Web Scraping with Beautiful Soup

.

Introduction to Beautiful Soup

- Web scraping use cases:
  - Existing blog → New blog Resource page

- Use Python to go through your blog for you, and extract every link from every page

Web scraping use cases:

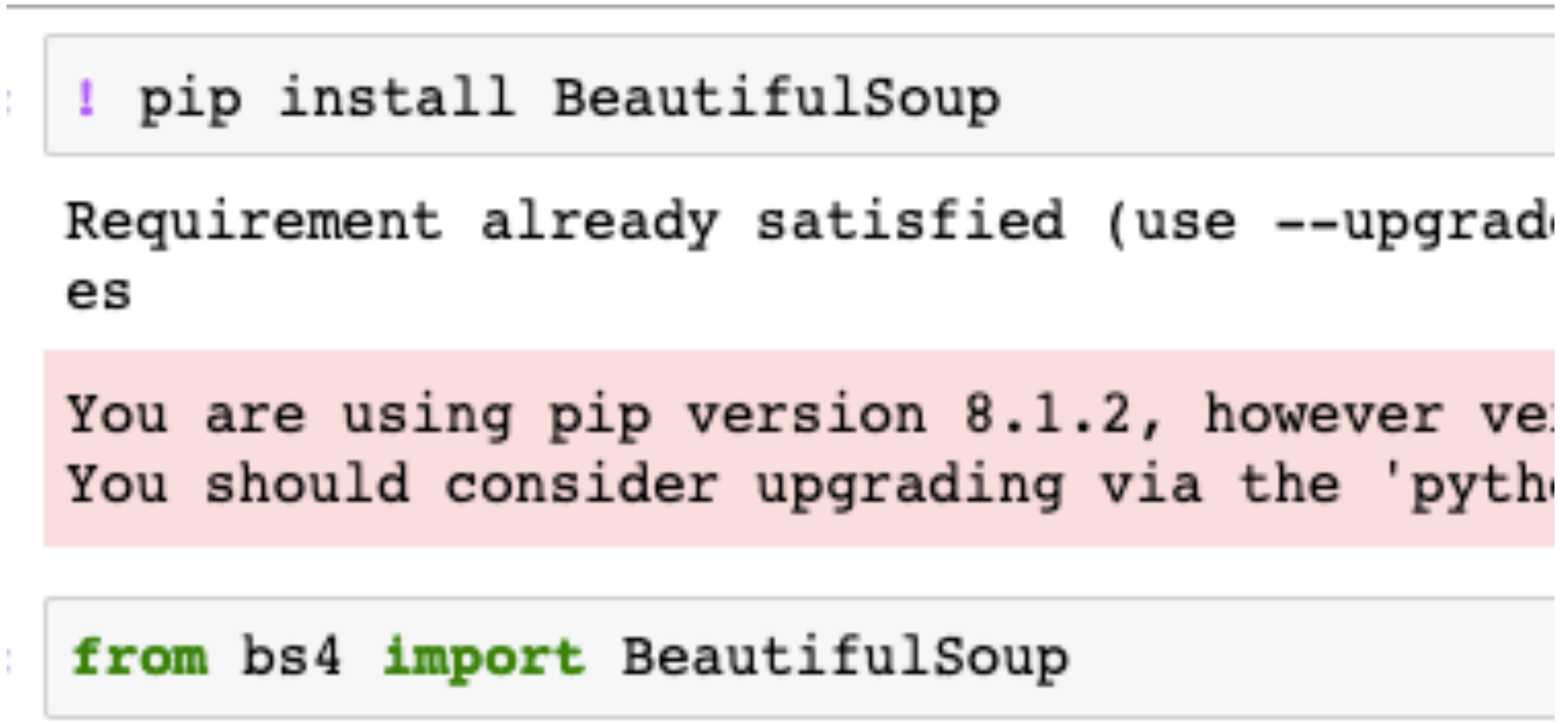
- Popular Use cases:
    - Ecommerce store automation
    - hydrological analysis
    - emergency resource allocation
    - oil and gas production intel

Four BeautifulSoup Object Types

- BeautifulSoup object
- Tag object
- NavigatableString object
- comment object

Working

- Install PIP
- Install PIP install BeautifulSoup
- Import BeautifulSoup library



```
! pip install BeautifulSoup
```

```
Requirement already satisfied (use --upgrad
es
```

