Web Scraping

LO 4

Objective

After attending this session, you should be able

- Introduction to web scraping
- Beautifulsoup package installation
- Hands on exercise

Web Scraping

- **Web scraping** is the process of gathering information from the Internet.
- However, the term web scraping usually involves **automation**.

Web Scraping

→Python has numerous libraries for approaching this type of problem, many of which are incredibly powerful

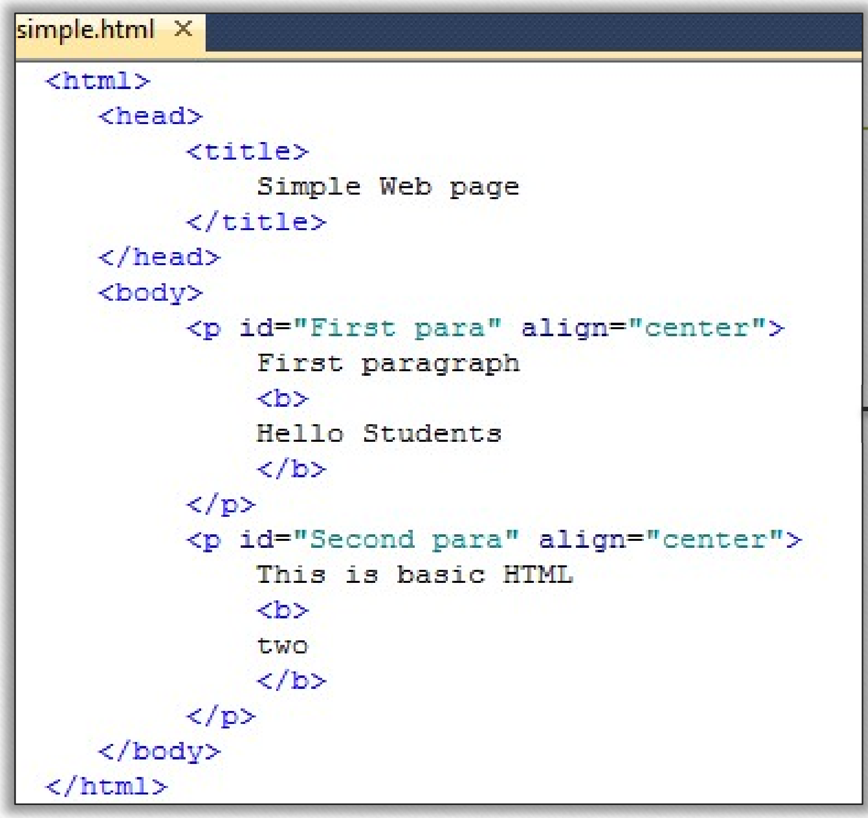
→Popular web scrapping python packages:

- » Pattern
- » Requests
- » Scrapy
- » BeautifulSoup
- » Mechanize

→In this course we are covering Beautiful Soup which is most popular in the lot

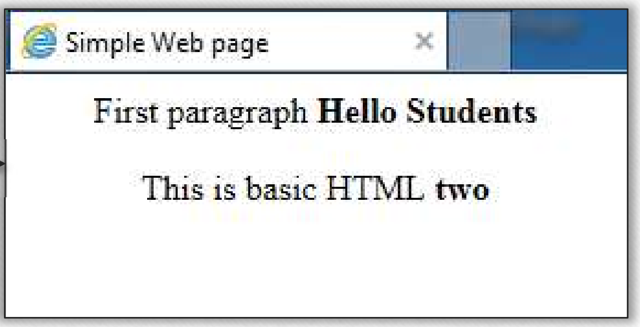
→But these packages can work together too

Typical HTML structure



simple.html

```
<html>
   <head>
        <title>
            Simple Web page
        </title>
   </head>
   <body>
        <p id="First para" align="center">
            First paragraph
            <b>
            Hello Students
            </b>
        </p>
        <p id="Second para" align="center">
            This is basic HTML
            <b>
            two
            </b>
        </p>
   </body>
</html>
```



Simple Web page

First paragraph **Hello Students**

This is basic HTML **two**

Note: Save this file with a .html extension

What's Beautiful Soup?

- Beautiful Soup is a Python library for pulling data out of HTML and XML files via screen scraping
- Three key features:
  - It provides a few simple methods for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need.
  - It converts incoming documents to Unicode and outgoing documents to UTF-8, so you don't have to think about encodings
  - It sits on top of popular Python parsers like **lxml** and **html5lib**, allowing you to try out different parsing strategies or trade speed for flexibility.
  - A **parser** is a compiler or interpreter component that breaks data into smaller elements for easy translation into another language
  - **lxml** provides a very simple and powerful API for parsing XML and HTML.
  - **html5lib** is a pure-python library for parsing HTML

BeautifulSoup Installation

→ If you run Debian or Ubuntu, you can install Beautiful Soup with the system package manager:

» sudo apt-get install python-bs4

→ To install from PyPi:

» easy_install beautifulsoup4
or
pip install beautifulsoup4

→ If you have downloaded the source tarball and want to install manually:

» python setup.py install

→ Refer http://www.crummy.com/software/BeautifulSoup/bs4/doc/#installing-beautiful-soup to avoid any installation related errors and to install other useful packages like lxml parser

