 2021

Outline

- Installation and brief introduction
- What is web scraping?
- Basic example
- Scraping dynamic websites
- Extracting text

Installing Python

The best option for beginners is Anaconda

- Enter `https://www.anaconda.com/download/` to download
- Just follow the steps to install
 - **Note:** In advanced options set “Anaconda3 as my default Python”
- Open Anaconda and launch Spyder

Libraries

- Python works with libraries
- Some are included in Anaconda, but not all
- To download a new library just type:

```
pip install package
```

Spyder

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\LR\Desktop\ME\Programación\Python\RA_jobs_example.py

```
1 '''
2 Example code for basic web-scraping using beautiful soup.
3 Author: Lucas Rosso
4 Date: July, 2021
5 '''
6
7 # Preliminaries
8 from os import chdir
9 chdir ("C:/Users/LR/Desktop/ME/Programación/Python") # (this must change)
10
11 from bs4 import BeautifulSoup
12 import requests
13 import pandas as pd
14
15 # Not-NBER positions
16 url = "https://www.nber.org/career-resources/research-assistant-positions-not-nber"
17 page = requests.get(url, 'html.parser')
18
19 soup = BeautifulSoup(page.text, features='lxml')
20
21 jobs = soup.find('div', class_='page-header__intro-inner').find_all('p')
22 jobs = jobs[1:len(jobs)-2] # delete introduction and closing paragraphs
23
24 pos_name = []
25 researcher = []
26 link = []
27 for job in jobs:
28     # position name
29     pos_name = job.text.split('\n')[0]
30     pos_name.append(pos_name)
31
32     # researcher
33     try:
34         researcher_ = job.text.split("\n")[1].split(";")[1]
35     except:
36         researcher = job.text.split("\n")[1].split("\x00")[1]
37         researcher.append(researcher)
```

Name	Type	Size	Value
df	DataFrame	(82, 3)	Column names: pos_name, researcher, link
field	list	33	[' Applied Microeconomics, Economics of Education, Health Economics, L ...
field_	str	1	University of Chicago, Center for the Economics of Human Development
id	list	0	[]
id_	int	1	183724
institution	list	34	[' University of Notre Dame, Wilson Sheehan Lab for Economic Opportuni ...
institution_	str	1	Poverty, Safety net, Inequality, Homelessness

Variable explorer Help Plots Files Profiler Code Analysis

Console 1/A

```
Out[33]: 'Link for Job Posting'
In [34]: job.text.split("\n")[2]
Out[34]: 'Field(s) of Research: Poverty, Safety net, Inequality, Homelessness
Institution: University of Chicago, Harris School of Public Policy (Comprehensive Income
Dataset Project)'
In [35]: job.text.split("\n")[1]
Out[35]: 'NBER Sponsoring Researcher(s): Bruce Meyer'
In [36]: job.text.split("\n")[0]
Out[36]: 'Pre-Doctoral Research Fellow'
In [37]: runfile('C:/Users/LR/Desktop/ME/Programación/Python/RA_jobs_example.py',
wdir='C:/Users/LR/Desktop/ME/Programación/Python')
In [38]:
```

LSP Python: ready conda: base (Python 3.8.5) Line 34, Col 60 UTF-8 CRLF RW Mem 72%

Setting the Working Directory

- From the library `os`, import the function `chdir` to change the working directory

```
from os import chdir
chdir(my_path)
```

- **Note:** the path must be written in one of the following ways:

```
os.chdir("C:\\Users\\LR") # double backslash
os.chdir("C:/Users/LR") # regular slash
```


A Basic Example: NBER RA Job Postings

We will use web scraping to create a dataframe with the open positions for RA posted at the [NBER](#).



