UNIVERSITI TENAGA NASIONAL

The Energy University

CISB5123 Text Analytics
Lab 3
Web Scraping

Web Scraping is used to ***extract data from website***.

The steps in Web Scraping

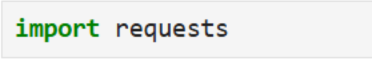
1. Inspect the structure the web site
2. Crawling – navigate the target website and download the response from the website
3. Parse the downloaded data into HTML parser and extract the required data
4. Store the extracted data for further use

Python Libraries used in Web Scraping

1. **requests** – Python library used for making various types of HTTP requests like GET, POST. It used to raw HTML codes from the website that you want to extract the data from
2. **BeautifulSoup** – Python library used for parsing HTML document (library that is used to extract data from HTML documents)

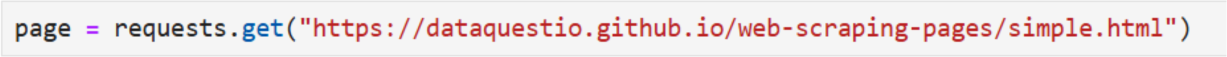
Scraping from A Simple Web Page

1. Let's try downloading a simple sample website, https://dataquestio.github.io/web-scraping-pages/simple.html
2. To view the HTML code: Right-click and select either "Inspect" or "View page source". You can use any HTM formatter (e.g. https://webformatter.com/html) to make it readable.
3. The first thing we'll need to do to scrape a web page is to download the page. We can download pages using the Python requests library.



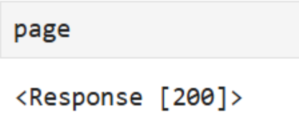
```
import requests
```

4. The requests library will make a GET request to a web server, which will download the HTML contents of a given web page for us.



```
page = requests.get("https://dataquestio.github.io/web-scraping-pages/simple.html")
```

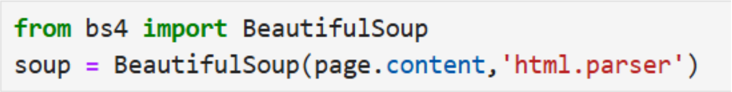
5. After running our request, we get a Response object. This object has a status_code property, which indicates if the page was downloaded successfully:



```
page
```

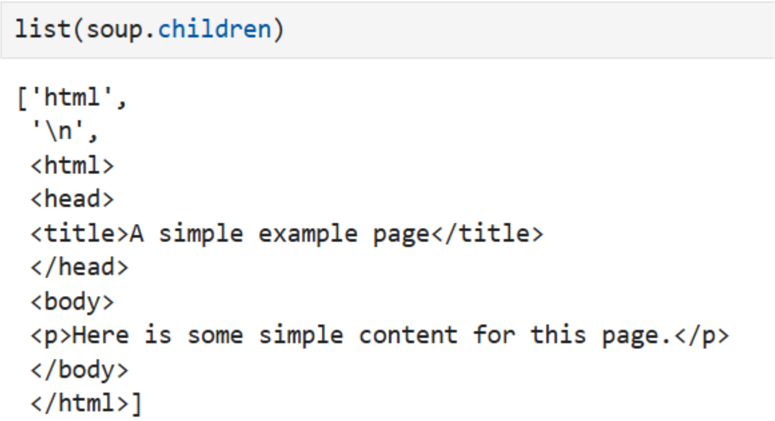
```
<Response [200]>
```

6. We can use the BeautifulSoup library to parse this document and extract the text from the p tag. We first have to import the library, and create an instance of the BeautifulSoup class to parse our document:



```
from bs4 import BeautifulSoup
soup = BeautifulSoup(page.content,'html.parser')
```