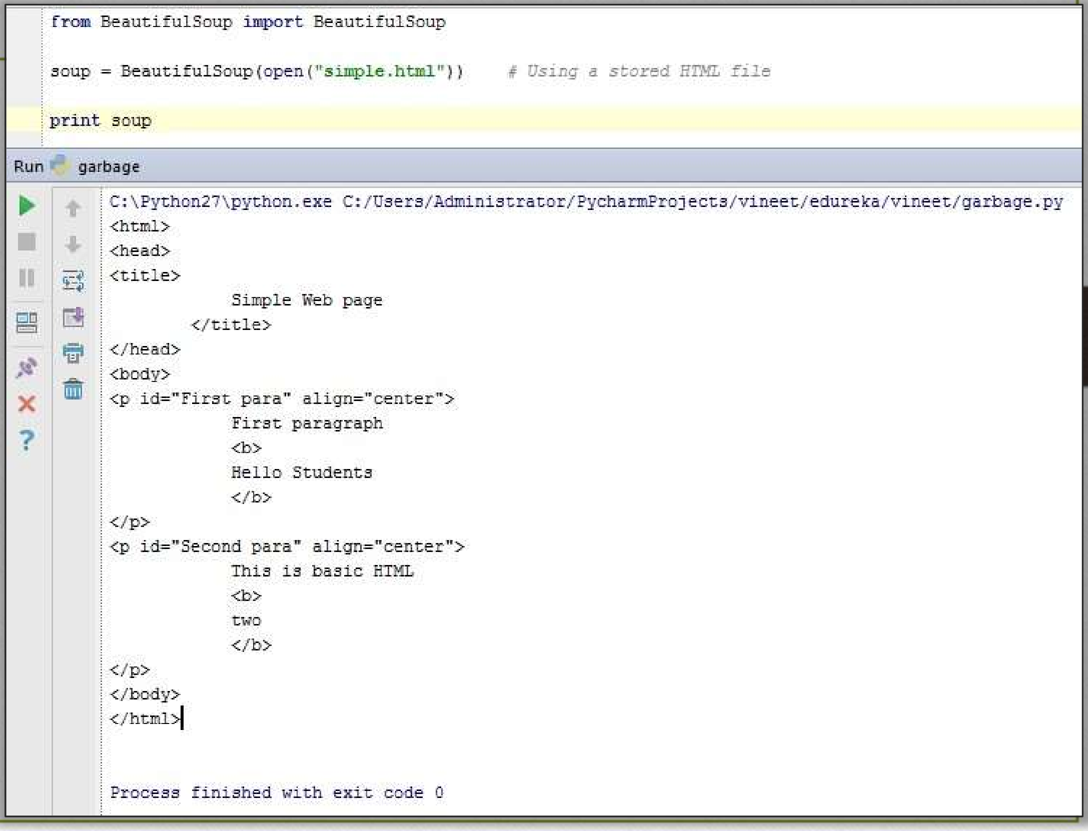
BeautifulSoup for Parsing a Doc

→ To parse a document, pass it into the BeautifulSoup constructor

→ We can pass in a string or an open filehandle

→ Example:

```
from bs4 import BeautifulSoup

# Using a stored HTML file
soup = BeautifulSoup(open("simple.html"))

# Entire HTML doc can be passed
soup = BeautifulSoup("<html>data</html>")
```



```
from BeautifulSoup import BeautifulSoup

soup = BeautifulSoup(open("simple.html"))     # Using a stored HTML file

print soup
```

```
C:\Python27\python.exe C:/Users/Administrator/PycharmProjects/vineet/edureka/vineet/garbage.py
<html>
<head>
<title>
            Simple Web page
        </title>
</head>
<body>
<p id="First para" align="center">
            First paragraph
            <b>
            Hello Students
            </b>
</p>
<p id="Second para" align="center">
            This is basic HTML
            <b>
            two
            </b>
</p>
</body>
</html>


Process finished with exit code 0
```

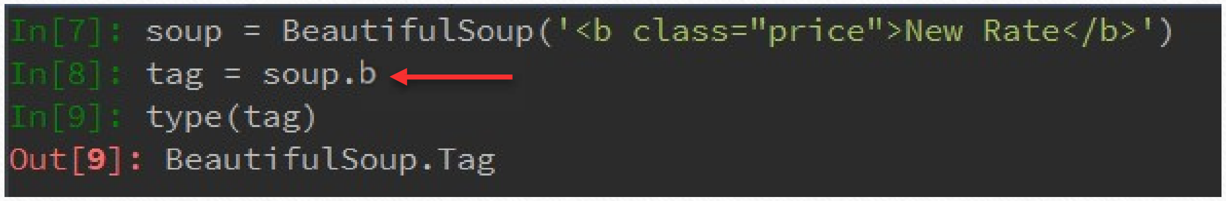
Different Objects

→ Beautiful Soup transforms a complex HTML document into a complex tree of Python objects.

→ Example: soup = BeautifulSoup('<b class="price">New Rate </b>')

→ Tag Object:

» A Tag object corresponds to a HTML tag in the original document.



```
In[7]: soup = BeautifulSoup('<b class="price">New Rate</b>')
In[8]: tag = soup.b
In[9]: type(tag)
Out[9]: BeautifulSoup.Tag
```

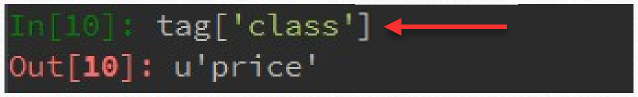
→ Attributes Object:

» A tag may have any number of attributes.

» The tag <p class="price"> has an attribute "class" whose value is "price".

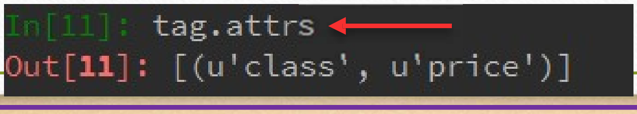
» You can access a tag's attributes by treating the tag like a dictionary:

» tag ['class']



```
In[10]: tag['class']
Out[10]: u'price'
```

» Access attributes by .attrs:

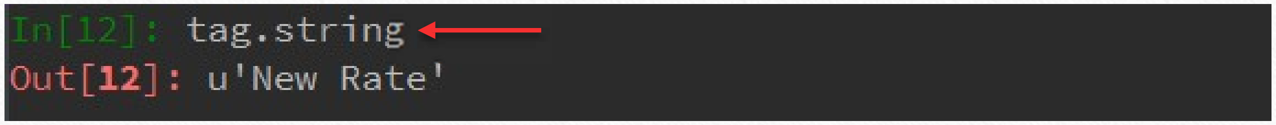


```
In[11]: tag.attrs
Out[11]: [(u'class', u'price')]
```

Different Objects(Contd.)

→ NavigableString Object:

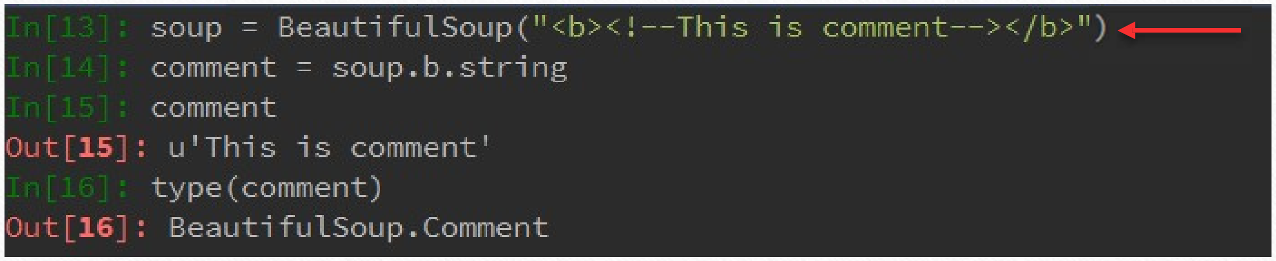
- » Beautiful Soup uses the NavigableString class to contain bits of text within a tag:



```
In[12]: tag.string
Out[12]: u'New Rate'
```