[RA POSITIONS – AT NBER](#) [RA POSITIONS – NOT AT THE NBER](#) [STAFF POSITIONS AT NBER](#) [CALLS FOR FELLOWSHIP APPLICATIONS](#)
[CURRENT FELLOWSHIP RECIPIENTS](#) [PHD CANDIDATES IN ECONOMICS](#)

This page provides links to full-time job listings associated with research projects led by NBER-affiliated researchers, but which do not involve employment through NBER. Similar positions that do involve NBER employment may be found [here](#). The positions listed below may be suitable for bachelor's degree candidates, graduate students, or post-docs. Positions typically have an expected duration of 1 to 3 years, and often give priority to candidates who are planning to continue to the next stage of their academic career upon completion of the position. This page does not include listings for jobs that utilize economic analysis for purposes other than academic research, such as postings for economics consulting firms and financial services firms. University career offices typically manage these job announcements.

Research Associate

NBER Sponsoring Researcher: Bill Evans

Institution: *University of Notre Dame, Wilson Sheehan Lab for Economic Opportunities (LEO)*

Fields of Research: Applied Microeconomics, Economics of Education, Health Economics, Labor Economics, Public Economics

[Link for Job Posting](#)

Pre-doc Research Associate, University of Notre Dame

NBER Sponsoring Researcher(s): Robert Collinson, Eric Chyn (Dartmouth College)

Fields: Public Economics, Labor Economics, Urban Economics

Institution: *University of Notre Dame, Department of Economics*

[Link for Job posting](#)

Full-time research analyst (pre-doctoral)

NBER Sponsoring Researcher(s): Lori Beaman

Institution: *Global Poverty Research Lab, Northwestern University*

Field(s) of Research: Development economics

[Link for Job posting](#)

Preliminaries

- For this example, we will use the most accessible tool: BeautifulSoup, a html parser
- **Main idea:** Select information by **navigating in the tree of the HTML code**
- We start by setting the working directory and downloading the libraries

```
# Preliminaries
from os import chdir
chdir ("C:/Users/LR/Desktop/ME") # (this must change)

pip install beautifulsoup4
from bs4 import BeautifulSoup # html parser
import requests # to download the page
import pandas as pd # to store the data
```

Steps

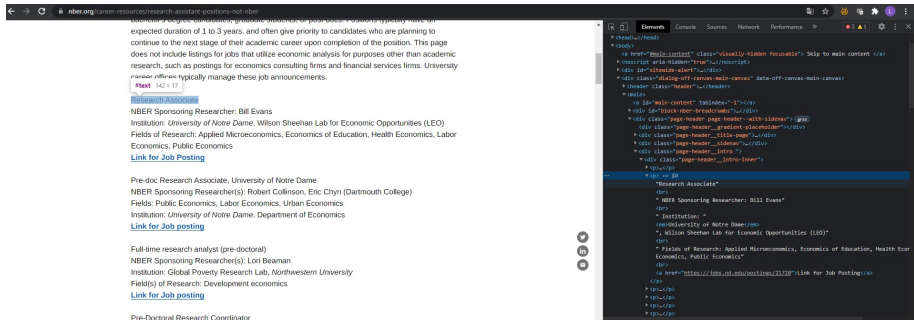
Just 3 steps to extract information:

1. Request: download the html code, which contains all the data you want
2. Parsing: parse the data to store the information you need.
3. Storing: finally, store your data.

Step 1: Request

```
# Not-NBER RA positions
url = "https://www.nber.org/career-resources/research-
      assistant-positions-not-nber"
page = requests.get(url, 'html.parser')
```