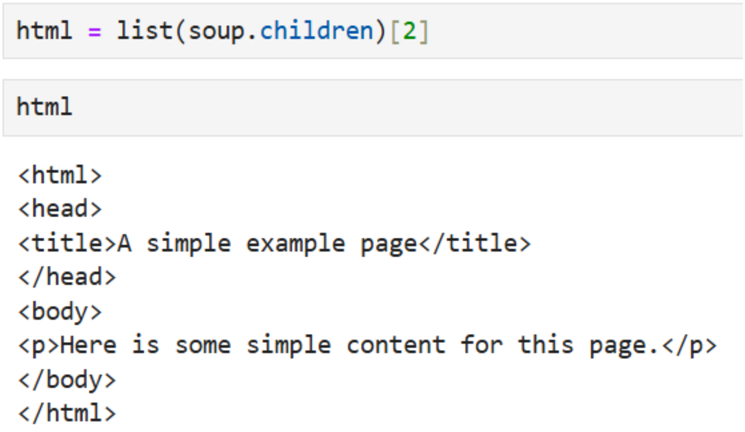
7. As all the tags are nested, we can move through the structure one level at a time. We can first select all the elements at the top level of the page using the children property of soup. Note that children returns a list generator, so we need to call the list function on it:



```
list(soup.children)
```

```
['html',
 '\n',
 <html>
 <head>
 <title>A simple example page</title>
 </head>
 <body>
 <p>Here is some simple content for this page.</p>
 </body>
 </html>]
```

8. The most important object type, and the one we'll deal with most often, is the Tag object. The Tag object allows us to navigate through an HTML document and extract other tags and text. The Tag object allows us to navigate through an HTML document and extract other tags and text. You can learn more about the various BeautifulSoup objects here.

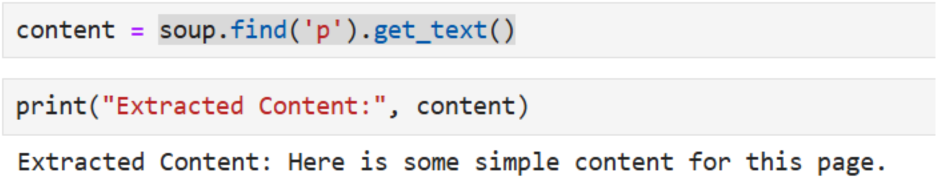


```
html = list(soup.children)[2]
```

```
html
```

```
<html>
<head>
<title>A simple example page</title>
</head>
<body>
<p>Here is some simple content for this page.</p>
</body>
</html>
```

9. Extract and print the desired data:

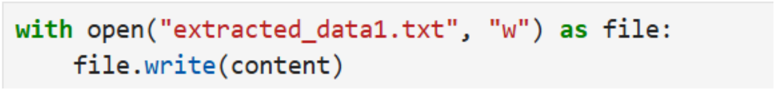


```
content = soup.find('p').get_text()
```

```
print("Extracted Content:", content)
```

```
Extracted Content: Here is some simple content for this page.
```

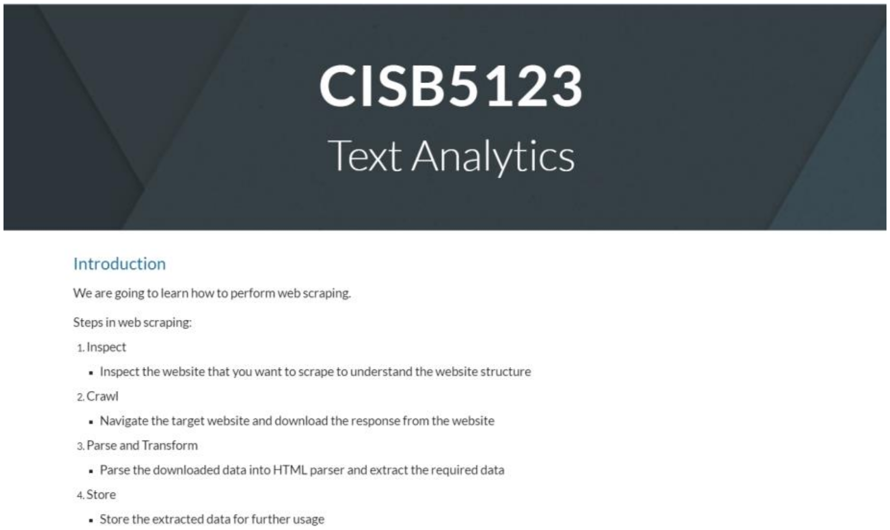
10. Store extracted data:



```
with open("extracted_data1.txt", "w") as file:
    file.write(content)
```

Scraping from A Web Page

1. We want to scrape this site: https://sites.google.com/view/cisb5123/home



CISB5123

Text Analytics

Introduction

We are going to learn how to perform web scraping.

