Contents

[Notes 2](#_Toc13826659)

[Difference between urllib and requests: 2](#_Toc13826660)

[CoreyMS - Web Scraping with BeautifulSoup and Requests 3](#_Toc13826661)

[e.g.01 3](#_Toc13826662)

[e.g.02 4](#_Toc13826663)

[BradTraversy 5](#_Toc13826664)

[# ATTRS 5](#_Toc13826665)

[# SELECT - like jquery 5](#_Toc13826666)

[# get\_text() 5](#_Toc13826667)

[# contents 6](#_Toc13826668)

[# next\_sibling, find\_next\_sibling(), find\_previous\_sibling(), find\_parent(), find\_next\_sibling('tag') 6](#_Toc13826669)

[Data Science Dojo 7](#_Toc13826670)

[freeCodeCamp.org 8](#_Toc13826671)

[e.g.01, Basic Introduction of BeautifulSoup and requests 8](#_Toc13826672)

[e.g.02, web Scrapping 100 pages of white house website 9](#_Toc13826673)

[e.g.03 - Summary of freeCodeCamp $V: 10](#_Toc13826674)

[e.g.03 – BeautifulSoup Objects 11](#_Toc13826675)

[# Tag: 11](#_Toc13826676)

[# Name: 12](#_Toc13826677)

[# Attributes: 12](#_Toc13826678)

[BradTraversy 14](#_Toc13826679)

# Notes

The following lines of codes are identical

shipping\_price = container.find('li', {'class':'price-ship'}).text.strip()

shipping\_price = container.find('li', 'price-ship').text.strip()

shipping\_price = container.find('li', class\_='price-ship').text.strip()

containers = page\_soup.findAll('div', class\_="item-container")

containers = page\_soup.find\_all('div', class\_="item-container")

namely findAll and find\_all are identical.

**Documentation:** In older versions of Beautiful Soup, which don’t have the class\_ shortcut, you can use the attrs trick mentioned above. Create a dictionary whose value for “class” is the string (or regular expression, or whatever) you want to search for.

Plus, the followings are identical as well:

uClient = urlopen(my\_url)

page\_html = uClient.read()

uClient.close()

**and**

page\_html = requests.get(my\_url).text

# [Difference between urllib and requests:](https://stackoverflow.com/questions/2018026/what-are-the-differences-between-the-urllib-urllib2-and-requests-module/39422481#39422481)

<https://stackoverflow.com/questions/2018026/what-are-the-differences-between-the-urllib-urllib2-and-requests-module/39422481#39422481>

# CoreyMS - Web Scraping with BeautifulSoup and Requests

**BeautifulSoup** is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

**lxml** is the most feature-rich and easy-to-use library for **processing XML and HTML** in the Python language.

**Requests** is an Apache2 Licensed HTTP library, written in Python. ... **Requests** will allow you to send HTTP/1.1 **requests** using Python. With it, you can add content like headers, form data, multipart files, and parameters via simple Python libraries. It also allows you to access the response data of Python in the same way.

Requests allows us to obtain information from various websites.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pip install beautifulsoup4

Pip install lxml

Pip install html5lib

Pip install requests

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## e.g.01

from bs4 import BeautifulSoup

import requests

with open('simple.html') as html\_file:

soup = BeautifulSoup(html\_file, 'lxml')

# print(soup.prettify()) # prints entire content in nice format

match = soup.title # <title>Test - A Sample Website</title>

match1 = soup.title.text # Test - A Sample Website

match2 = soup.find('div', class\_='footer')

print(match2)

# <div class="footer">

# <p>Footer Information</p>

# </div>

for article in soup.find\_all('div', class\_='article'):

headline = article.h2.a.text

print(headline)

summary = article.p.text

print(summary)

print()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## e.g.02

from bs4 import BeautifulSoup

import requests

import csv

source = requests.get('http://coreyms.com').text

soup = BeautifulSoup(source, 'lxml')

# create and write on csv file

csv\_file = open('cms\_scrape.csv', 'w')

csv\_writer = csv.writer(csv\_file)

csv\_writer.writerow(['headline', 'summary', 'video\_link'])

for article in soup.find\_all('article'):

headline = article.h2.a.text

print(headline)

summary = article.find('div', class\_='entry-content').p.text

print(summary)

try:

vid\_src = article.find('iframe', class\_='youtube-player')['src']

vid\_id = vid\_src.split('/')[4]

vid\_id = vid\_id.split('?')[0]

yt\_link = f'https://youtube.com/watch?v={vid\_id}'

except Exception as e:

yt\_link = None

print(yt\_link)

print()

csv\_writer.writerow([headline, summary, yt\_link])

csv\_file.close()

# BradTraversy

from bs4 import BeautifulSoup

html\_var = """"

<html>

<body>

<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters;

their names:

<a href="http://example.com/elsie" class="sister"

data-head='BestFriend' id="link1">Elsie</a>,

<a href="http://example.com/lacie" class="sister"

id="link2">Lacie</a> and

<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;

and they lived at the bottom of a well.</p>

<p class="story">...</p>

<div data-content='slug'>Lorem Ipsum Dolor Set Amet</div>

<b class="boldest">Extremely bold</b>

<b id="first\_test">Test 1</b>

</body>

</html>

"""

soup = BeautifulSoup(html\_var, 'html.parser')

# find items based on id

res1 = soup.find(id='link1')

# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>

## # ATTRS

res2 = soup.find(attrs={'data-head':'BestFriend'})

# <a class="sister" data-head="BestFriend" href="http://example.com/elsie" id="link1">Elsie</a>

## # SELECT - like jquery

res3 = soup.select('#first\_test') # id='first\_test'

res3\_1 = soup.select('.item') # class='item'

# [<b id="first\_test">Test 1</b>]

## # get\_text()

res4 = soup.find(id='first\_test').get\_text()

# Test 1

## # contents

res5 = soup.body.contents

# this will print out the entire elements of body as list

res6 = soup.select('.title')[0].contents

# [<b>The Dormouse's story</b>]

## # next\_sibling, find\_next\_sibling(), find\_previous\_sibling(), find\_parent(), find\_next\_sibling('tag')

res7 = soup.select('.story')[0].next\_sibling.next\_sibling

# <p class="story">...</p>

res8 = soup.select('.story')[0].find\_next\_sibling()

# <p class="story">...</p>

res9 = soup.select('.story')[0].find\_previous\_sibling()

# <p class="title"><b>The Dormouse's story</b></p>

res10 = soup.select('b')[0].find\_parent()

# <p class="title"><b>The Dormouse's story</b></p>

res11 = soup.select('.story')[0].find\_next\_sibling('div')

# <div data-content="slug">Lorem Ipsum Dolor Set Amet</div>

# Data Science Dojo

from urllib.request import urlopen as uReq

from bs4 import BeautifulSoup as soup

import requests

my\_url = 'https://www.newegg.com/Video-Cards-Video-Devices/Category/ID-38?Tpk=graphics%20cards'

uClient = uReq(my\_url)

page\_html = uClient.read()

uClient.close()

# (NOTE): if you don't add text at the end you will get this error: [object of type 'Response' has no len()]

# (ALTERNATIVE APPROACH): page\_html = requests.get(my\_url).text

page\_soup = soup(page\_html, 'html.parser')

# (ALT): page\_soup = soup(page\_html, 'lxml')

containers = page\_soup.findAll('div', {"class":"item-container"})

# (ALT): containers = page\_soup.findAll('div', class\_="item-container")

csv\_export = open('product\_info.csv', 'w')

csv\_export.write('brand, product\_name, shipping\_price \n')

for container in containers:

try:

brand = container.find('div', 'item-info').find('div', 'item-branding').a.img["title"]

# (ALT): brand = container.find('div', class\_='item-info').find('div', class\_='item-branding').a.img["title"]

title = container.find('a', class\_= 'item-title').text

# (ALT): title = container.find('a', {'class':'item-title'}).text

shipping\_price = container.find('li', {'class':'price-ship'}).text.strip()

# (ALT): shipping\_price = container.find('li', class\_='price-ship').text.strip()

except Exception:

brand = 'Brand not specified'

title = 'Title not specified'

shipping\_price = 'Shipping price not specified'

csv\_export.write(brand + ',' + title.replace(',', '|') + ',' + shipping\_price + '\n')

# freeCodeCamp.org

## e.g.01, Basic Introduction of BeautifulSoup and requests

import requests

from bs4 import BeautifulSoup

# Using the requests module, we use the "get" function provided to access the

# webpage provided as an argument to this function:

result = requests.get("https://www.google.com/")

# To make sure that the website is accessible, we can ensure that we obtain a 200

# OK response to indicate that the page is indeed present:

print(result.status\_code)

# For other status codes: https://en.wikipedia.org/wiki/List\_of\_HTTP\_status\_codes

# We can also check the HTTP header of the website to

verify that we have indeed accessed the correct page:

print(result.header**s**) # not print(src.headers)

# For more information on HTTP headers:

https://en.wikipedia.org/wiki/List\_of\_HTTP\_header\_fields

# let us store the page content of the website accessed from requests to a variable:

src = result.content

# Now that we have the page source stored, we will use the BeautifulSoup module

# to parse and process the source. To do so, we create a BeautifulSoup object

# based on the source variable we created above:

soup = BeautifulSoup(src, 'lxml')

# Now that the page source has been processed via Beautifulsoup

# we can access specific information directly from it. For instance,

# say we want to see a list of all of the links on the page:

links = soup.find\_all("a")

print(links)