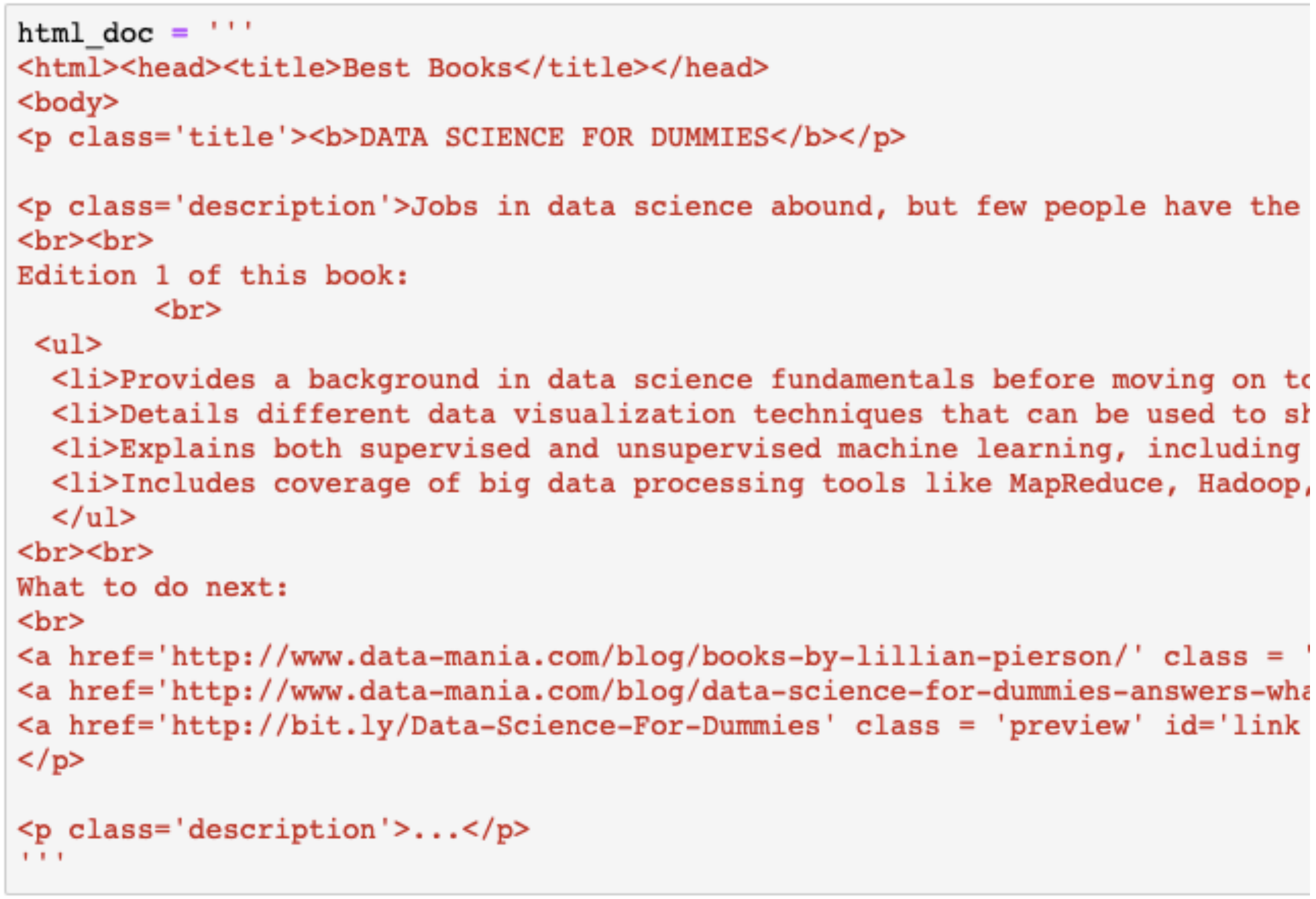
```
You are using pip version 8.1.2, however ve
You should consider upgrading via the 'pyth
```

```
from bs4 import BeautifulSoup
```

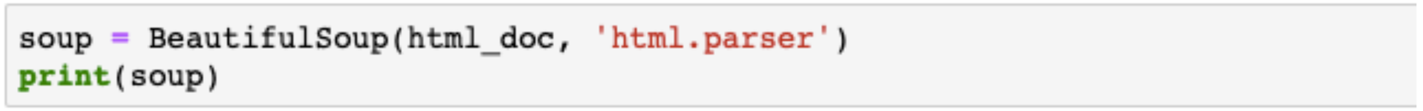
The BeautifulSoup Object

- Set an HTML document that we want to use as a markup (html_doc)

- Select a parser for the BeautifulSoup constructor.



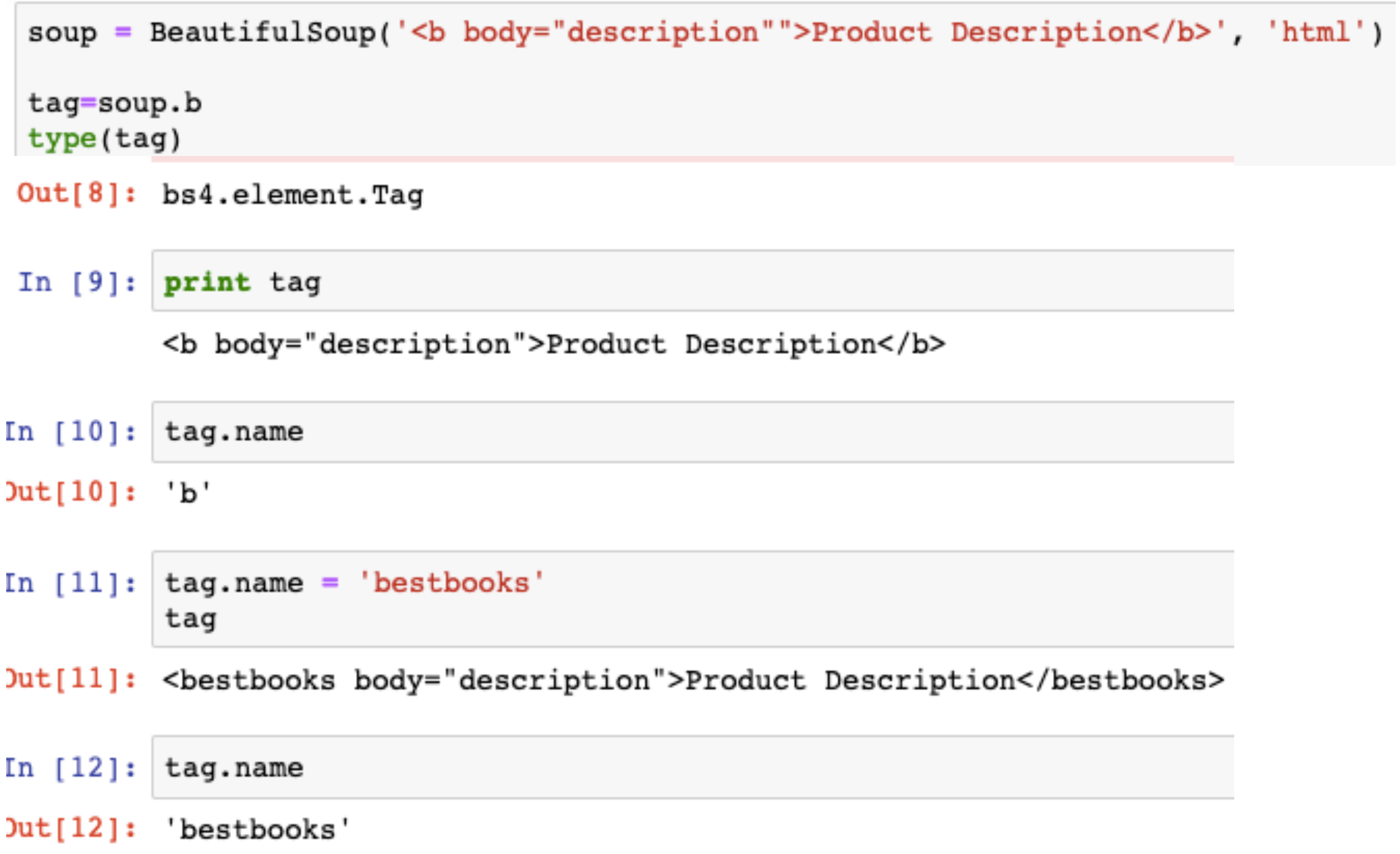
```
html_doc = '''
<html><head><title>Best Books</title></head>
<body>
<p class='title'><b>DATA SCIENCE FOR DUMMIES</b></p>

<p class='description'>Jobs in data science abound, but few people have the
<br><br>
Edition 1 of this book:
        <br>
 <ul>
  <li>Provides a background in data science fundamentals before moving on t
  <li>Details different data visualization techniques that can be used to s
  <li>Explains both supervised and unsupervised machine learning, including
  <li>Includes coverage of big data processing tools like MapReduce, Hadoop
  </ul>
<br><br>
What to do next:
<br>
<a href='http://www.data-mania.com/blog/books-by-lillian-pierson/' class = '
<a href='http://www.data-mania.com/blog/data-science-for-dummies-answers-wh
<a href='http://bit.ly/Data-Science-For-Dummies' class = 'preview' id='link
</p>

<p class='description'>...</p>
'''
```



```
soup = BeautifulSoup(html_doc, 'html.parser')
print(soup)
```

