Data Science

Intro to beautifulSoup

10.23.22

Web Scraping

beautiful soup

What is beautiful soup?

Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages. Say you've found some webpages that display data relevant to your research, such as date or address information, but that do not provide any way of downloading the data directly. Beautiful Soup helps you pull particular content from a webpage, remove the HTML markup, and save the information. It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web.

Web Scraping

Installation

Installing Beautiful Soup:

Installing Beautiful Soup is easiest if you have pip or another Python installer already in place. Run the following command in the terminal to install Beautiful Soup:

```
!pip install beautifulsoup4
pip install beautifulsoup4
```

You will also need to install requests:

```
!pip install requests
pip install requests
```

Web Scraping

Import

Now we need to import the libraries. You will import using the following commands:

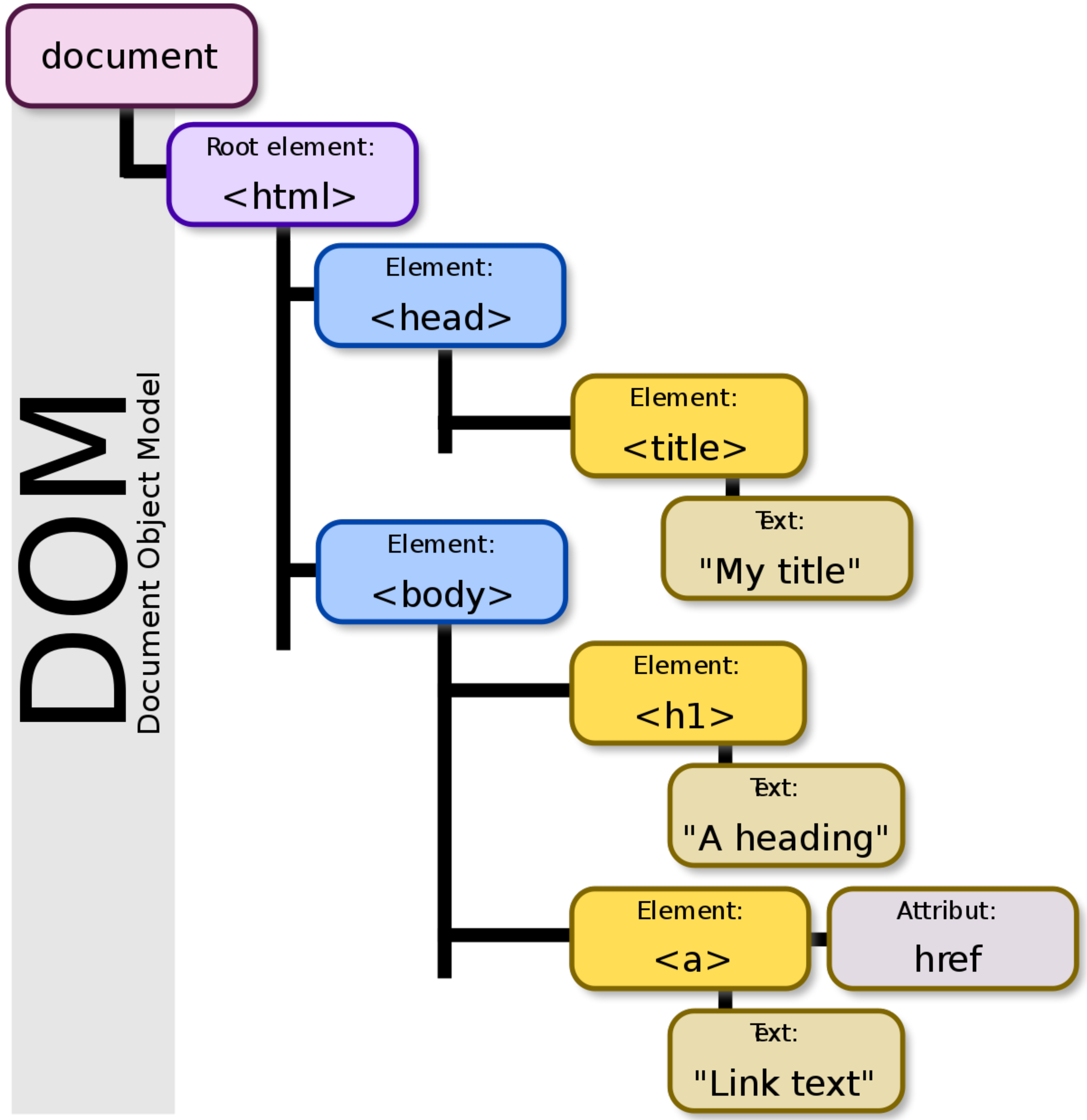
```
from bs4 import BeautifulSoup
import requests
```

Beautiful Soup is a library useful to extract data from HTML and XML files. A sort of parse tree is built for the parsed page. Indeed, an HTML document is composed of a tree of tags. I will show an example of HTML code to make you grasp this concept.

Webscraping

Html

```
<!DOCTYPE html>
<html>
<head>
<title> My Title </title>
</head>
<body>
<h1> 1. A Heading </h1>
<p> paragraph </p>
</body>
</html>
```



Webscraping

Create Response Object

To get the web page, the first step is to create a response object, passing the URL to the get method.

```
url = “some_url.com"
req = requests.get(url)
```

The client communicates with the server using a HyperText Transfer Protocol(HTTP). In this line of code, it’s like when we type the link on the address bar, the browser transmits the request to the server and then the server performs the requested action after it looked at the request.

Webscraping

Create a beautiful soup object

Create the Beautiful Soup object, which parses the document using the HTML parser. By doing this, we transform the HTML code into a tree of Python objects, as shown before in the illustration.

```
soup = BeautifulSoup(req.text, “html.parser”)
print(soup)
```

Webscraping

Some issues

Sometimes we can’t scrape every website because either the server does not recognize you as a browser or the website itself does not load all the information but loads if a user decides to take action on the website.

For the first case we can always hack our way through (as we will see in the demo portion of the class)

In the second case one can take a look at selenium (we will not be covering selenium in this course but one may find it very useful when scraping the web)

Webscraping

Exploring the html tree

We can get access directly to the tags, just writing:

```
soup.head
soup.body
soup.body.h1
```

A more efficient way is to use the `find` and `find_all` methods, which filter the element(s) in case of `find_all` method.

```
row1 = tab.find('tr')
```

Webscraping

Exploring html tree

Using the `find` method, we zoom a part of the document within the tags, which are used to build each row of the table. In this case, we got only the first row because the function extracts only one element. Instead, if we want to gather all the rows of the table, we use the other method:

```
rows = tab.find_all(‘tr')
print(len(rows))
print(rows[0])
```

Webscraping

Exploring html tree

Most items in the html tree are listed by what they are.

What do I mean? For example tables will be listed as <table>

And each item in the table will be under <tr>. For lists there is <li>.

Since many items exist in one table or list you will want to iterate through it to get data quickly. After all, what's the point of web scraping for data if its no faster or efficient than inserting data manually?

```
List = []
table = soup.find('table')
rows = table.find_all('tr')
for row in rows:
   List.append(row)
```