→ Comments:

- » This is a special type of NavigableString Object:



```
In[13]: soup = BeautifulSoup("<b><!--This is comment--></b>")
In[14]: comment = soup.b.string
In[15]: comment
Out[15]: u'This is comment'
In[16]: type(comment)
Out[16]: BeautifulSoup.Comment
```

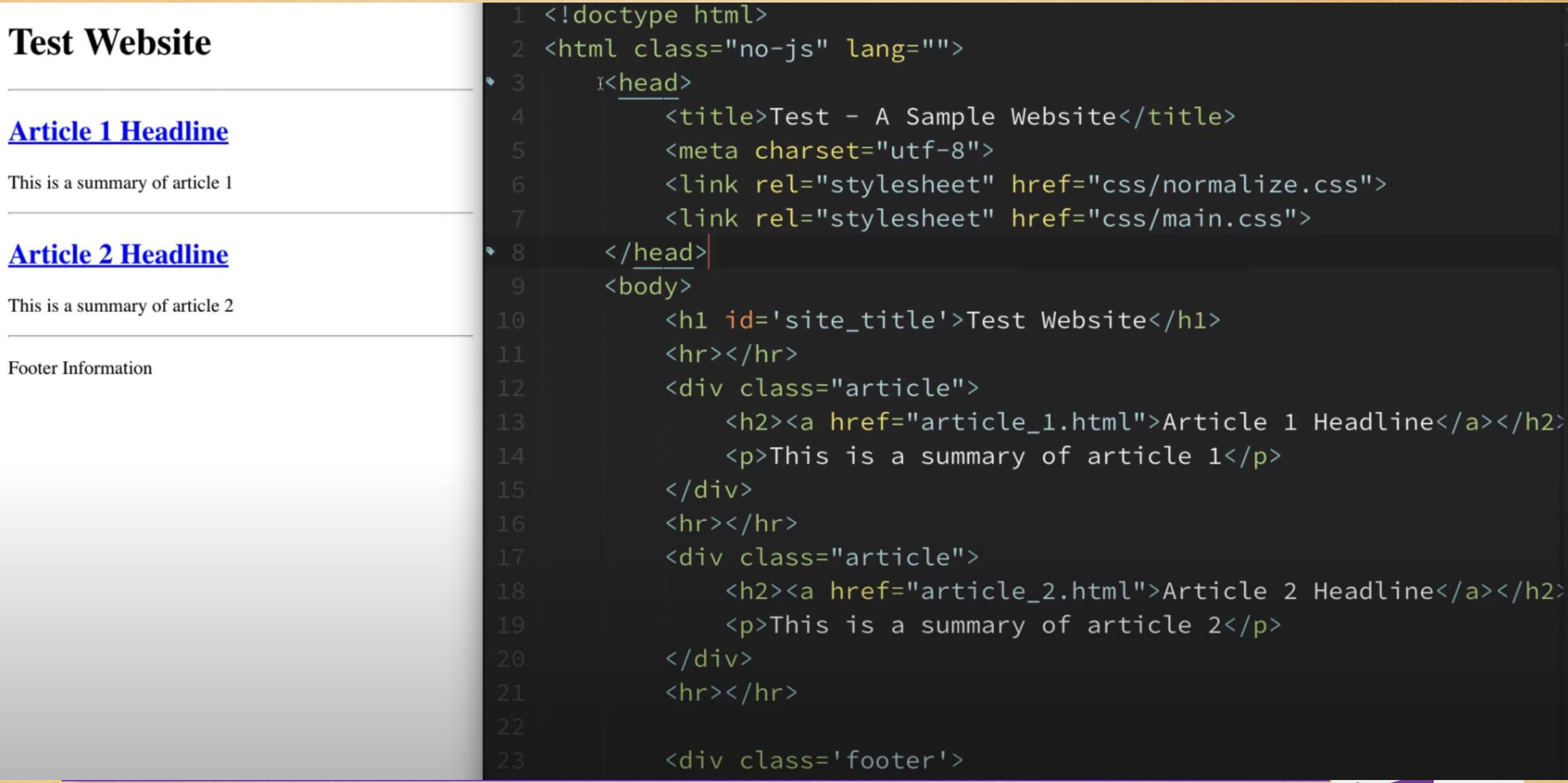
Let's explore practical

- Installation of modules

Pip install beautifulsoup4

Pip install lxml   #parsers to correct missing information of the html data

Pip install requests # allows you to send HTTP requests using Python



Test Website

Article 1 Headline

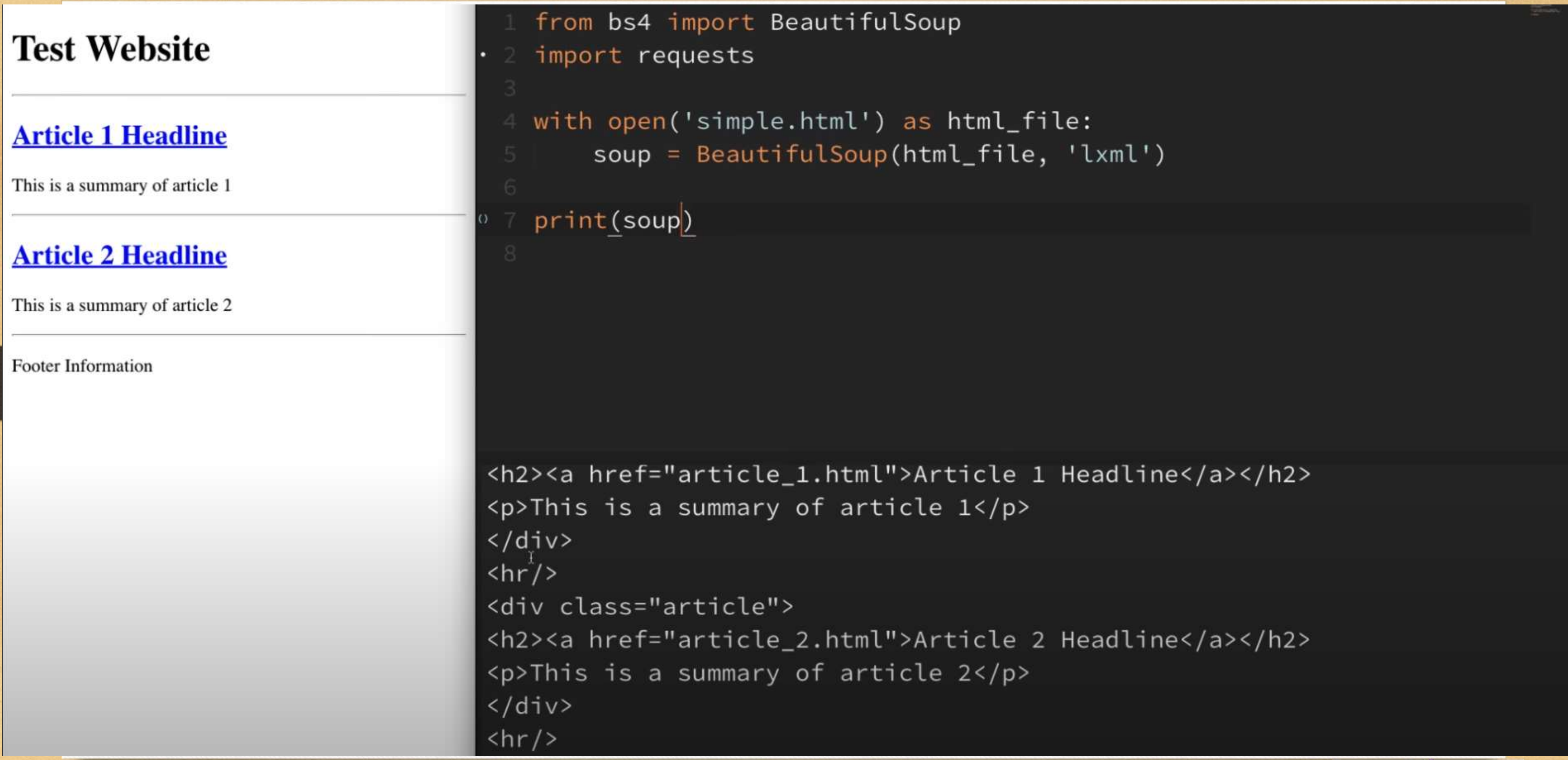
This is a summary of article 1

Article 2 Headline

This is a summary of article 2

Footer Information

```html
<!doctype html>
<html class="no-js" lang="">
    <head>
        <title>Test - A Sample Website</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="css/normalize.css">
        <link rel="stylesheet" href="css/main.css">
    </head>
    <body>
        <h1 id='site_title'>Test Website</h1>