Tag Objects

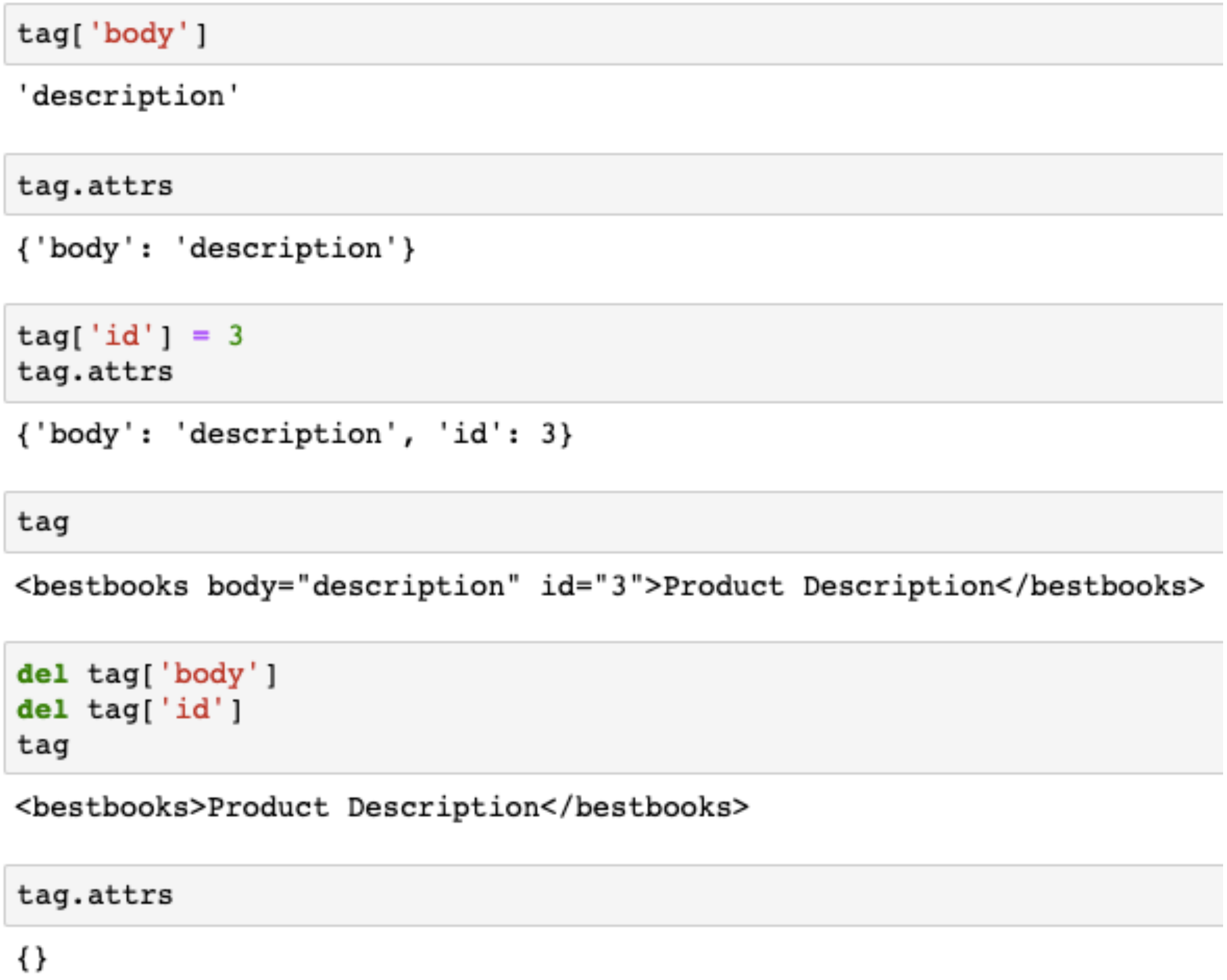
- A tag object represents HTML or XML elements that are present in the original markup document.
- Features:
  - Name
  - Attribute



```
soup = BeautifulSoup('<b body="description"">Product Description</b>', 'html')

tag=soup.b
type(tag)

Out[8]: bs4.element.Tag

In [9]: print tag

        <b body="description">Product Description</b>

In [10]: tag.name

Out[10]: 'b'

In [11]: tag.name = 'bestbooks'
         tag

Out[11]: <bestbooks body="description">Product Description</bestbooks>

In [12]: tag.name

Out[12]: 'bestbooks'
```

