PYLADIESATX INTRO TO WEB SCRAPING

http://irmakramer.com/presentations/web_scraping.html

LET'S WEB SCRAPE!

WAIT, WHAT IS WEB SCRAPING?

Web scraping is the act of extracting data from websites via software.

Web scraping converts html data into useable structured data which can be pulled into files or databases.

WHY SHOULD I SCRAPE?

To Rule the World!

Or..

To have easily searchable and useable data for personal use, to make an application, or to make the data more accessible.

SERIOUS NOTES

Read the terms of service before you scrape. If a website has them and doesn't allow scrapers, please don't scrape them unless you've asked for permission.

Don't repeatedly scrape a site. Set a sleep timer for at least a second per scrape.

IS PYTHON THE WAY TO GO?

YES!

Python is a robust language with great documentation, a fantastic community and an amazing amount of libraries to help you do most anything!

There's even a library built just for web scraping...

BUT FIRST! WHAT'S A LIBRARY?

A library is code. Plain and simple, a library is the work of one or many people working to add functionality to Python. They release their packaged code in a way that is easily installable and useable.

No one wants to reinvent the wheel every time they write a piece of code. Libraries make it easy to do a myriad of things.

BEAUTIFUL SOUP

Beautiful Soup was built in 2004 to help scrape websites and has steadily been updated ever since.
It breaks down a website into a BeautifulSoup object that can be easily searched and extracted.

BeautifulSoup documentation

LET'S INSTALL IT!

If you use a virtual environment, start one up!

```
pip install beautifulsoup4
```

Let me know once you're all set!

HAVING TROUBLE?

Open a terminal/CMD window.

Navigate to a folder to hold your project.

```
mkdir my_folder
cd my_folder
```

Figure out what version you're on

```
python -V
```

```
pip install beautifulsoup4
```

Raise your hand if you still need help.

SCRAPING OUR LIBRARIES

Sorry, I couldn't help myself. We're going to scrape the Austin Public Libraries tonight. So let's go look at the site and what we'll be looking at.

http://library.austintexas.gov/

WHAT WE LOOKED AT:

How to inspect a webpage. Right click or CTRL+ click to get the inspect function. On Chrome you can also use the key-combo: F12 , Ctrl + Shift + I, Cmd + Opt + I
You'll be referring back to this often as you scrape, get to know it!

HTML Basics. Divs, Headers and tags.
You can find more documentation here: W3Schools

LET'S CODE!

Create a new python file, called `myfirstscrape.py`
Open it up in your text editor.

START TYPING!

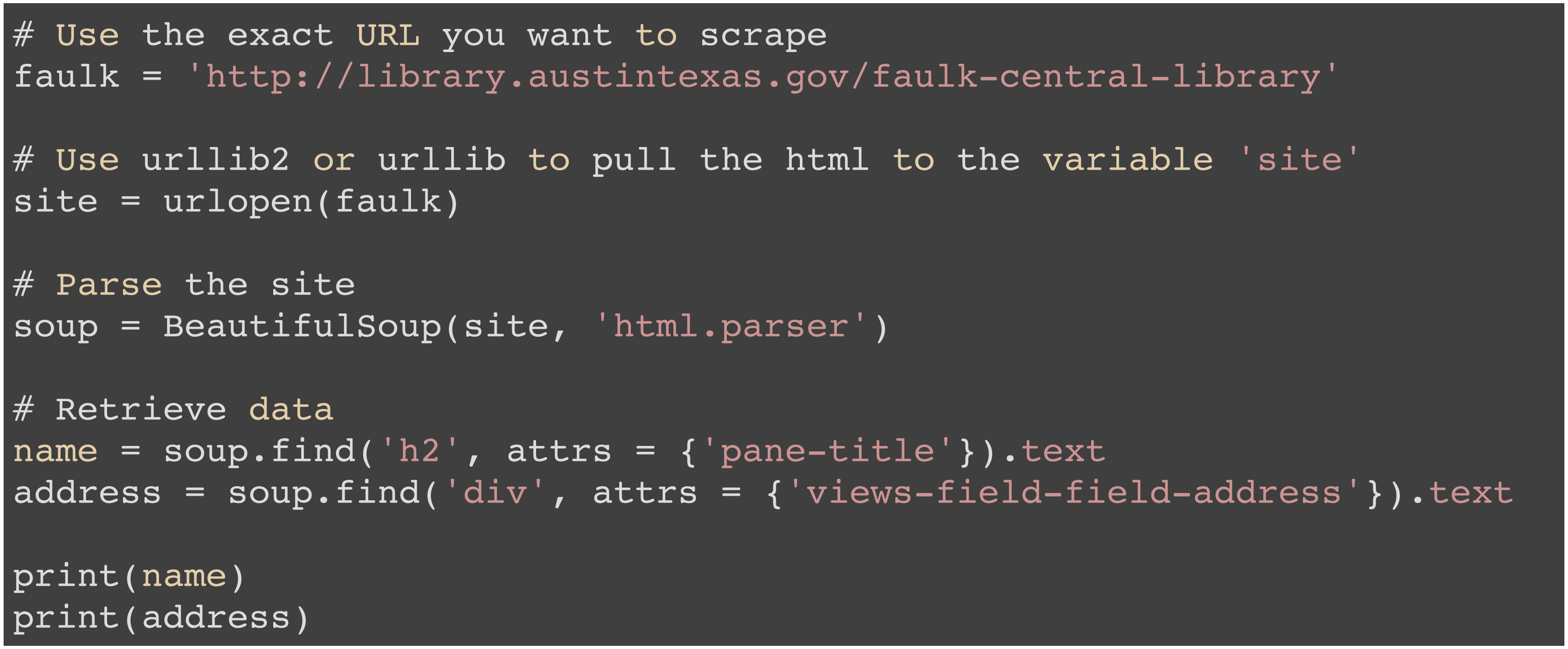
First we're going to import libraries.

```
# import libraries
import sys
from bs4 import BeautifulSoup
# Determine which urllib library to use
if sys.version_info[0] < 3:
    from urllib2 import urlopen
else:
    from urllib.request import urlopen
```