        <hr></hr>
        <div class="article">
            <h2><a href="article_1.html">Article 1 Headline</a></h2
            <p>This is a summary of article 1</p>
        </div>
        <hr></hr>
        <div class="article">
            <h2><a href="article_2.html">Article 2 Headline</a></h2
            <p>This is a summary of article 2</p>
        </div>
        <hr></hr>

        <div class='footer'>
```



Test Website

Article 1 Headline

This is a summary of article 1

Article 2 Headline

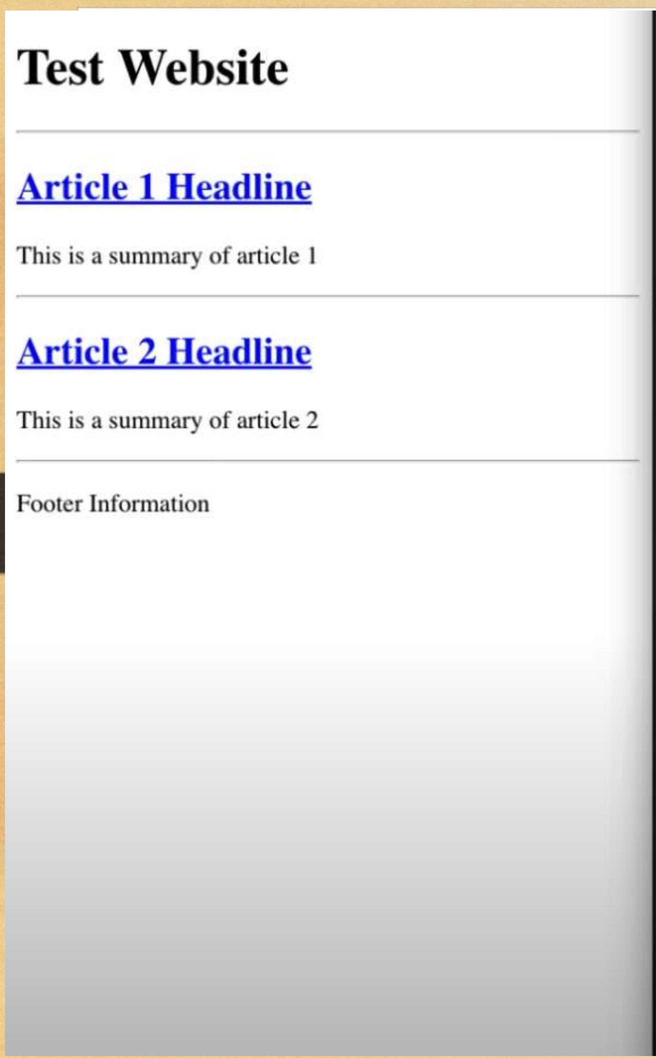
This is a summary of article 2

Footer Information

```python
from bs4 import BeautifulSoup
import requests

with open('simple.html') as html_file:
    soup = BeautifulSoup(html_file, 'lxml')

print(soup)

```

```
<h2><a href="article_1.html">Article 1 Headline</a></h2>
<p>This is a summary of article 1</p>
</div>
<hr/>
<div class="article">
<h2><a href="article_2.html">Article 2 Headline</a></h2>
<p>This is a summary of article 2</p>
</div>
<hr/>
```

