Web Scraping

Renato R. Maaliw III, *DIT*
College of Engineering
Southern Luzon State University
Lucban, Quezon, Philippines

Web Scraping

- Refers to the automated collection of data from websites.
- Instead of manually copying information from web pages, scripts or codes gather and process the data programatically

Uses Cases

- Data aggregation and analysis
- Price monitoring and competitive analysis
- Research

Request Headers

- User-Agent can be customized to mimic a browser request in avoiding blocking and ensuring the server returns the expected content.

HTML and DOM Structure

- Tags, attributes, text
- Class, IDs, and other attributes

Parsing Techniques

- BeautifulSoup
- lxml
- Xpath

Handling Dynamic Content

- Selenium (Not covered in this course)

Rate Limiting & Politeness

- Respect the site’s rules for what is allowed and disallowed paths for crawlers

Ethical and Legal Considerations

- Ensure your scraping activity does not violate the website's Terms of Service (ToS). Some sites explicitly forbid scraping.
- Be cautious when handling personal data. Follow data protection & privacy laws when collection and storing scraped data.

Data Cleaning

- Raw data often requires cleaning
- Converting data types
- Handling missing values

Data Storage

- CSV, JSON, XML, etc.

Post-Processing

- Libraries such as Pandas in Python are helpful transforming & analyzing scraped data.

BeautifulSoup Library

Process

1. Install or import library
2. Send an HTTP request
3. Parse the HTML Content
4. Locate and Extract Data
5. Clean and Process the Data
6. Store or Output the Extracted Data

Data Extraction Methods

find() – locate the first occurrence of a tag that matches the criteria

Example:

```
soup.find('h1')
soup.find('p', class_ = 'description')
soup.find('div', id='main')
soup.find('th', attrs = {'class': 'country'})
```

Data Extraction Methods

.find_all() – locate all instances of a tag that matches the criteria and return them as a list

Example:

soup.**find_all**('p') – find all <p> tags regardless of their attributes

Data Extraction Methods

.find_all() – locate all instances of a tag that matches the criteria and return them as a list

Example:
soup.**find_all**('p', attrs = {'class': 'intro}) – find all <p> tags with specific attribute of class='intro'

Data Extraction Methods

.find_all() – locate all instances of a tag that matches the criteria and return them as a list

Example:

```
div_main = soup.find('div', id = 'main')
all_p_recursive = div_main.find_all('p', recursive = True)
```

(default) searches all descendants

Data Extraction Methods

.find_all() – locate all instances of a tag that matches the criteria and return them as a list

Example:

```
div_main = soup.find('div', id = 'main')
all_p_recursive = div_main.find_all('p', recursive = False)
```

limit the search to immediate children

Data Extraction Methods

.find_all() – locate all instances of a tag that matches the criteria and return them as a list

Example:

soup.**find_all**('p', string = 'Hello')

<p> tags whose text exactly matches 'Hello'

Data Extraction Methods

.find_all() – locate all instances of a tag that matches the criteria and return them as a list

Example:

```
soup.find_all('p', string = re.compile('Hello'))
```

<p> tags whose text contains 'Hello'

Data Extraction Methods

.find_all() – locate all instances of a tag that matches the criteria and return them as a list

Example:

soup.**find_all**('p', limit = 2)

<p> tags returned to the first two matches

Data Extraction Methods

.select() – extract elements using CSS selectors

Example:

soup.**select**('div#main p.description')

soup.**select**('div > ol.list')

Data Extraction Methods

.select_one() – extract the first element that matches a given CSS selector

Example:

soup.**select_one**('div.panels p')
soup.**select_one**('div.header > ol.list')

Data Extraction Methods

.children() – iterator for immediate children of a tag

Example:

```
div_main = soup.find('div', id = 'main')

for child in div_main.children:

    // code here
```

Data Extraction Methods

.descendants() – iterator for all descendants (children, grandchildren, etc.) of a tag

Example:

```
div_main = soup.find('div', id = 'main')

for child in div_main.descendants:

    // code here
```

Data Extraction Methods

.find_next_sibling() – navigate to adjacent elements at the same hierarchical level

Example:

```
first_p = soup.find('p')
next_p = first_p.find_next_sibling('p')
```

Data Extraction Methods

.find_previous_sibling() – navigate to adjacent elements at the same hierarchical level

Example:

```
first_p = soup.find('p')
previous_p = first_p.find_previous_sibling('p')
```

Data Extraction Methods

.get_text() – retrieve all the text within a tag, stripping away the HTML tags

Example:

```
p_tag = soup.find('p')
p_text = p_tag.get_text()
```

```
.get_text('separator = " ", strip = True)
```

extract text, joining content with a space & removing extra whitespace

Data Extraction Methods

.attrs – returns a dictionary of all attributes for a given tag.

Example:

```
div_tag = soup.find('div', id = 'main')
div_tag.attrs
```

Data Extraction Methods

.get – retrieves the value of a specific attribute from a tag

Example:

```
div_id = div_tag.get('id')
div_class = div_tag.get('class')
div_data_info = div_tag.get('data-info')
```

[Web Scraping End-to-End Process]

Thank you very much for listening.