Steps in web scraping:

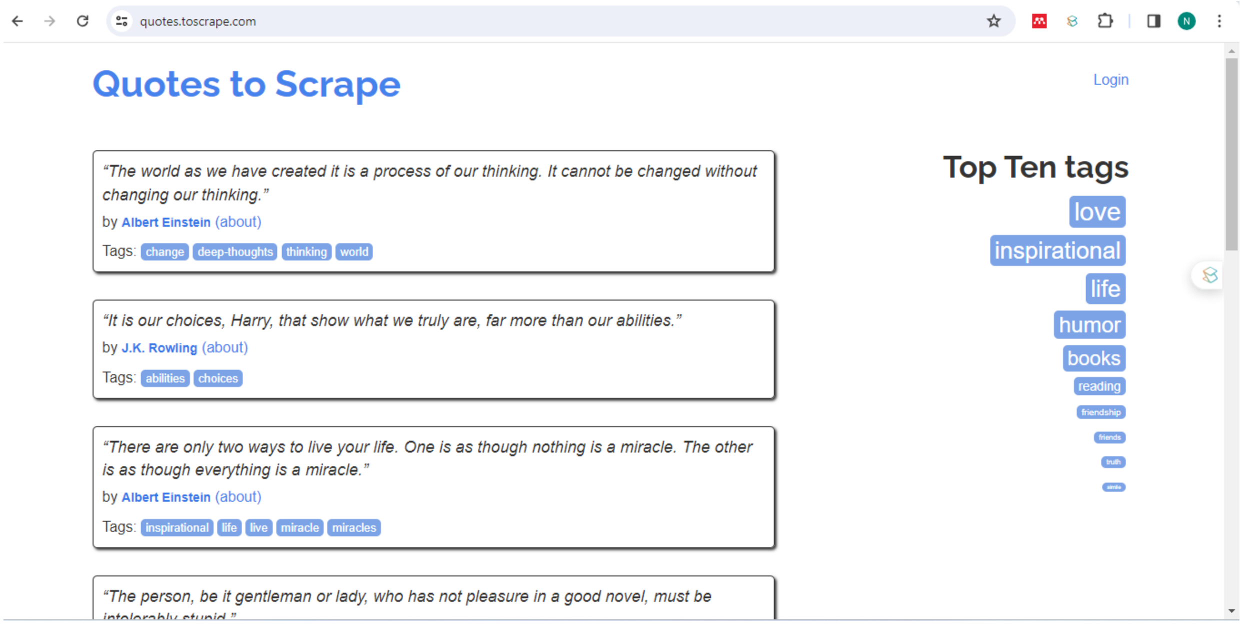
1. Inspect
    - Inspect the website that you want to scrape to understand the website structure
2. Crawl
    - Navigate the target website and download the response from the website
3. Parse and Transform
    - Parse the downloaded data into HTML parser and extract the required data
4. Store
    - Store the extracted data for further usage

2. Code to scrape the content:

```
import requests
from bs4 import BeautifulSoup
import csv

# Send a GET request to the website
page = requests.get('https://quotes.toscrape.com')

# Parse the HTML content
soup = BeautifulSoup(page.text, 'html.parser')

# Create a list to store quotes
quotes = []

# Find all quote elements
quote_elements = soup.find_all('div', class_='quote')

# Extract information from each quote element
for quote_element in quote_elements:
    # extract the text of the quote
    text = quote_element.find('span', class_='text').text
    # extract the author of the quote
    author = quote_element.find('small', class_='author').text
    # extract the tag <a> HTML elements related to the quote
    tag_elements = quote_element.select('.tags .tag')

    # store the list of tag strings in a list
    tags = []
    for tag_element in tag_elements:
        tags.append(tag_element.text)

    quotes.append(
    {
        'text': text,
        'author': author,
        'tags': ', '.join(tags) # merge the tags into a "A, B, ..., Z" string
    }
)

# Print the scraped quotes - optional
for quote in quotes:
    print("Quote: ", quote['text'])
    print("Author: ", quote['author'])
    print("Tags: ", quote['tags'])
    print()

# Save quotes to a CSV file
with open('quotes.csv', 'w', encoding='utf-8', newline='') as csvfile:
    fieldnames = ['text', 'author', 'tags']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    # Write headers
    writer.writeheader()

    # Write quotes
    for quote in quotes:
        writer.writerow(quote)

print("Quotes have been saved to quotes.csv")
```

Print Specific Content of A Single Page

1. The webpage to be scraped:



Quotes to Scrape

Login

“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”
by Albert Einstein (about)
Tags: change deep-thoughts thinking world

“It is our choices, Harry, that show what we truly are, far more than our abilities.”
by J.K. Rowling (about)
Tags: abilities choices

“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.”
by Albert Einstein (about)
Tags: inspirational life live miracle miracles

“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.”