Prettify method



Test Website

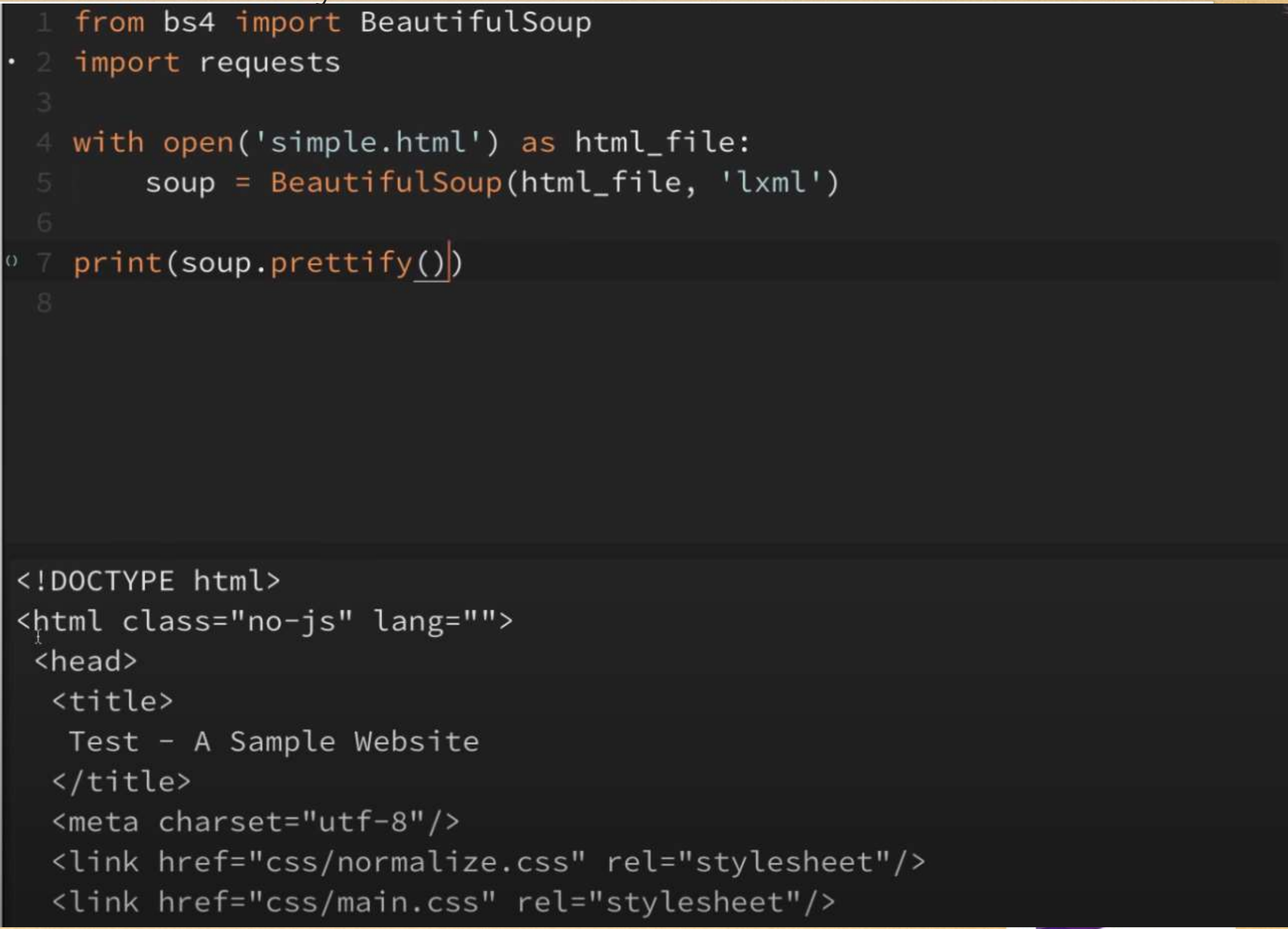
Article 1 Headline

This is a summary of article 1

Article 2 Headline

This is a summary of article 2

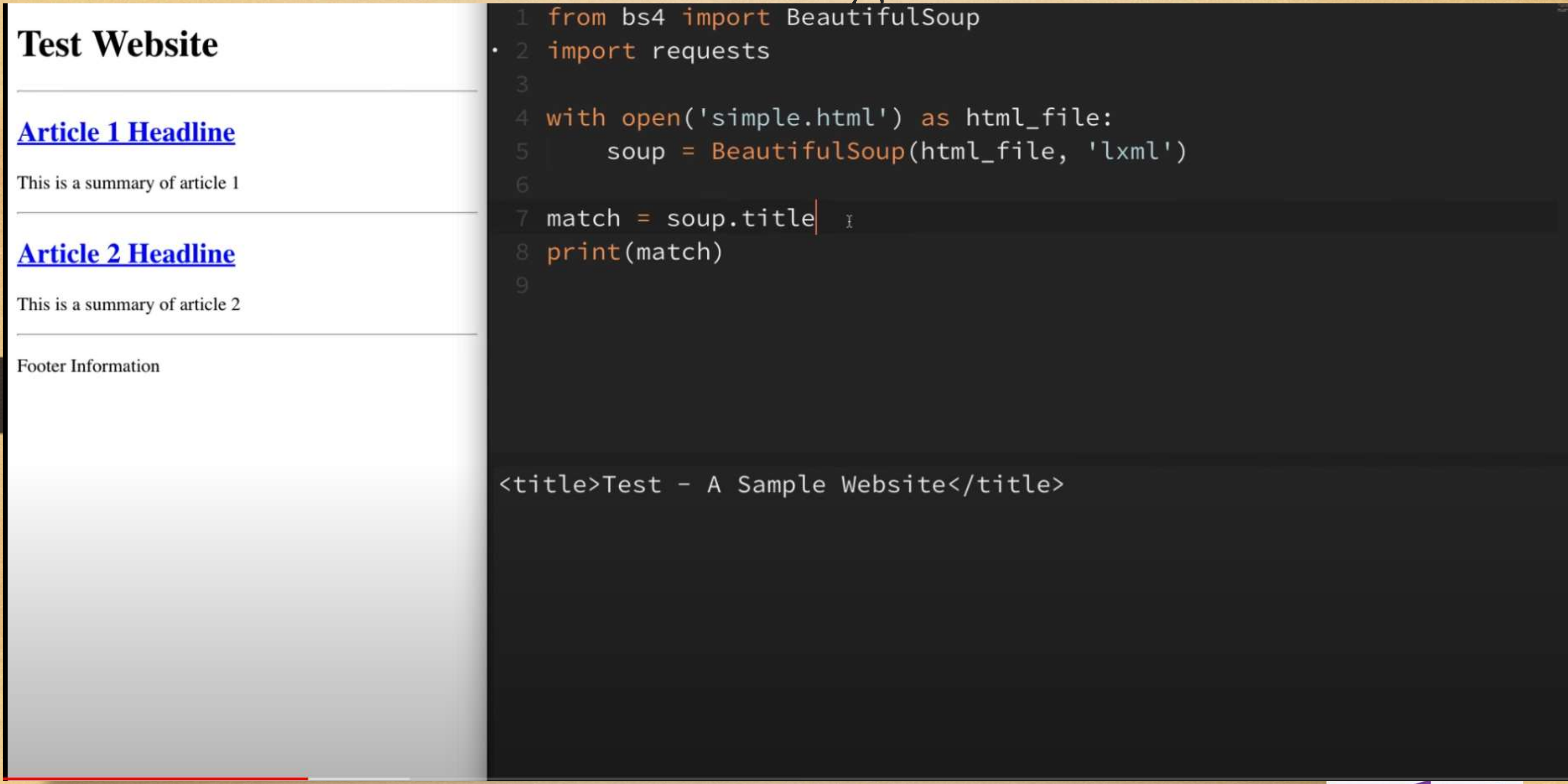
Footer Information



```python
from bs4 import BeautifulSoup
import requests

with open('simple.html') as html_file:
    soup = BeautifulSoup(html_file, 'lxml')

print(soup.prettify())