Step 2: Parse



The screenshot shows a web browser displaying a job listing page from nber.org. The page content includes a description of the position, a list of job postings, and a search bar. The developer tools are open, showing the DOM tree with the following structure:

```
<div class="page-header__intro-inner">
  <h3>Research Associate</h3>
  <p>NBER Sponsoring Researcher: Bill Evans</p>
  <p>Institution: University of Notre Dame, Wilson Sheehan Lab for Economic Opportunities (LEO)</p>
  <p>Fields of Research: Applied Microeconomics, Economics of Education, Health Economics, Labor Economics, Public Economics</p>
  <a href="#">Link for Job Posting</a>
  <p>Pre-doc Research Associate, University of Notre Dame</p>
  <p>NBER Sponsoring Researcher(s): Robert Collinson, Eric Chyn (Dartmouth College)</p>
  <p>Fields: Public Economics, Labor Economics, Urban Economics</p>
  <p>Institution: University of Notre Dame, Department of Economics</p>
  <a href="#">Link for Job posting</a>
  <p>Full-time research analyst (pre-doctoral)</p>
  <p>NBER Sponsoring Researcher(s): Lori Beaman</p>
  <p>Institution: Global Poverty Research Lab, Northwestern University</p>
  <p>Field(s) of Research: Development economics</p>
  <a href="#">Link for Job posting</a>
  <p>Pre-Doctoral Research Coordinator</p>
```

```
# we parse the web page in order to navigate the tree
soup = BeautifulSoup(page.text, features="lxml")
```

```
# we find all 'p' tags within the 'div' class 'page-
header__intro-inner'
jobs = soup.find('div', class_='page-header__intro-inner').
find_all('p')
```

```
jobs = jobs[1:len(jobs)-2] # delete introduction and closing
paragraphs
```

Step 2 (continuation): we loop over the elements of the list jobs to get the information we want

```
pos_name      = []
researcher    = []
link          = []
for job in jobs:
    # position name
    pos_name_ = job.text.split('\n')[0]
    pos_name.append(pos_name_)

    # researcher
    researcher_ = job.text.split("\n")[1].split(":")[1]
    researcher.append(researcher_)

    # job posting link
    link_ = job.find('a').get('href')
    link.append(link_)
```

Step 3: Store data and export to csv file

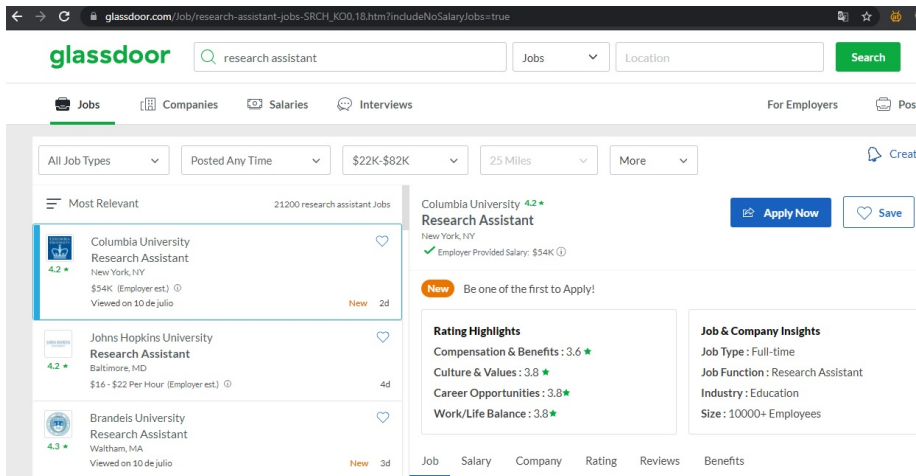
```
# Create dataframe with newly collected data
df = pd.DataFrame({
    'pos_name': pos_name,
    'researcher': researcher,
    'institution': institution,
    'field': field,
})

# save it as a csv file
df.to_csv(path_or_buf='RA_jobs.csv',na_rep='.',sep=';',index
          =False)
```

Scraping Data in Dynamic Websites

- Some websites are dynamic or have hidden information → Selenium WebDriver allows to deal with this issue
- Basically it allows to set a remote machine to browse through the website as a “human” would
- The WebDriver can click, scroll down, etc
- To use it one must have **chromedriver** inside the working directory

Example: Extracting RA jobs at Glassdoor



glassdoor.com/job/research-assistant-jobs-SRCH_K00.18.htm?includeNoSalaryJobs=true

glassdoor

research assistant

Jobs

Location

Search

Jobs Companies Salaries Interviews For Employers

All Job Types Posted Any Time \$22K-\$82K 25 Miles More

21200 research assistant Jobs

Most Relevant

Columbia University 4.2 ★
Research Assistant
New York, NY
\$54K (Employer est.) ⓘ
Viewed on 10 de julio New 2d

Johns Hopkins University 4.2 ★
Research Assistant
Baltimore, MD
\$16 - \$22 Per Hour (Employer est.) ⓘ 4d

Brandels University 4.3 ★
Research Assistant
Waltham, MA
Viewed on 10 de julio New 3d

Columbia University 4.2 ★
Research Assistant
New York, NY
✓ Employer Provided Salary: \$54K ⓘ

New Be one of the first to Apply!