Tag Objects

- Working with attributes
- A tag can have any variety of attributes and can be accessed by treating the tag like a dictionary.



```
tag['body']
```

```
'description'
```

```
tag.attrs
```

```
{'body': 'description'}
```

```
tag['id'] = 3
tag.attrs
```

```
{'body': 'description', 'id': 3}
```

```
tag
```

```
<bestbooks body="description" id="3">Product Description</bestbooks>
```

```
del tag['body']
del tag['id']
tag
```

```
<bestbooks>Product Description</bestbooks>
```

```
tag.attrs
```

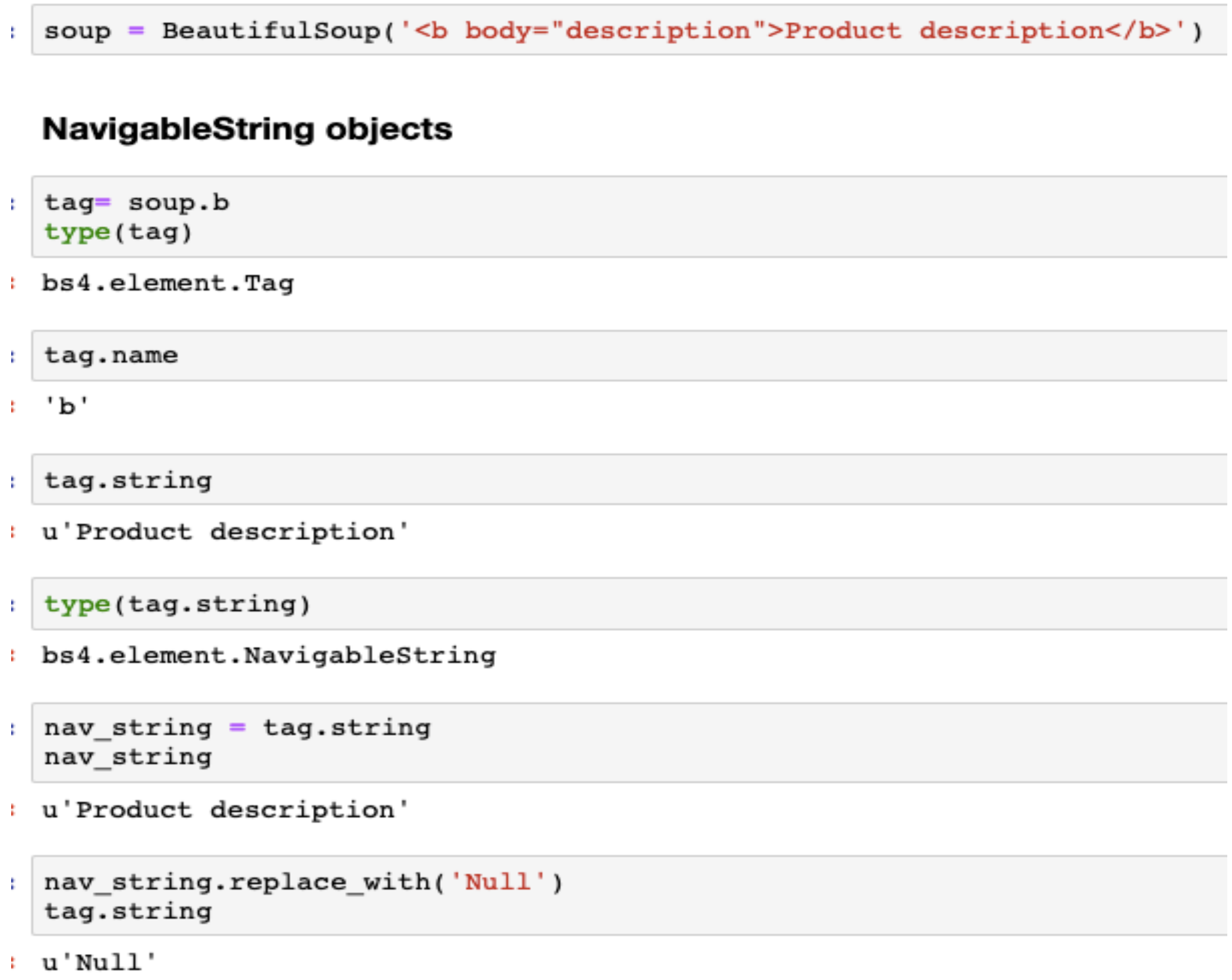
```
{}
```

Explore NavigatableString objects

- Import Beautiful Soup

- Create Beautiful Soup object

- Call the Beautiful Soup constructor and pass in a tag.



```
soup = BeautifulSoup('<b body="description">Product description</b>')
```

NavigableString objects

```
tag= soup.b
type(tag)
```

```
bs4.element.Tag
```

```
tag.name
```

```
'b'
```

```
tag.string
```

```
u'Product description'
```

```
type(tag.string)
```

```
bs4.element.NavigableString
```

```
nav_string = tag.string
nav_string
```

