ITP 115
Programming in Python

Web Scraping

What is Web Scraping?

- Web scraping is a technique used to collect data and other information from websites for further use
- Python is great for writing web scraping scripts to parse websites

How is Web Scraping Used?

- Retailers scraping other websites for price comparisons (flights, hotels, etc.)
- Scraping product reviews to detect fraudulent reviews
- Help companies and businesses understand user behaviors and reactions (for product performance, new audience, etc.)
- Job search websites
- And many other ways

HTML

- **H**yper **T**ext **M**arkup **L**anguage
- HTML is a language used for creating web pages and applications
- Web browsers read "tags" to translate text into a visual format (user interface)

Other Web Scraping Formats

- XML: e**X**tensible **M**arkup **L**anguage
  - Markup language, similar to HTML
  - Designed to carry data (as opposed to displaying data like HTML is)
- JSON: **J**ava**S**cript **O**bject **N**otation
  - Minimal, easily readable data format
  - Primarily used for data transfer between servers and web applications

HTML Tags

- Tags separate pieces of content
- Tags **do not appear** on the final web page, but the **effects** of the tags do
- Tags are usually written in pairs and can be nested inside of each other

```
<tagname>content goes here…</tagname>
```

Start tag

End tag

HTML Tags

- Each tag has a name and may also have attributes which provide additional information

```
<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">
```

Tag name **Attribute name** **Attribute value**

Viewing HTML

- To view the HTML code behind a webpage, right click → Inspect
- Browse through the HTML tree and expand tags to find patterns and where pieces of data are stored

Viewing HTML Example

http://www.usc.edu/


```
…<!DOCTYPE html> == $0
  <html lang="en" id="www-usc-edu">
  ▶#shadow-root (open)
  ▶<head>…</head>
  ▼<body cz-shortcut-listen="true">
    ▼<nav class="accessibility" id="accessibility">
      ▼<ul>
        ▼<li>
            <a href="#primary">Skip to navigation</a>
          </li>
        ▼<li>
            <a href="#features">Skip to features</a>
          </li>
        ▼<li>
            <a href="#modules">Skip to news modules</a>
          </li>
        ▼<li>
            <a href="#current">Skip to current conditions</a>
          </li>
      </ul>
    </nav>
```

HTML Tags: id Attribute

- The **`id`** attribute is a **unique** id for an HTML element (i.e. only one per document).
- Primarily used to identify elements in CSS and JavaScript for further manipulation

```
<h2><a id="top">Some heading</a></h2>
```

HTML Tags: Class Attribute

- **`class`** is a special HTML attribute used to group multiple tags to make styling elements simpler
- The attribute value can be set by the programmer
- Makes scraping for similar or related pieces of data easier

```
<div class="cities">
<h2>London</h2>
<p>London is the capital of England. It is the most populous city in
the United Kingdom, with a metropolitan area of over 13 million
inhabitants.</p>
</div>

<div class="cities">
<h2>Paris</h2>
<p>Paris is the capital and most populous city of France.</p>
</div>
```

HTML Tags: Title and Body

- The **`title`** tag is required for all HTML documents and sets the title of the webpage
- The **body** tag contains all of the page's visible content

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

<body>
The content of the document......
</body>

</html>
```

The content of the document.....

HTML Tags: Headings

- HTML has 6 different heading tags: **h1**, **h2**, **h3**, **h4**, **h5**, and **h6**
- These heading tags format the text between the start and end tag

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
```

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

HTML Tags: Div

- The **`div`** tag defines a section or division in the HTML document

```
<p>This is some text.</p>

<div style="color:#0000FF;
    border:thick dotted red" >
  <h3>Heading in a div </h3>
  <p>Text in a div</p>
</div>

<p>This is some text.</p>
```

This is some text.

Heading in a div

Text in a div

This is some text.

HTML Tags: a

- The **a** tag defines a hyperlink
- Use the **`href`** attribute to set the url
- Link is displayed as the text is between the tags

```
<!DOCTYPE html>
<html>
<body>

<a href="https://www.usc.edu">
    Visit USC Online!
</a>

</body>
</html>
```

Visit USC Online!