Rating Highlights
Compensation & Benefits: 3.6 ★
Culture & Values: 3.8 ★
Career Opportunities: 3.8 ★
Work/Life Balance: 3.8 ★

Job & Company Insights
Job Type: Full-time
Job Function: Research Assistant
Industry: Education
Size: 10000+ Employees

Job Salary Company Rating Reviews Benefits

Downloading the libraries and setting the driver options

We start by loading the libraries

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions
    as EC
from selenium.webdriver.chrome.options import Options
import time
import pandas as pd
import numpy as np
from os import chdir
chdir("C:/Users/LR/Desktop/ME/Programaci n/Python")
```


and setting the chrome driver options.

```
# chrome webdriver options
options = Options()
options.page_load_strategy = 'normal'
options.add_argument("--start-maximized")
options.add_argument('--disable-extensions')
options.add_argument("--disable-notifications")
options.add_argument('--no-sandbox')
options.add_argument('--verbose')
options.add_argument('--disable-gpu')
options.add_argument('--disable-software-rasterizer')
```

Next Steps

Again, the process consists of 3 simple steps:

1. Identify patterns in the website (e.g. in the xpath)
2. Loop to get information
3. Store final data in a dataframe

We open the WebDriver

```
driver = webdriver.Chrome(options=options)
url = 'https://www.glassdoor.com/Job/research-assistant-jobs
      -SRCH_K00,18.htm'
driver.get(url)
```

Step 1: Identify patterns

- One can easily note the patterns that follow the institution and position name xpaths.

```
# Position name xpath: '//*[@id="MainCol"]/div[1]/ul/li['+  
                        str(i)+'']/div[2]/a/span'
```

- but how do we get the # of elements? we can count all elements by class name

```
#elements in page  
elements = driver.find_elements_by_class_name("react-job-  
listing")
```

Step 2: Loop to get information

```
opening = []
pos_name = []
for i in range(1, len(elements)+1):
    # Opening
    opening_ = WebDriverWait(driver, 10)\
        .until(EC.presence_of_element_located((By.XPATH, '
                                                    /*[@id="MainCol"]/div
                                                    [1]/ul/li['+str(i)+']/
                                                    div[2]/div[1]/a/span'
                                                    ))\
                .text
    opening.append(opening_)

    # Position name
    pos_name_ = WebDriverWait(driver, 10)\
        .until(EC.presence_of_element_located((By.XPATH, '
                                                    /*[@id="MainCol"]/div
                                                    [1]/ul/li['+str(i)+']/
                                                    div[2]/a/span'
                                                    ))))\
                .text
    pos_name.append(pos_name_)
```

Loop over pages

- Finally, one can identify the button to go the next page, and loop over pages to store all available positions.
- We only need to click the button and wait for the next page to load

```
WebDriverWait(driver, 10)\
    .until(EC.presence_of_element_located((By.XPATH, '//*[\
        @id="FooterPageNav"]/div/\
        ul/li[7]/a/span')))\
    .click()
time.sleep(np.random.randint(5,10)) # sleep for a few
    seconds

# after the loop close the driver
driver.close()
```

Step 3: Export data

As in the other example, we use the library pandas to export the data to a csv

```
# Create dataframe with newly collected data
df = pd.DataFrame({
    'pos_name': pos_name,
    'opening': opening,
})
df.to_csv(path_or_buf='RA_jobs_glassdooe.csv', na_rep='.', sep
          =';', index=False)
```