Top Ten tags

love
inspirational
life
humor
books
reading
friendship
friends
truth
simile

URL of the webpage

https://quotes.toscrape.com/

2. Code to scrape specific content from a single page:

```
import requests
from bs4 import BeautifulSoup
import csv

# Send a GET request to the website
page = requests.get('https://quotes.toscrape.com')

# Parse the HTML content
soup = BeautifulSoup(page.text, 'html.parser')

# Create a list to store quotes
quotes = []

# Find all quote elements
quote_elements = soup.find_all('div', class_='quote')

# Extract information from each quote element
for quote_element in quote_elements:
    # extract the text of the quote
    text = quote_element.find('span', class_='text').text
    # extract the author of the quote
    author = quote_element.find('small', class_='author').text

    # extract the tag <a> HTML elements related to the quote
    tag_elements = quote_element.select('.tags .tag')

    # store the list of tag strings in a list
    tags = []
    for tag_element in tag_elements:
        tags.append(tag_element.text)

    quotes.append(
    {
        'text': text,
        'author': author,
        'tags': ', '.join(tags) # merge the tags into a "A, B, ..., Z" string
    }
)

# Print the scraped quotes - optional
for quote in quotes:
    print("Quote: ", quote['text'])
    print("Author: ", quote['author'])
    print("Tags: ", quote['tags'])
    print()

# Save quotes to a CSV file
with open('quotes.csv', 'w', encoding='utf-8', newline='') as csvfile:
    fieldnames = ['text', 'author', 'tags']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    # Write headers
    writer.writeheader()

    # Write quotes
    for quote in quotes:
        writer.writerow(quote)

print("Quotes have been saved to quotes.csv")
```

Code explanation:

import requests

- Send HTTP request and retrieve data from web pages

from bs4 import BeautifulSoup

- Library to parse HTML and XML documents

import csv

- To read and write CSV files

page = requests.get('https://quotes.toscrape.com')

- Sends an HTTP GET request to the specified URL and retrieves the page content.

soup = BeautifulSoup(page.text, 'html.parser')

- Creates a BeautifulSoup object called soup from the HTML content of the response.

quotes = []

- Creates an empty list quotes to store the extracted quotes.

quote_elements = soup.find_all('div', class_='quote')

- Finds all <div> elements with the class 'quote', which contain individual quotes.

for quote_element in quote_elements:

- Iterate Over Each Quote and Extract Information
    - Iterates over all <div class='quote'> elements found in the soup object.
    - Each quote_element represents a single quote.
    - Finds the <span> element with the class 'text' within quote_element and extracts the text content.
    - Finds the <small> element with the class 'author' within quote_element and extracts the author's name.
    - Finds all <a> elements with the class 'tag' within quote_element.
    - Extracts the text content of each <a> tag element and joins them into a single string separated by commas.
    - Appends a dictionary containing the extracted quote text, author, and tags to the quotes list.

for quote in quotes:

- Print the Extracted Quotes
    - Iterates through the quotes list and prints each quote along with the author and tags.

with open('quotes.csv', 'w', encoding='utf-8', newline='') as csvfile:

- Save the Extracted Quotes to a CSV File
    - Opens a CSV file named 'quotes.csv' in write mode with UTF-8 encoding and creates a file object csvfile.
    - Defines the column headers for the CSV file.
    - Creates a DictWriter object called writer to write dictionaries to the CSV file with the specified column headers.
    - Writes the header row (text, author, tags) to the CSV file.
    - Writes each quote stored in the quotes list to the CSV file.
    - Prints a success message indicating that the quotes have been saved.