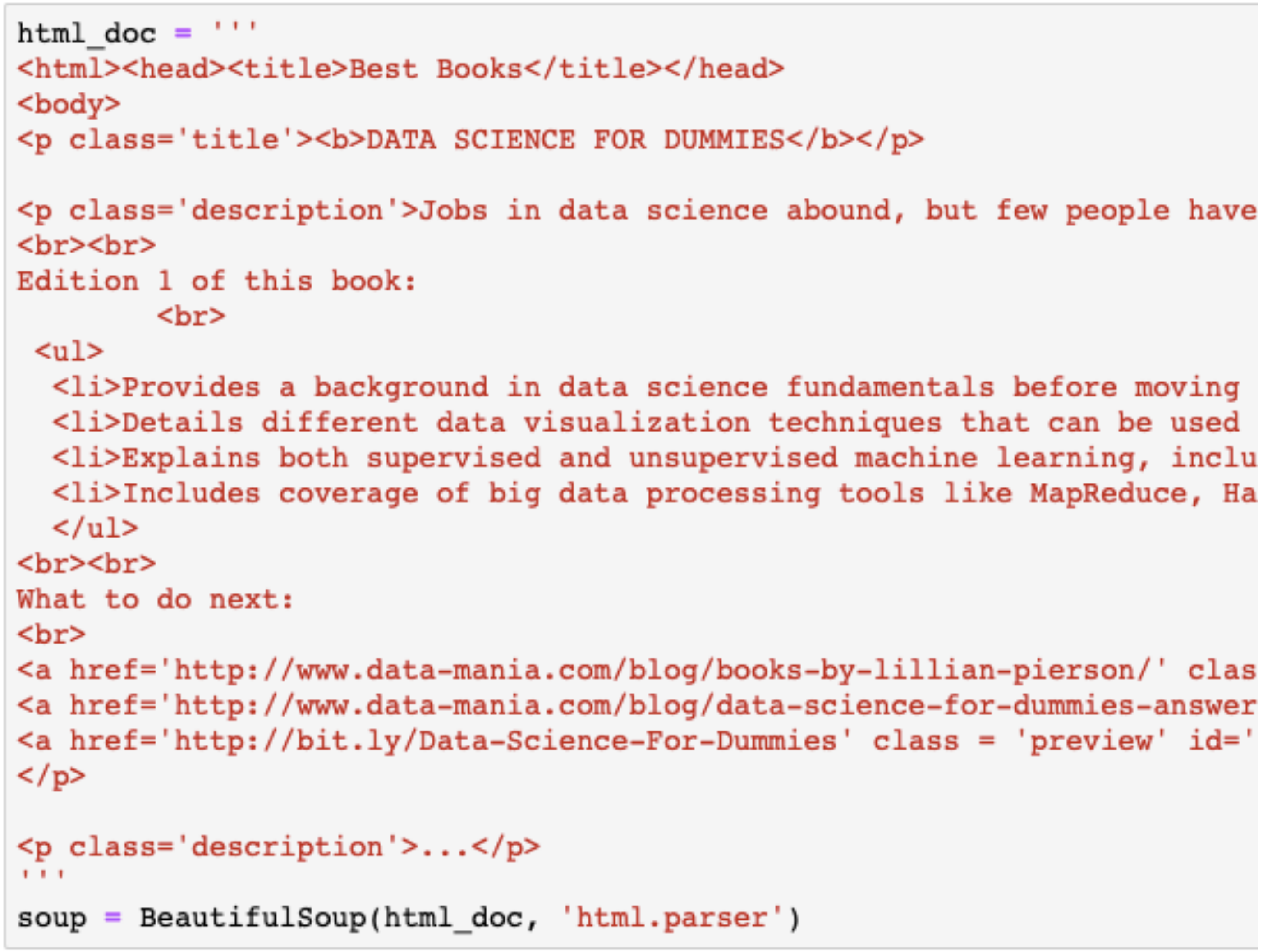
```
u'Product description'
```

```
nav_string.replace_with('Null')
tag.string
```

```
u'Null'
```

Working with NavigatableString

- Use the product description markup and convert that to a parse tree.



```
html_doc = '''
<html><head><title>Best Books</title></head>
<body>
<p class='title'><b>DATA SCIENCE FOR DUMMIES</b></p>

<p class='description'>Jobs in data science abound, but few people have
<br><br>
Edition 1 of this book:
        <br>
 <ul>
  <li>Provides a background in data science fundamentals before moving
  <li>Details different data visualization techniques that can be used
  <li>Explains both supervised and unsupervised machine learning, inclu
  <li>Includes coverage of big data processing tools like MapReduce, Ha
  </ul>
<br><br>
What to do next:
<br>
<a href='http://www.data-mania.com/blog/books-by-lillian-pierson/' clas
<a href='http://www.data-mania.com/blog/data-science-for-dummies-answer
<a href='http://bit.ly/Data-Science-For-Dummies' class = 'preview' id='
</p>

<p class='description'>...</p>
'''
soup = BeautifulSoup(html_doc, 'html.parser')
```

Parsed Data in Beautiful Soup

- Parsing Data:
  - An HTML or XML document is just passed to the BeautifulSoup() constructor.
  - The constructor converts the document to Unicode and then parses it with a built in HTML parser (by default)