Save, run, debug if necessary. Let me know when you're done!

NOW LET'S GET TO THE GOOD STUFF

I'm going to give you all the code and walk you through it
Feel free to type while I talk.



```
# Use the exact URL you want to scrape
faulk = 'http://library.austintexas.gov/faulk-central-library'

# Use urllib2 or urllib to pull the html to the variable 'site'
site = urlopen(faulk)

# Parse the site
soup = BeautifulSoup(site, 'html.parser')

# Retrieve data
name = soup.find('h2', attrs = {'pane-title'}).text
address = soup.find('div', attrs = {'views-field-field-address'}).text

print(name)
print(address)
```

WHAT YOU SHOULD SEE

```
$ python myfirstscrape.py
Faulk Central Library
 800 Guadalupe St., Austin, TX, 78701
```

Yay! You scraped a site! Congratulations!

CHALLENGE

Can you scrape the phone number? I'll give you 5 minutes to try it and if anyone is catching up, I'll walk around and help.

```
phone = soup.find('div', attrs = {'views-field-field-phone-num'}).text
print(phone)
```

NOW LET'S TRY THAT LIST

Let's make a new file before we start.
Copy over the code from our first scrape.

What are the pieces of code we'll be changing?

```
locations = 'http://library.austintexas.gov/locations'
site = urlopen(locations)
# Retrieve data
???
```

BEAUTIFULSOUP OBJECTS

The soup we create has all the elements in the HTML of the website, but they are easily found with functions on the soup object.

We'll be focusing on the find_all function which is similar to find. find_all returns an iterable list of objects that all contain the search term.

LET'S LOOK AT THE LIST

What is common with each location?
What tag can we look for to give us all the locations?

```
div class = "apl-box"
```

```
all_locations = soup.find_all('div', attrs = {"apl-box"})
```

WORKING ON YOUR SCRAPER

10 minutes of work time.

THINGS YOU MIGHT WANT TO TRY

1. Inspecting the list
2. Writing a for loop
3. Printing everything!

SOLUTIONS

I've put up the code we worked on and how you can iterate over the list in a github repo. Feel free to look through it.
https://github.com/bluflowr/webscraping_workshop

HOMEWORK

- Now that you can pull down the data, why not save it?
- Try saving it to a CSV or better yet, try saving it to a database!
- Clean up your code.
- Scrape something else and show us!

RESOURCES

- BeautifulSoup Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- GitHub Repo: https://github.com/bluflowr/webscraping_workshop
- Slide Deck: http://irmakramer.com/presentations/web_scraping.html

THANK YOU!

Irma Kramer
@bluflowr
irma@irmakramer.com

pyladies
Austin