HTML Tags: Lists

- Lists can be unordered (bulleted) or ordered (numbered)
- Unordered lists are started with the **`ul`** tag and ordered lists are started with the **`ol`** tag
  - Both use the **`li`** tag to list the actual items

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

- Coffee
- Tea
- Milk

HTML Tags: Tables

- Defined with the **`table`** tag
- Sometimes contains a **`tbody`** tag to group table body content
- Nested inside are **`tr`** tags for each table row
- Nested inside the **`tr`** tags are:
  - **`th`** tags for table headers (one for each column)
  - **`td`** tags for table data/cell (one for each column)

HTML Tags: Tables Cont.

```
<table>

</table>
```

HTML Tags: Tables Cont.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>

</table>
```

Firstname	Lastname	Age

HTML Tags: Tables Cont.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>



  </tr>
```

```
</table>
```

Firstname	Lastname	Age
Jill		

HTML Tags: Tables Cont.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
</table>
```

Firstname	Lastname	Age
Jill	Smith	50

HTML Tags: Tables Cont.

```html
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

HTML Tags: Tables Cont.

```html
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
  <tr>
    <td>John</td>
    <td>Doe</td>
    <td>80</td>
  </tr>
</table>
```

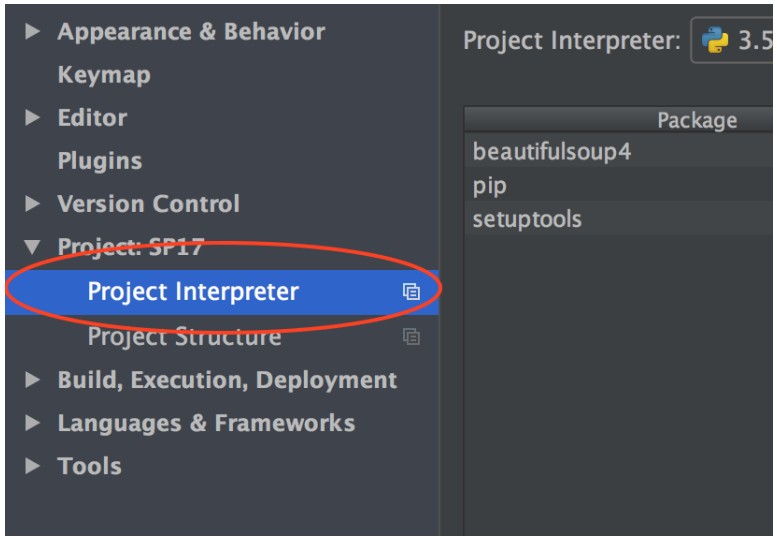
Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

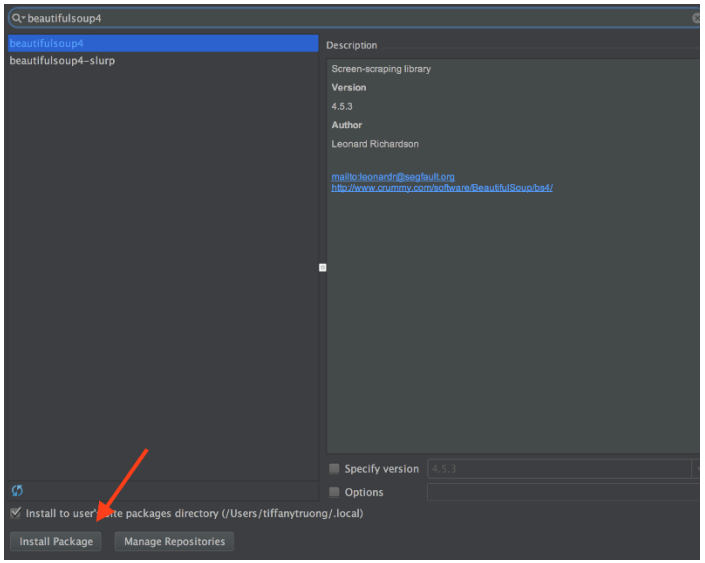
What is Beautiful Soup?

