Lab 11: Web Scraping

CSE/IT 107

NMT Computer Science

"The limits of my language mean the limits of my world."

— Ludwig Wittgenstein

"Any sufficiently advanced technology is indistinguishable from magic."

— Arthur Clarke

"Imagination is more important than knowledge."

— Albert Einstein

---

1 Introduction

In this lab, you will be extracting information from web pages and learning how to plot some of it.

2 HTML

As you may be familiar, HTML is the main formatting language of the Internet.

3 Requesting a web page

The following is an example code snippet to get the source code of a particular web page:

```
1 from urllib import request
2
3 def get_page_source(webpage):
4   req = request.Request(webpage, headers={'User-Agent' : 'Python Browser'})
5   page = request.urlopen(req)
6   return page.read()
7
8 print(get_page_source('https://cs.nmt.edu/~ckoch/cse107/beautifulsoup.html'))
```

This will give you the HTML source of the web address you give it.

4 BeautifulSoup

BeautifulSoup is a library that will let you "scrape" web pages for certain information. Instead of having to deal with the HTML source of a web page directly, you can use BeautifulSoup to find information easily. If, for example, I want to find all the a elements (link elements) on a web page, I could do the following:

```
import bs4

# Using previous function
data = get_page_source('https://cs.nmt.edu/~ckoch/cse107/beautifulsoup.html')
soup = bs4.BeautifulSoup(data)

print(soup.prettify())

elements = soup.findAll('a')
print(elements)
```

Listing 1: BeautifulSoup

The output of `.prettify()` looks like this:

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   CSE107 BeautifulSoup Test
  </title>
 </head>
 <body>
  <p>
   Haha
  </p>
  <p>
   <a href="https://cs.nmt.edu/~ckoch/">
    Chris's site
   </a>
   <a href="https://arctem.com">
    Russell's site
   </a>
  </p>
 </body>
</html>
```

On this page, we have two a elements, and thus `.findAll()` returns the following list:

```
[<a href="https://cs.nmt.edu/~ckoch/">Chris's site</a>,
<a href="https://arctem.com">Russell's site</a>]
```

Now, to find just the first element, we use the `.find()` function. Also note that the above list is not a list of strings; the type of the list elements is a beautiful soup element tag. These tags have a few properties, as you can see in the following example:

```
import bs4
# calling beautiful soup with my own data
```

```
soup = bs4.BeautifulSoup('<html><body><p>Test</p><p>T</p></body></html>')

paragraph = soup.find('p') # finds <p>Test</p>
print(paragraph)
print(paragraph.string) # Test
print(paragraph.contents) # ['Test']
print(paragraph.name) # prints p

body = soup.find('body') # finds <body><p>Test</p></body>
print(body.contents) # [<p>Test</p>, <p>T</p>]
print(body.string) # None, because there is more than one thing
```

Listing 2: BeautifulSoup tag usage

4.1 Summary

Tables 1 and 2 show the BeautifulSoup methods and the tag attributes we will probably need in this lab. If you want to know more about BeautifulSoup, take a look at its documentation:

`http://www.crummy.com/software/BeautifulSoup/bs4/doc/`

Method	What it does
`s.find(name)`	Finds the first tag named `name` in document `s`
`s.findAll(name)`	Finds all tags named `name` in document `s`
`s.prettify()`	Returns a string of the HTML formatted prettily

Table 1: BeautifulSoup methods, where `s` is a BeautifulSoup object. See Listings 1 and 2 for example usage.

Attribute	What it gives
`.contents`	List of all the things a tag contains
`.name`	Name of the tag (e.g. `p` or `title`)
`.string`	If `.contents` only contains one thing, return a string of that. Otherwise, `None` since it will not know which one to choose

Table 2: Tag attributes. See Listing 2 for example usage.

5 Matplotlib

Matplotlib is a neat library for plotting in Python. It works similarly to MATLAB plotting so that people with experience in that can easily switch over; however, we will give you our own introduction to it.

Let's plot the following x and y coordinates:

x	0	2	4	6	8	10	12	14	16	18
y	0	4	8	12	16	20	24	28	32	36

Table 3: Values for Figure 1