```

```
<!DOCTYPE html>
<html class="no-js" lang="">
 <head>
  <title>
   Test - A Sample Website
  </title>
  <meta charset="utf-8"/>
  <link href="css/normalize.css" rel="stylesheet"/>
  <link href="css/main.css" rel="stylesheet"/>
```

Title tag



Test Website

Article 1 Headline

This is a summary of article 1

Article 2 Headline

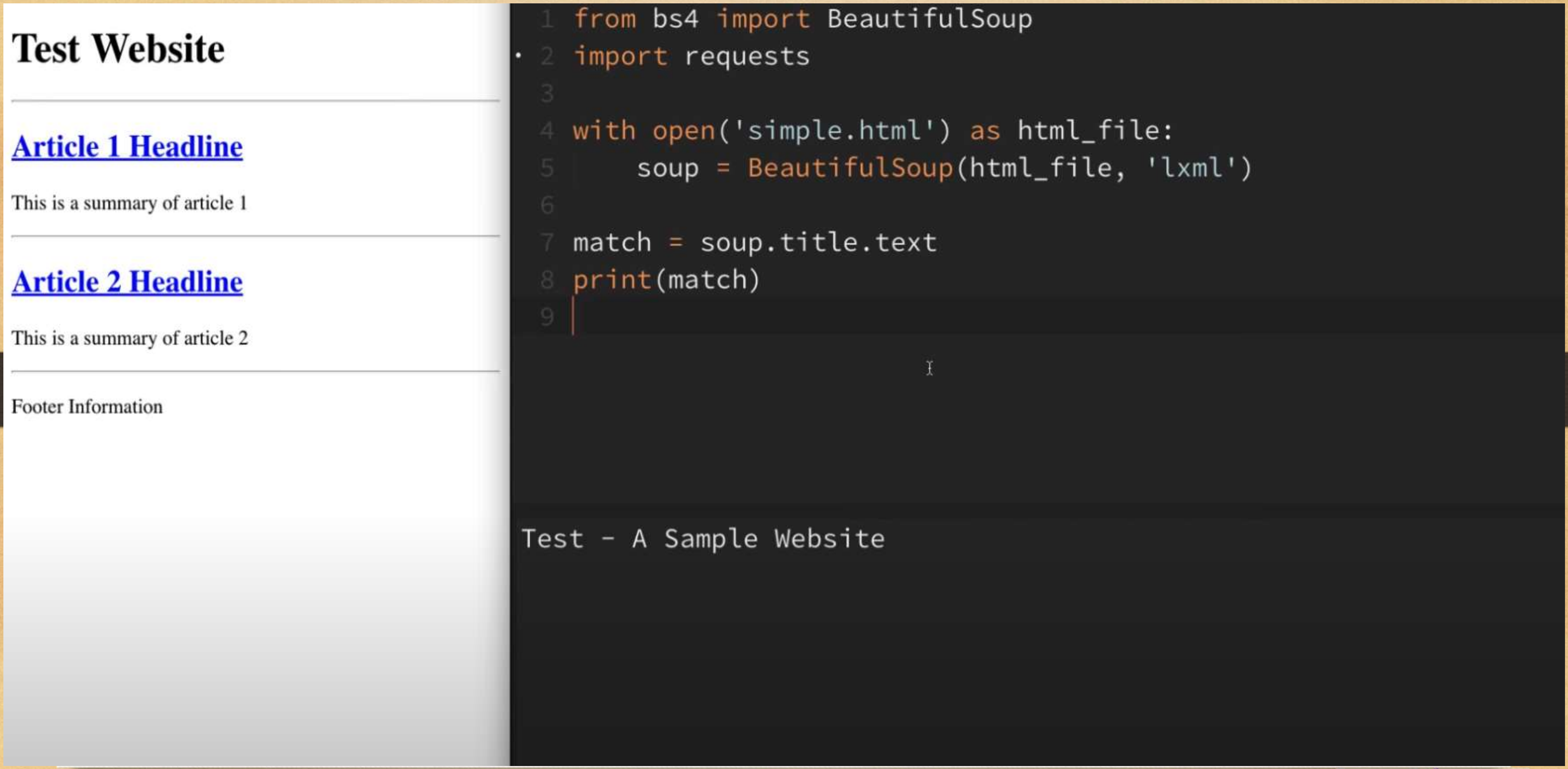
This is a summary of article 2

Footer Information

```python
from bs4 import BeautifulSoup
import requests

with open('simple.html') as html_file:
    soup = BeautifulSoup(html_file, 'lxml')

match = soup.title
print(match)
```

```
<title>Test - A Sample Website</title>
```