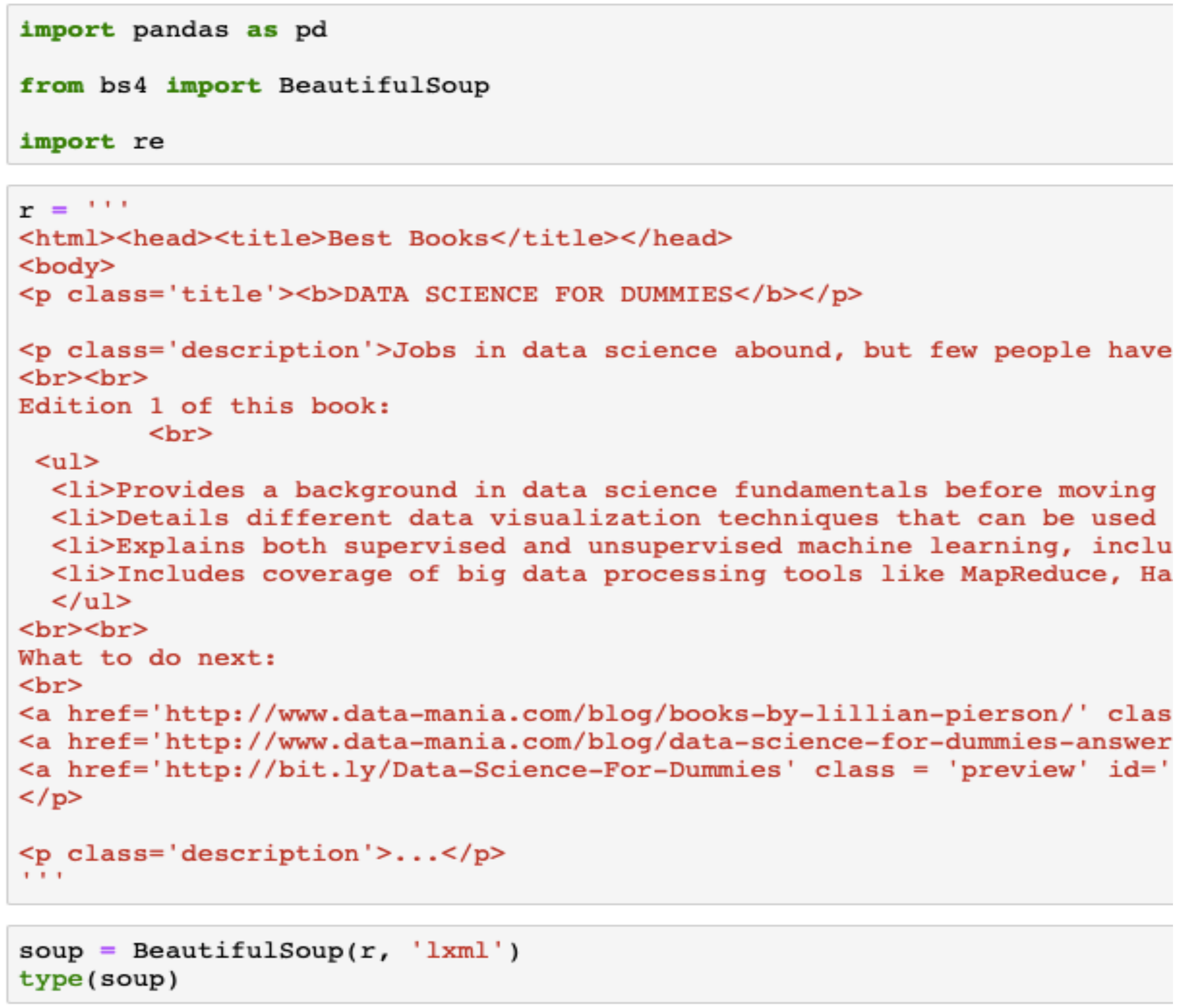
- Printing data that's in a parse tree.
- Searching and retrieving data from a parse tree

Methods for Searching and Filtering a Parse Tree

- Name argument
- Keyword argument
- String argument
- Lists
- Boolean values
- Strings
- Regular expression

Data Parsing

- Data parsing can be done by passing it in an HTML or XML document to the BeautifulSoup constructor.



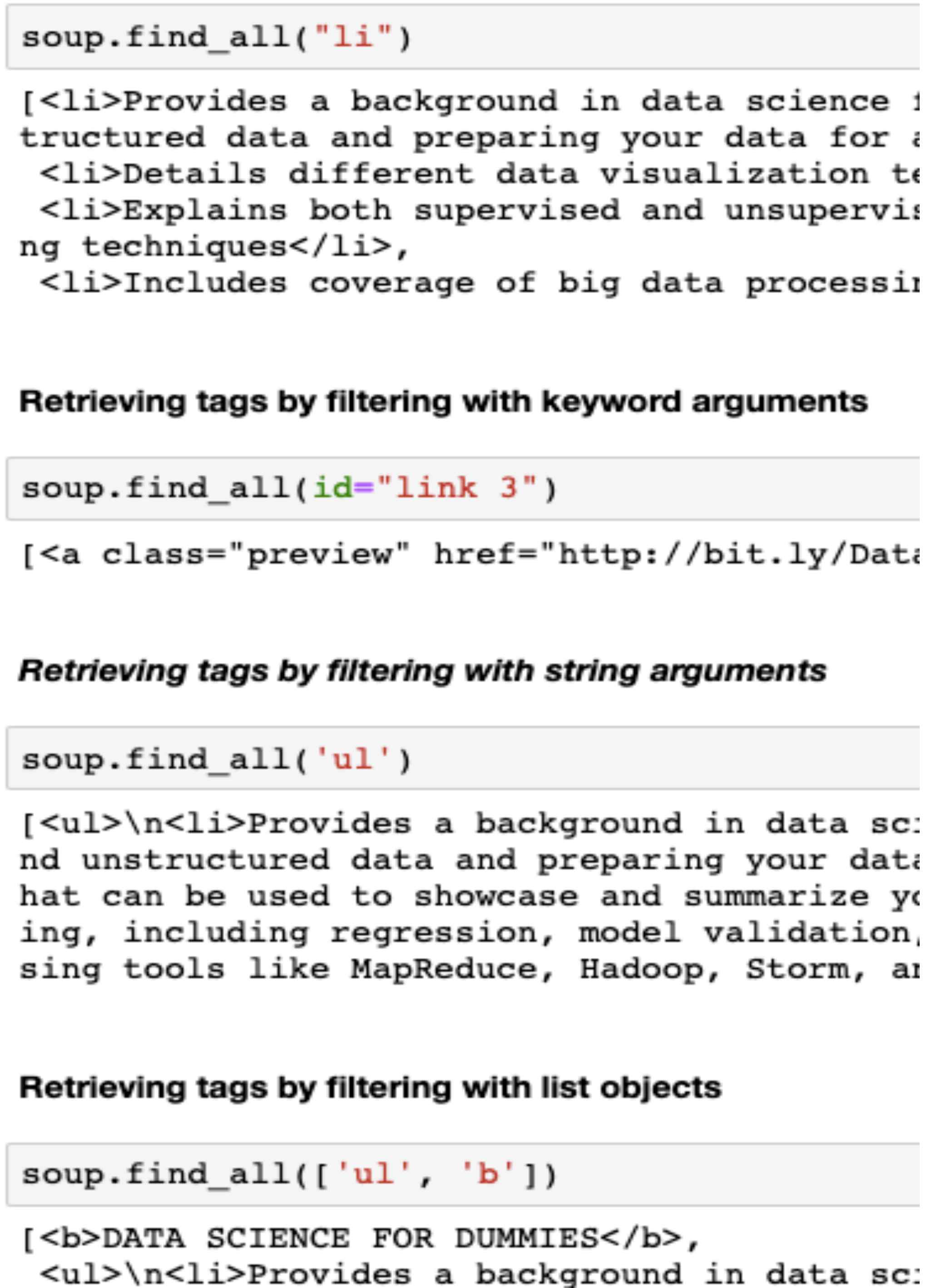
```
import pandas as pd

from bs4 import BeautifulSoup

import re
```

```
r = '''
<html><head><title>Best Books</title></head>
<body>
<p class='title'><b>DATA SCIENCE FOR DUMMIES</b></p>

<p class='description'>Jobs in data science abound, but few people have
<br><br>
Edition 1 of this book:
        <br>
 <ul>
  <li>Provides a background in data science fundamentals before moving
  <li>Details different data visualization techniques that can be used
  <li>Explains both supervised and unsupervised machine learning, inclu
  <li>Includes coverage of big data processing tools like MapReduce, Ha
  </ul>
<br><br>
What to do next:
<br>
<a href='http://www.data-mania.com/blog/books-by-lillian-pierson/' clas
<a href='http://www.data-mania.com/blog/data-science-for-dummies-answer
<a href='http://bit.ly/Data-Science-For-Dummies' class = 'preview' id='
</p>

<p class='description'>...</p>
'''
```

```
soup = BeautifulSoup(r, 'lxml')
type(soup)
```