- Beautiful Soup is a Python module made for parsing HTML and XML files
- It is a powerful library which allows programmers to easily modify, search, and navigate through tags.

Installing BeautifulSoup Module

- Steps to install a module in PyCharm:
- File → Settings (PC) or PyCharm → Preferences (Mac)
- Project Interpreter → (Plus Sign)
- Search "beautifulsoup4" → Install Package





Using BeautifulSoup

- Beautiful Soup works with both HTML and XML files. There are 3 ways to read in data (or "make soup"):
  1. Opening an HTML or XML file
  2. Passing in a string
  3. Requesting a web page via its URL (what we'll do in class)

Example

- Go to http://goo.gl/vy9dzM (a modified version of the website Hacker News)
- Inspect the page to explore its HTML

Making Soup

- To parse a webpage from its URL:

```
from bs4 import BeautifulSoup
import urllib.request
```

Import necessary modules

Making Soup

- To parse a webpage from its URL:

```
from bs4 import BeautifulSoup
import urllib.request

url = "https://www.google.com/"
page = urllib.request.urlopen(url)
```

URL of website

Request webpage

Making Soup

- To parse a webpage from its URL:

```
from bs4 import BeautifulSoup
import urllib.request

url = "https://www.google.com/"
page = urllib.request.urlopen(url)

soup = BeautifulSoup(page.read(), "html.parser")
```

Get HTML tag "soup" from page

Viewing the HTML

- To view all of a page's HTML code we can print the "soup"
- Optional: view the output in a readable format by using the **`prettify()`** method

```
print(soup)
```

OR

```
print(soup.prettify())
```

Navigating The Tags

- We can navigate through all of the tags using a for loop
- This goes through every outermost single tag in the document from top to bottom

```
for tag in soup:
    # "tag" is a tag from the HTML doc
```

Getting Tag Information

- With a tag, you can access its name:

```
tag.name
```

- A dictionary of its attributes

```
tag.attrs
```

- An attribute's value (accessed same as dictionary key-value pairs) :

```
tag[attrName] # attrName is a string
```

Accessing Tags

- Specific tags can be accessed using the tag name
- This will give you the first tag in the HTML with that name:

```
# Examples
soup.title # Gets the title tag
soup.a # Gets the first a tag
```

Finding Multiple Tags

- Find a single tag using:

```
soup.find(tagName)
```

- Find multiple tags using:

```
soup.find_all(tagName)
```

  – Returns a list of tags

```
# Examples
hSixTag = soup.find("h6")
hSixTagList = soup.find_all("h6")
```

Finding Multiple Tags

- You can use these functions to also search for tags inside of tags:

```
bodyTag = soup.find("body")
bodyLinks = bodyTag.find_all("a")
# bodyLinks now has a list of only
the 'a' tags within the body tag
```

Finding Multiple Tags Continued

- Similar to the **`print`** function these methods use additional optional parameters

Finding Multiple Tags Continued

- You can use any combination of 1+ of these parameters:
  - Tag name (as a string)
  - Multiple tag names (use a list of strings)
  - Contains a certain attribute (use `attrName=True`)
  - Contains an attribute with a specific value (use `attrName=val`)
  - The text between the tags(use `.text`)
  - Class Name (use `class_="…"`)

Finding Multiple Tags Continued

- This allows us to refine our search for certain tags even more:

```
# Example – find only h6 tags with the class
# "heading"
hTitleTags = soup.find_all("h6", class_="heading")
```

Exercise

Links and Resources

- BeautifulSoup Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Read more about HTML: http://www.w3schools.com/html/default.asp
- List of HTML tags http://www.w3schools.com/TAgs/
- Basic examples of common tags http://www.w3schools.com/html/html_basic.asp