Title Text



Test Website

Article 1 Headline

This is a summary of article 1

Article 2 Headline

This is a summary of article 2

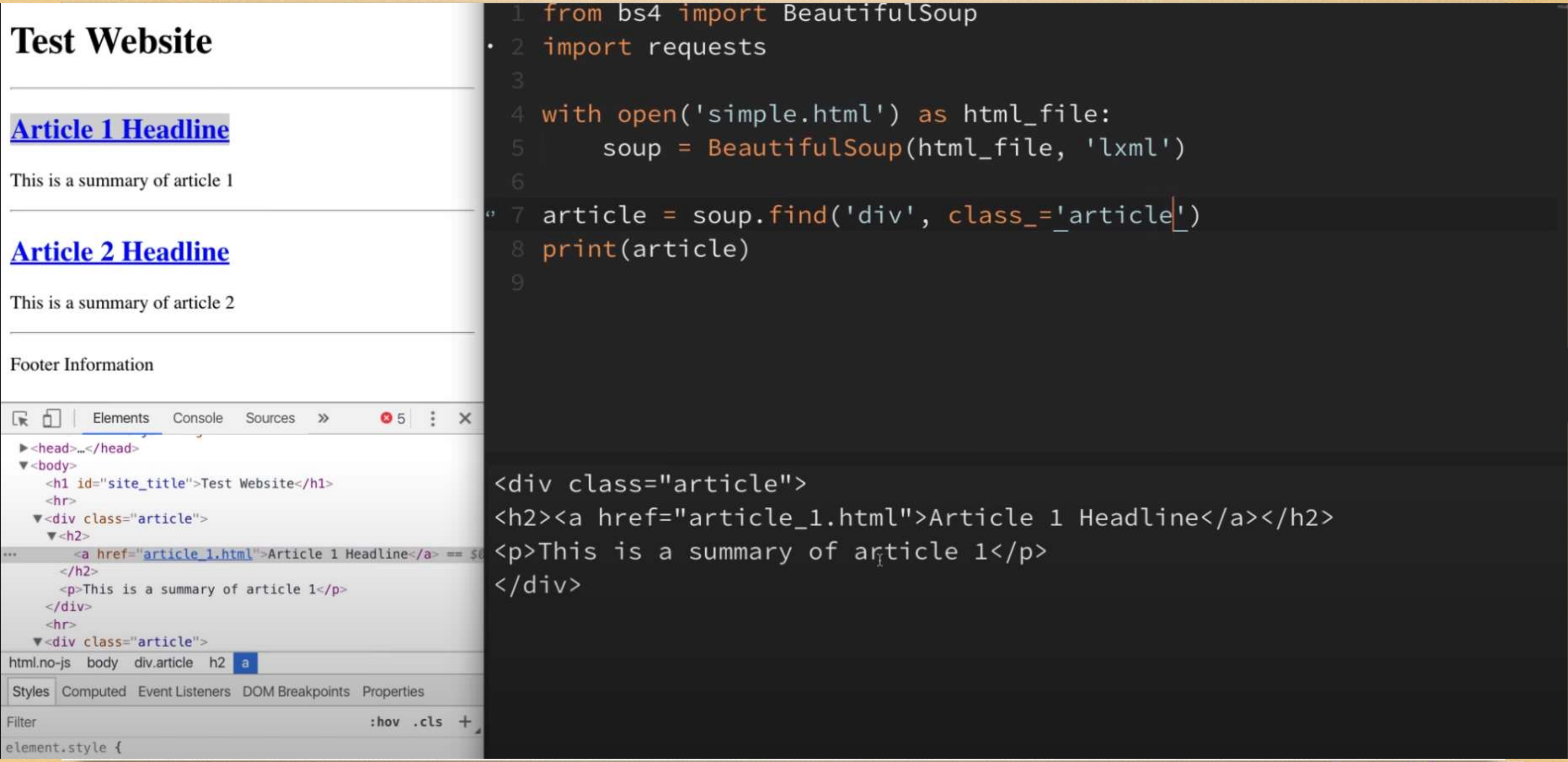
Footer Information

```python
from bs4 import BeautifulSoup
import requests

with open('simple.html') as html_file:
    soup = BeautifulSoup(html_file, 'lxml')

match = soup.title.text
print(match)

```

```
Test - A Sample Website
```

Try yourself

- Match =soup.div   #first div
- Match = soup.find('div') #first div
- Match = soup.find('div', class_='_footer_') # class is keyword in python
- Inspect the html page



Test Website

Article 1 Headline

This is a summary of article 1

Article 2 Headline

This is a summary of article 2

Footer Information

Elements
Console
Sources
5
html.no-js body div.article h2 a
Styles Computed Event Listeners DOM Breakpoints Properties
Filter :hov .cls +

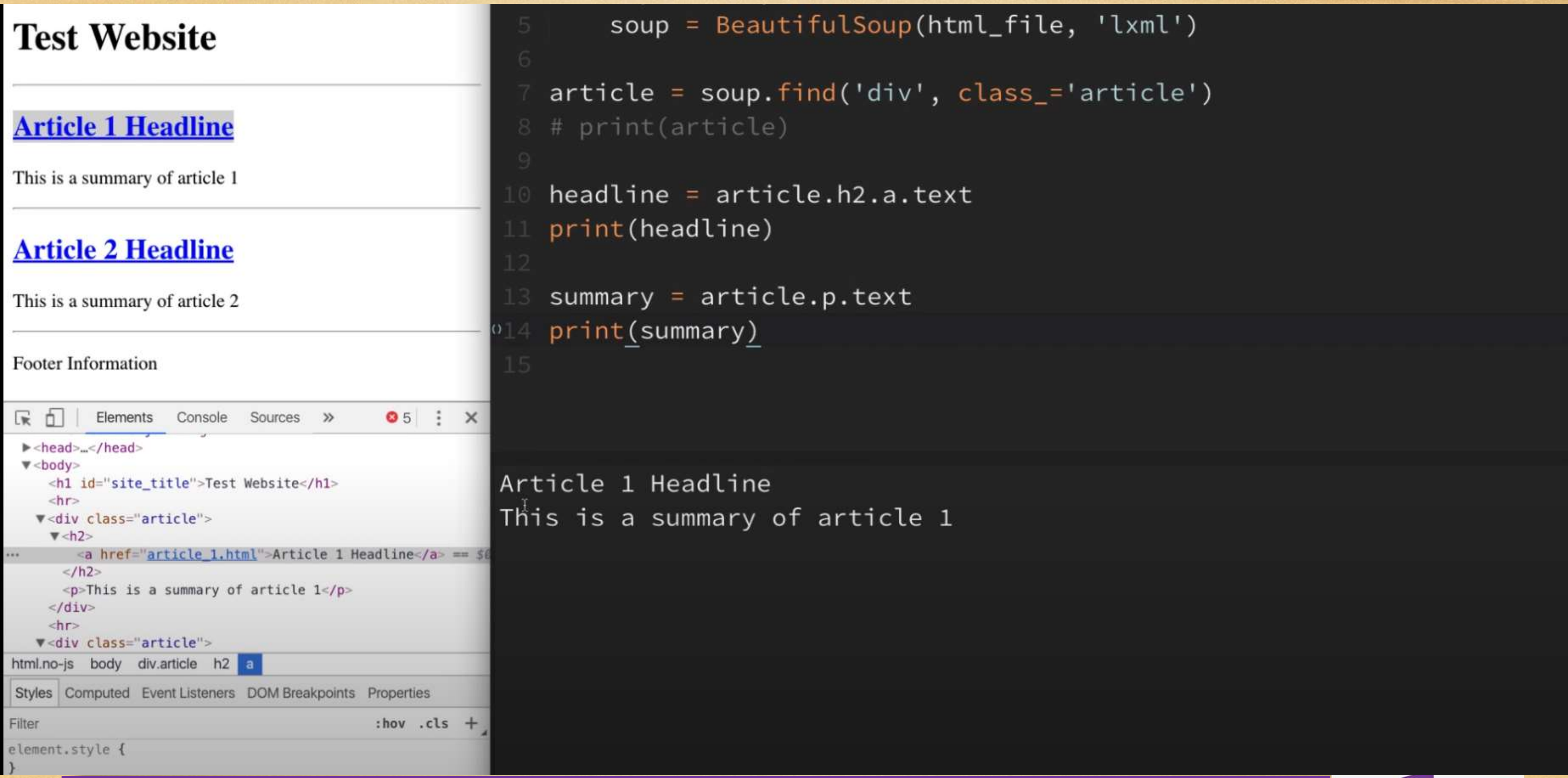
```
▶<head>…</head>
▼<body>
   <h1 id="site_title">Test Website</h1>
   <hr>
  ▼<div class="article">
    ▼<h2>
        <a href="article_1.html">Article 1 Headline</a> == $0
      </h2>
      <p>This is a summary of article 1</p>
   </div>
   <hr>
  ▼<div class="article">
```

```
element.style {
```

```python
from bs4 import BeautifulSoup
import requests

with open('simple.html') as html_file:
    soup = BeautifulSoup(html_file, 'lxml')

article = soup.find('div', class_='article')
print(article)
```

```
<div class="article">
<h2><a href="article_1.html">Article 1 Headline</a></h2>
<p>This is a summary of article 1</p>
</div>
```