Searching and retrieving data from a parse tree

- To return all the tags that contain HTML list items call the find_all method off as a soup object and then pass in the name of the tag, li.



```
soup.find_all("li")
```

```
[<li>Provides a background in data science f
tructured data and preparing your data for a
 <li>Details different data visualization te
 <li>Explains both supervised and unsupervis
ng techniques</li>,
 <li>Includes coverage of big data processir
```

Retrieving tags by filtering with keyword arguments

```
soup.find_all(id="link 3")
```

```
[<a class="preview" href="http://bit.ly/Data
```

Retrieving tags by filtering with string arguments

```
soup.find_all('ul')
```

```
[<ul>\n<li>Provides a background in data sci
nd unstructured data and preparing your data
hat can be used to showcase and summarize yo
ing, including regression, model validation,
sing tools like MapReduce, Hadoop, Storm, ar
```

Retrieving tags by filtering with list objects

```
soup.find_all(['ul', 'b'])
```

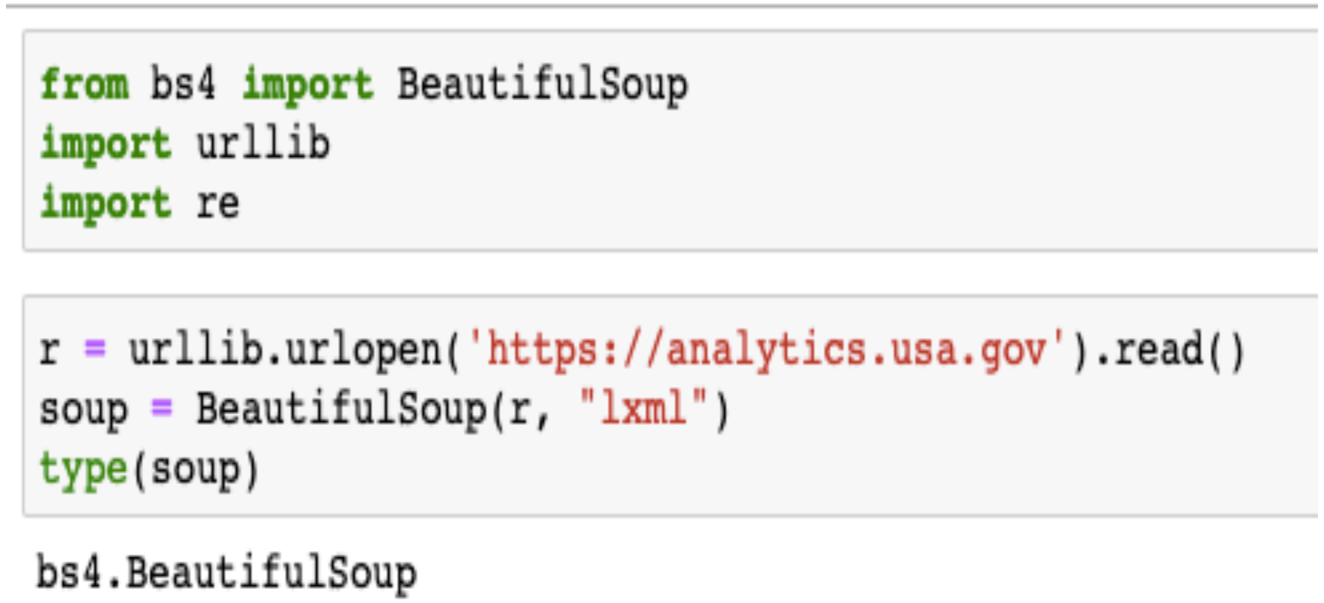
```
[<b>DATA SCIENCE FOR DUMMIES</b>,
 <ul>\n<li>Provides a background in data sci
```

