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. HTML stands for HyperText Markup Language. As a language, it defines where content appears on a website, and how it looks. For an example, open up `Google Chrome` and navigate to `http://www.nmt.edu`. Right click on the webpage, and select “View Source”. What follows is the source code for `nmt.edu`!

HTML is actually fairly easy to read. All content exists between various kinds of “tags”. Each tag has a special kind of meaning. Here is a simple example:

```html
<html>
  <body>
    <p> This is a paragraph in the body of the page </p>
    <p> This text is <b> BOLD </b></p>
  </body>
</html>
```

To learn about all the various tags, see W3Schools.

3 BeautifulSoup

BeautifulSoup is a python library used to read HTML files in a nice and easy way. In this lab, you will be using it along with urllib to interact with Wikipedia pages.

Below is an example of a simple program using urllib. This program gets a random Wikipedia article and prints out its URL. This is identical to what would happen if you click on the Random Article button on the Wikipedia homepage.

```python
from urllib import request

base = 'https://en.wikipedia.org'

def get_random():
  req = request.Request(base + '/wiki/Special:Random',
    headers={'User-Agent' : 'Python Browser'})
  page = request.urlopen(req)
  url = page.geturl()
  url = url[len(base):]
  return url

def main():
  rand_url = get_random()
  print('Found random page {}.'.format(rand_url))

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

All of the logic for this program is inside the `get_random` function. First, it creates a `Request` object. This is used to tell urllib what page we want to load. First, we construct our URL using the `base` variable (since all of our URLs are going to start the same way), then we pass a special headers argument. Don't worry too much about what headers are – in this case, we are only specifying a `User-Agent` string in order to identify ourselves to the remote server. We need to do this because the default urllib `User-Agent` string is blocked by Wikipedia.

Once we have our `Request` object created, we simply pass it to `request.urlopen` in order to get an object representing the page we've loaded. Creating the `Request` object and calling `request.urlopen` will be done no matter what page we are loading.

Once we have the page object, we simply check its URL (since the Wikipedia random article button redirects you to a new page, this will not be the same URL that we requested!) and strip off the beginning section, leaving us with only /wiki/Article.

4 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 1: Values for Figure 1

```python
import matplotlib.pylab as pl

x = range(0, 20, 2)
y = range(0, 40, 4)
pl.plot(x, y, ’.’)
pl.xlabel(’X values’)
pl.ylabel(’Y values’)
pl.title(’Random plot of X vs Y’)
pl.show()
```

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

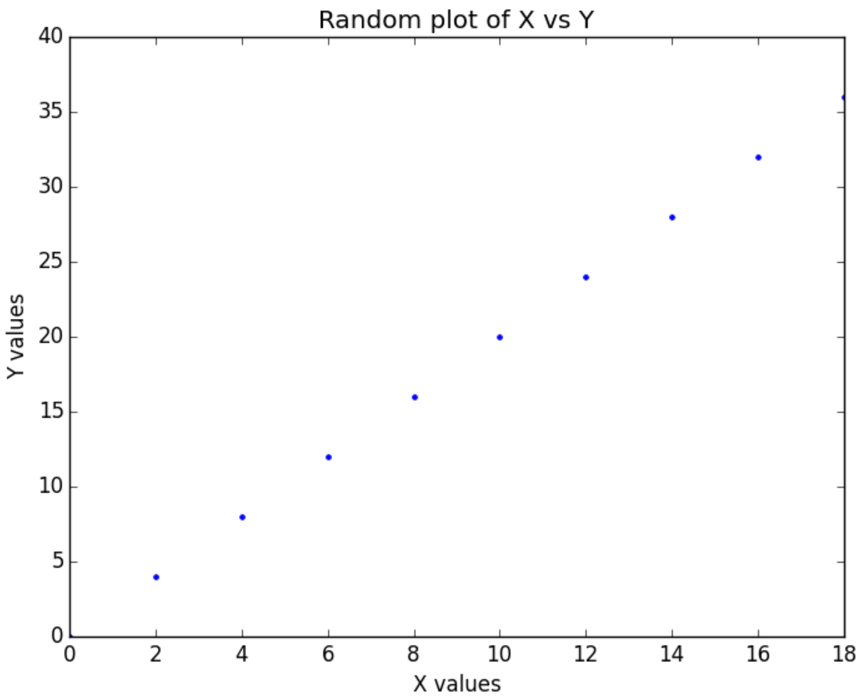

Figure 1: Graph produced by Listing 1

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.

```python
import matplotlib.pylab as pl

x = range(1, 11, 2)
y = range(1, 21, 4)
pl.bar(x, y)
pl.show()
```

Listing 2: Code to produce Figure 2

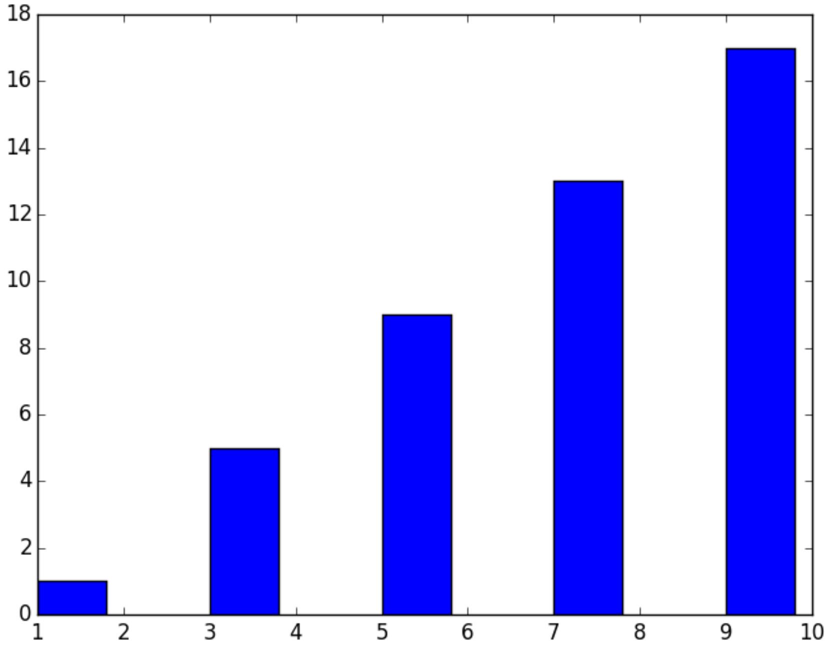

Figure 2: Graph produced by Listing 2

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
```

5 Wikipedia

In case you feel like wasting time, did you know that wikipedia has some odd properties about the philosophy page? You should read more at:
`http://en.wikipedia.org/wiki/Wikipedia:Getting_to_Philosophy`.

6 Exercises

Boilerplate

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

file.py

7 Submitting

Files to submit:

- file.py (Section 6)

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.