CISB5123 Text Analytics
Lab 3
Web Scraping

***Note: Before proceeding with this lab, ensure that you have tried the web scraping steps for a single page from the lecture slide for Topic 2 – Section 3*

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

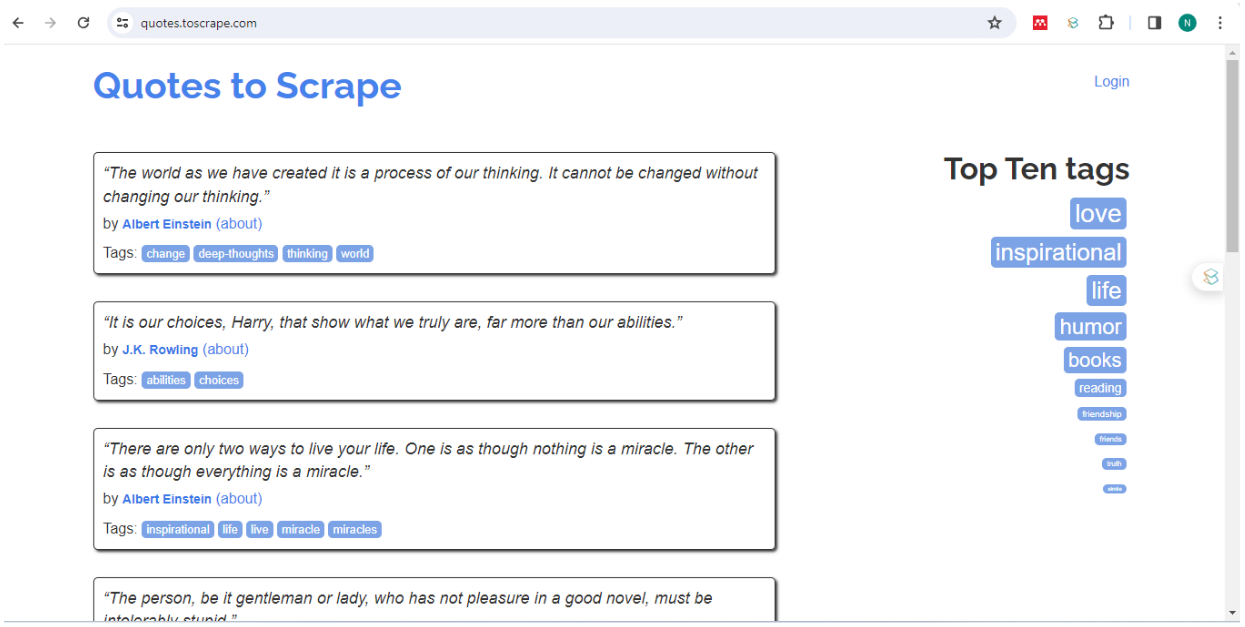
The steps in Web Scraping

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)

The webpage to be scraped


quotes.toscrape.com

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

Top Ten tags
love
inspirational
life
humor
books
reading
friendship
friends
truth
simile

URL of the webpage

https://quotes.toscrape.com/

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.

Python code to scrape the 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.