SASKATCHEWAN POLYTECHNIC Tomorrow in the making



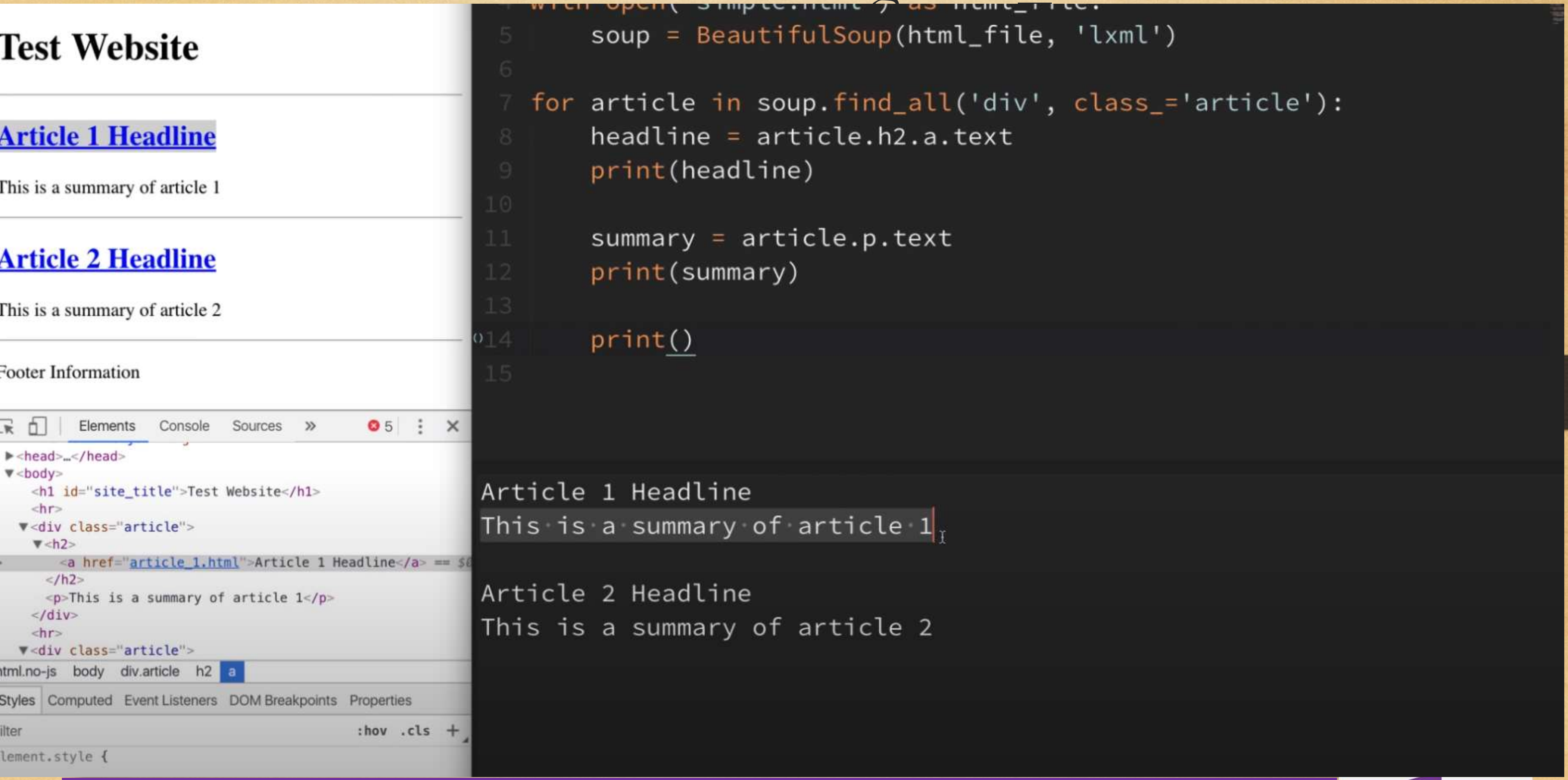
```html
▶<head>…</head>
▼<body>
  <h1 id="site_title">Test Website</h1>
  <hr>
 ▼<div class="article">
  ▼<h2>
     <a href="article_1.html">Article 1 Headline</a> == $0
   </h2>
   <p>This is a summary of article 1</p>
  </div>
  <hr>
 ▼<div class="article">
```

```
element.style {
}
```

```python
    soup = BeautifulSoup(html_file, 'lxml')

article = soup.find('div', class_='article')
# print(article)

headline = article.h2.a.text
print(headline)

summary = article.p.text
print(summary)
```

```
Article 1 Headline
This is a summary of article 1
```

All articles using find_all



```
<head>…</head>
<body>
  <h1 id="site_title">Test Website</h1>
  <hr>
  <div class="article">
    <h2>
      <a href="article_1.html">Article 1 Headline</a>
    </h2>
    <p>This is a summary of article 1</p>
  </div>
  <hr>
  <div class="article">
```

```python
    soup = BeautifulSoup(html_file, 'lxml')

for article in soup.find_all('div', class_='article'):
    headline = article.h2.a.text
    print(headline)

    summary = article.p.text
    print(summary)

    print()
```

```
Article 1 Headline
This is a summary of article 1

Article 2 Headline
This is a summary of article 2
```

Summary

- Introduced web scraping
- Beautifulsoup package installation
- Hands on exercise

Himanshu Patel, Instructor
Saskatchewan Polytechnic
email: patelh@saskpolytech.ca
Mining building, Saskatoon