```
1 import matplotlib.pylab as pl
2
3 x = range(0, 20, 2)
4 y = range(0, 40, 4)
5 pl.plot(x, y, '.')
6 pl.xlabel('X values')
7 pl.ylabel('Y values')
8 pl.title('Random plot of X vs Y')
9 pl.show()
```

Listing 3: Code to produce Figure 1 with values from Table 3

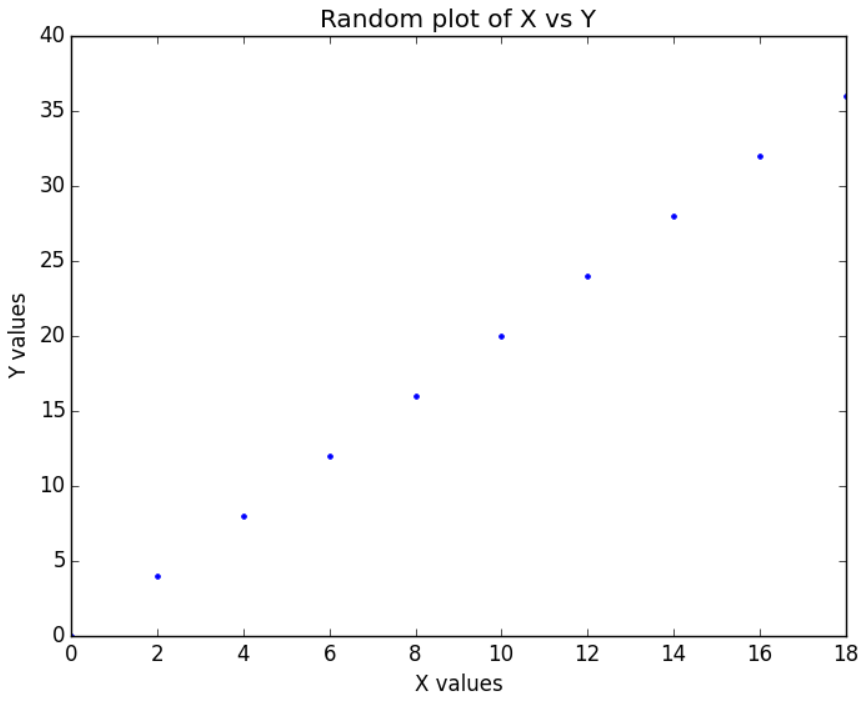

Figure 1: Graph produced by Listing 3

The `pl.plot()` function takes as arguments a sequence for x, a sequence for y, and optionally a formatting specifier. The important ones are "-" for a solid line and "." for point markers. You

can add more options, such as colors or labels. You can find more docs on these specifiers on the Matploblib website:

`http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot`

You have to call `pl.show()` for the graph to actually show. There are ways to print the graphs to image files, but we are not covering those for now. You can look them up in the documentation if you want.

Bar plots work in a similar way. The `pl.bar()` takes a sequence of numbers to label the left side of the bars with and a sequence of heights of the bars.

```
1 import matplotlib.pylab as pl
2
3 x = range(1, 11, 2)
4 y = range(1, 21, 4)
5 pl.bar(x, y)
6 pl.show()
```

Listing 4: Code to produce Figure 2

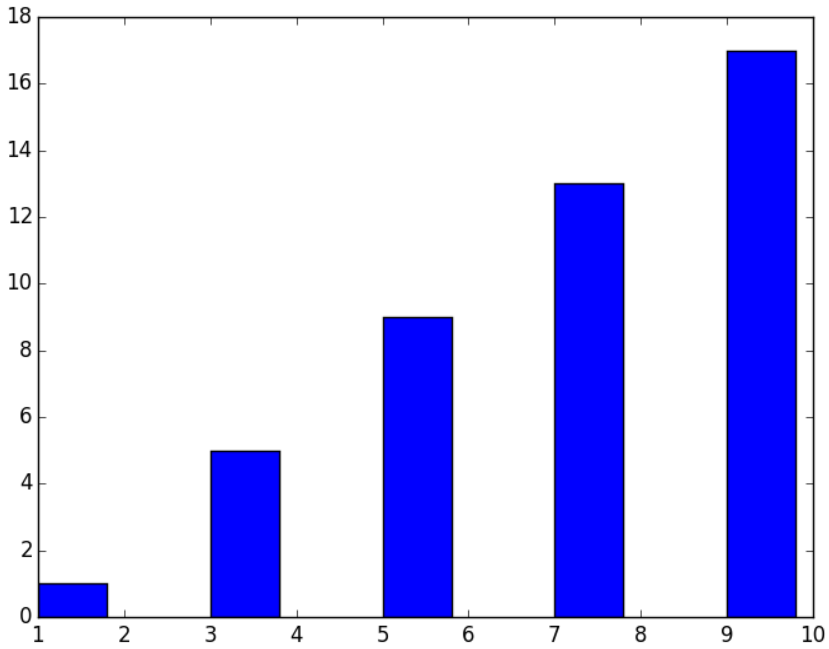

Figure 2: Graph produced by Listing 4

These are the two main functions you will need for plotting in this lab. If you want to customize more, please see the matplotlib pylab documentation at

`http://matplotlib.org/api/pyplot_summary.html`

6 Wikipedia

That one thing that russell found (about the philosophy link)

7 Exercises

Boilerplate

Remember that this lab *must* use the boilerplate syntax introduced in Lab 5.

file.py

8 Submitting

Files to submit:

- file.py (Section 7)

You may submit your code as either a tarball (instructions below) or as a .zip file. Either one should contain all files used in the exercises for this lab. The submitted file should be named either `cse107_firstname_lastname_lab11.zip` or `cse107_firstname_lastname_lab11.tar.gz` depending on which method you used.

For Windows, use a tool you like to create a `.zip` file. The TCC computers should have `7z` installed. For Linux, look at lab 1 for instructions on how to create a tarball or use the "Archive Manager" graphical tool.

Upload your tarball or .zip file to Canvas.