# We want to extract the link that contains the text "About" on the page instead of every

# link. We can use the built-in "text" to access the text content between the

# <a> </a> tags.

for link in links:

if "About" in link.text:

print(link)

print(link.attrs['href'])

## e.g.02, web Scrapping 100 pages of white house website

import requests

from bs4 import BeautifulSoup

import csv

file = open('white\_house\_briefing.csv', 'w', encoding='utf-8')

writer = csv.writer(file)

writer.writerow(['title', 'link', 'author', 'publish\_date'])

# looping through 100 pages of this website to webscrape

for i in range(100):

url = 'https://www.whitehouse.gov/briefings-statements/page/{0}/'.format(i)

result = requests.get(url)

src = result.content

soup = BeautifulSoup(src, 'lxml')

articles = soup.find\_all('article', class\_='briefing-statement')

for article in articles:

try:

title = article.find('h2',

{'class':'briefing-statement\_\_title'}).a.text

link = article.find('h2',

{'class':'briefing-statement\_\_title'}).a.attrs['href']

author = article.find('p',

{'class':'issue-flag'}).find('a').text.strip()

publish\_date = article.find('p', 'meta\_\_date').time.text.strip()

except Exception as e:

title = 'No title'

link = 'No link'

author = 'No author'

publish\_date = 'No publish\_date'

writer.writerow([title, link, author, publish\_date])

## e.g.03 - Summary of freeCodeCamp $V:

html\_doc = """

<html>

<head>

<title>The Dormouse's story</title>

</head>

<body>

<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were 3 little sisters; their names:

<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,

<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and

<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;

and they lived at the bottom of a well.</p>

...

</body>

</html>

"""

with open('new\_file.html', 'w') as file:

\_new = file.write(html\_doc)

soup = BeautifulSoup(html\_doc, 'lxml')

res = soup.find\_all('a')[1]

# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>

res1 = soup.find\_all('a')[1].attrs

# {'href': 'http://example.com/lacie', 'class': ['sister'], 'id': 'link2'}

res3 = soup.find\_all('a')[1].string

# Lacie

del res['class']

# <a href="http://example.com/lacie" id="link2">Lacie</a> # class is deleted

res4 = soup.find\_all('a')[1].attrs['href']

# <http://example.com/lacie>

res5 = soup.find(id='section')

## e.g.03 – BeautifulSoup Objects

# YouTube Link: https://www.youtube.com/watch?v=oDtLJEc5Ako

from bs4 import BeautifulSoup

# To keep things simple and also reproducible, consider the following HTML code

html\_doc = """

<html>

<head>

<title>The Dormouse's story</title>

</head>

<body>

<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters;

their names:

<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,

<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and

<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;

and they lived at the bottom of a well.</p>

<p class="story">...</p>

<b class="boldest">Extremely bold</b>

<blockquote class="boldest">Extremely bold</blockquote>

<b id="1">Test 1</b>

<b another-attribute="1" id="verybold">Test 2</b>

</body>

</html>

"""

with open('index.html', 'w') as f:

f.write(html\_doc)

soup = BeautifulSoup(html\_doc, "lxml")

print(soup.prettify())

### # Tag:

# Finds the first occurrence of usage for a "b" bold tag.

print(soup.b)

# The "find" function also does the same, where it only finds the first

# occurrence in the HTML doc of a tag with "b".

print(soup.find('b'))

# If we want to find all of the elements on the page with the "b" tag, we can use

# the "find\_all" function.

print(soup.find\_all('b'))

### # Name:

# This gives the name of the tag. In this case, the

# tag name is "b".

print(soup.b.name)

# We can alter the name and have that reflected in the

# source. For instance:

tag = soup.b

print(tag)

tag.name = "blockquote"

print(tag)

### # Attributes:

tag = soup.find\_all('b')[2] # notice we used index, [2], with find\_all not find

print(tag)

# This specific tag has the attribute "id", which can be accessed like so:

print(tag['id'])

tag = soup.find\_all('b')[3]

print(tag)

# We can even access multiple attributes that are non-standard HTML attributes:

print(tag['id'])

print(tag['another-attribute'])

# If we want to see all attributes, we can access them as a dictionary object:

tag = soup.find\_all('b')[3]

print(tag)

print(tag.attrs)

# These properties are mutable, and we can alter them in the following manner.

print(tag)

tag['another-attribute'] = 2

print(tag)

# We can also use Python's del command for lists to remove attributes:

del tag['id']

del tag['another-attribute']

print(tag)

# Multi-valued Attributes

tag = soup.find\_all('b')[3]

print(tag)

print(tag.string)

# We can use the "replace\_with" function to replace

# the content of the string with something different:

tag.string.replace\_with("This is another string")

print(tag)

# BradTraversy