Demonstrating Web Scraping

- 1. Scraping a web page
- 2. Saving web scraping results

Web Scraping

- Import Beautiful Soup library.
- Import ‘urlib’ library in order to read in the data from the internet
- Import ‘re’ which is the regular expression library



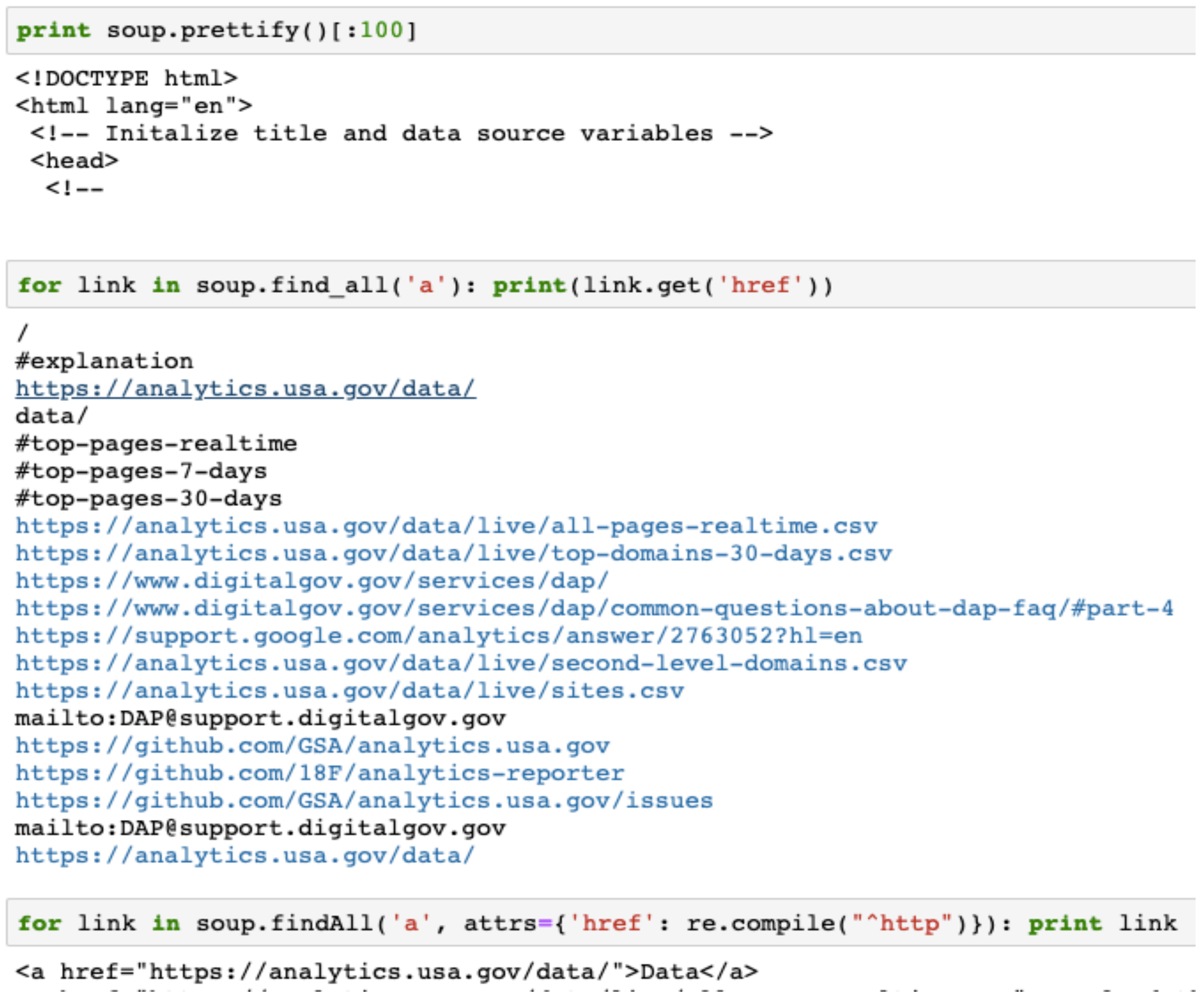
```
from bs4 import BeautifulSoup
import urllib
import re
```

```
r = urllib.urlopen('https://analytics.usa.gov').read()
soup = BeautifulSoup(r, "lxml")
type(soup)
```

```
bs4.BeautifulSoup
```

Web Scraping

- Using 'prettify' function to add structure and make it a bit easier to read.
- The function is : soup.prettify()



```
print soup.prettify()[:100]
```

```
<!DOCTYPE html>
<html lang="en">
 <!-- Initalize title and data source variables -->
 <head>
  <!--
```

```
for link in soup.find_all('a'): print(link.get('href'))
```

```
/
#explanation
https://analytics.usa.gov/data/
data/
#top-pages-realtime
#top-pages-7-days
#top-pages-30-days
https://analytics.usa.gov/data/live/all-pages-realtime.csv
https://analytics.usa.gov/data/live/top-domains-30-days.csv
https://www.digitalgov.gov/services/dap/
https://www.digitalgov.gov/services/dap/common-questions-about-dap-faq/#part-4
https://support.google.com/analytics/answer/2763052?hl=en
https://analytics.usa.gov/data/live/second-level-domains.csv
https://analytics.usa.gov/data/live/sites.csv
mailto:DAP@support.digitalgov.gov
https://github.com/GSA/analytics.usa.gov
https://github.com/18F/analytics-reporter
https://github.com/GSA/analytics.usa.gov/issues
mailto:DAP@support.digitalgov.gov
https://analytics.usa.gov/data/
```

```
for link in soup.findAll('a', attrs={'href': re.compile("^http")}): print link
```

```
<a href="https://analytics.usa.gov/data/">Data</a>
```