Print Specific Content of Multiple Pages

Code to scrape specific content from multiple pages:

```
import requests
from bs4 import BeautifulSoup
import csv

# Function to scrape quotes from a page
def scrape_page(soup, quotes):
    for quote in soup.find_all('div', class_='quote'):
        text = quote.find('span', class_='text').text
        author = quote.find('small', class_='author').text
        tags = ', '.join(tag.text for tag in quote.find_all('a', class_='tag'))
        quotes.append({'Text': text, 'Author': author, 'Tags': tags})

# Base URL and headers
base_url = 'https://quotes.toscrape.com'
headers = {'User-Agent': 'Mozilla/5.0'}

# List to store quotes
quotes = []

# Function to scrape quotes from multiple pages
def scrape_all_pages(url):
    while url:
        response = requests.get(url, headers=headers)
        soup = BeautifulSoup(response.text, 'html.parser')
        scrape_page(soup, quotes)
        next_page = soup.find('li', class_='next')
        url = base_url + next_page.find('a')['href'] if next_page else None

# Scrape quotes from all pages
scrape_all_pages(base_url)

# Save quotes to CSV file
with open('quotes2.csv', 'w', newline='', encoding='utf-8') as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames=['Text', 'Author', 'Tags'])
    writer.writeheader()
    writer.writerows(quotes)
```

Code explanation:

import requests

- Send HTTP request and retrieve data from web pages

from bs4 import BeautifulSoup

- Library to parse HTML and XML documents

import csv

- To read and write CSV files

def scrape_page(soup, quotes):

- A function called scrape_page that takes two arguments:
    - soup: a BeautifulSoup object representing the HTML content of a web page
    - quotes: a list to store the scraped quotes.

for quote in soup.find_all('div', class_='quote'):

- Iterates over all <div> elements with the class 'quote' found in the soup object.
- Each quote object represents a quote found on the web page.

text = quote.find('span', class_='text').text

- Finds the <span> element with the class 'text' within the quote object and extracts the text content of the element.

author = quote.find('small', class_='author').text

- Finds the <small> element with the class 'author' within the quote object and extracts the text content of the element.

tags = ', '.join(tag.text for tag in quote.find_all('a', class_='tag'))

- Finds all <a> elements with the class 'tag' within the quote object, extracts the text content of each element, and joins them into a single string separated by commas.

quotes.append({'Text': text, 'Author': author, 'Tags': tags})

- Appends a dictionary containing the extracted text, author, and tags to the quotes list.

base_url = 'https://quotes.toscrape.com'

- Defines the base URL of the website from which quotes will be scraped.

quotes = []

- Initializes an empty list called quotes to store the scraped quotes.

def scrape_all_pages(url):

- A function called scrape_all_pages that takes a single argument url, representing the URL of a web page from which quotes will be scraped.

while url:

- A while loop that continues as long as the url variable is not None.

response = requests.get(url, headers=headers)

- Sends a GET request to the specified url using the requests.get() function.
- The headers argument is optional and can be used to specify custom headers for the request.

soup = BeautifulSoup(response.text, 'html.parser')

- Creates a BeautifulSoup object called soup from the HTML content of the response.

scrape_page(soup, quotes)

- Calls the scrape_page function to scrape quotes from the current page and adds them to the quotes list.

next_page = soup.find('li', class_='next')

- Finds the <li> element with the class 'next', which contains the link to the next page of quotes.

url = base_url + next_page.find('a')['href'] if next_page else None

- Updates the url variable to the URL of the next page of quotes if it exists, or sets it to None otherwise.

scrape_all_pages(base_url)

- Calls the scrape_all_pages function with the base URL to start scraping quotes from the first page.

```
with open('quotes2.csv', 'w', newline='', encoding='utf-8') as csvfile:
```

- Opens a CSV file named 'quotes2.csv' in write mode with UTF-8 encoding and creates a file object called csvfile.

```
writer = csv.DictWriter(csvfile, fieldnames=['Text', 'Author', 'Tags'])
```

- Creates a DictWriter object called writer to write dictionaries to the CSV file with fieldnames specified as ['Text', 'Author', 'Tags'].

```
writer.writeheader()
```

- Writes the header row to the CSV file with the fieldnames specified above.

```
writer.writerows(quotes)
```

- Writes all the quotes stored in the quotes list to the CSV file.