web scraping with BeautifulSoup

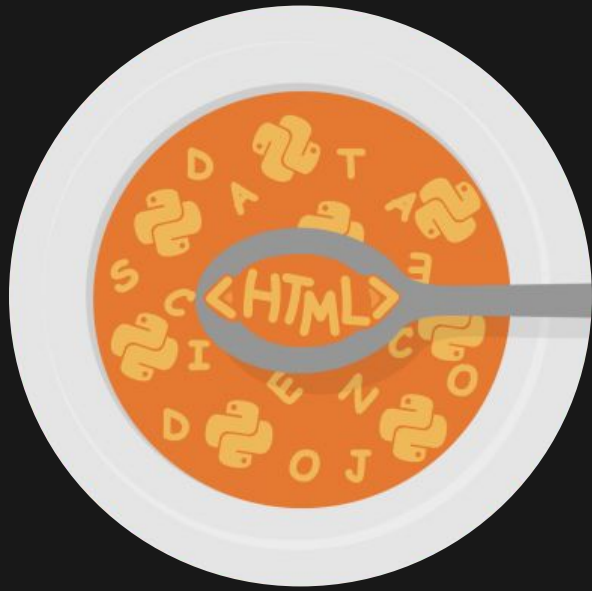


image from Data Science Dojo Blog

what is web scraping?

the computer software technique of extracting information from websites

unstructured data → structured data

HTML → database/spreadsheet

why web scrape?

soooooo much information !!

- over 1.8 billion websites
- 8,000 tweets per second
- 800 instagram posts per second
- 1,300 tumblr posts per second

why web scrape?

- price tracking
- market research
- keeping track of business competitors
- centralizing information
- keeping accountability
- personal research
- ...

collecting information from websites

- Application Program Interfaces (APIs)
- publicly downloadable files
- web scraping

what is `BeautifulSoup`?

Python library for pulling data out of HTML and XML files

image from ScrapingAuthority.com

what is `BeautifulSoup`?

parses and breaks down a HTML page into a tree of objects

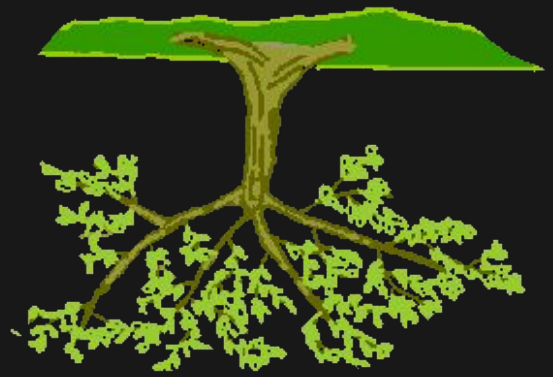


image from University of Alberta Computing Science

HTML tree structure

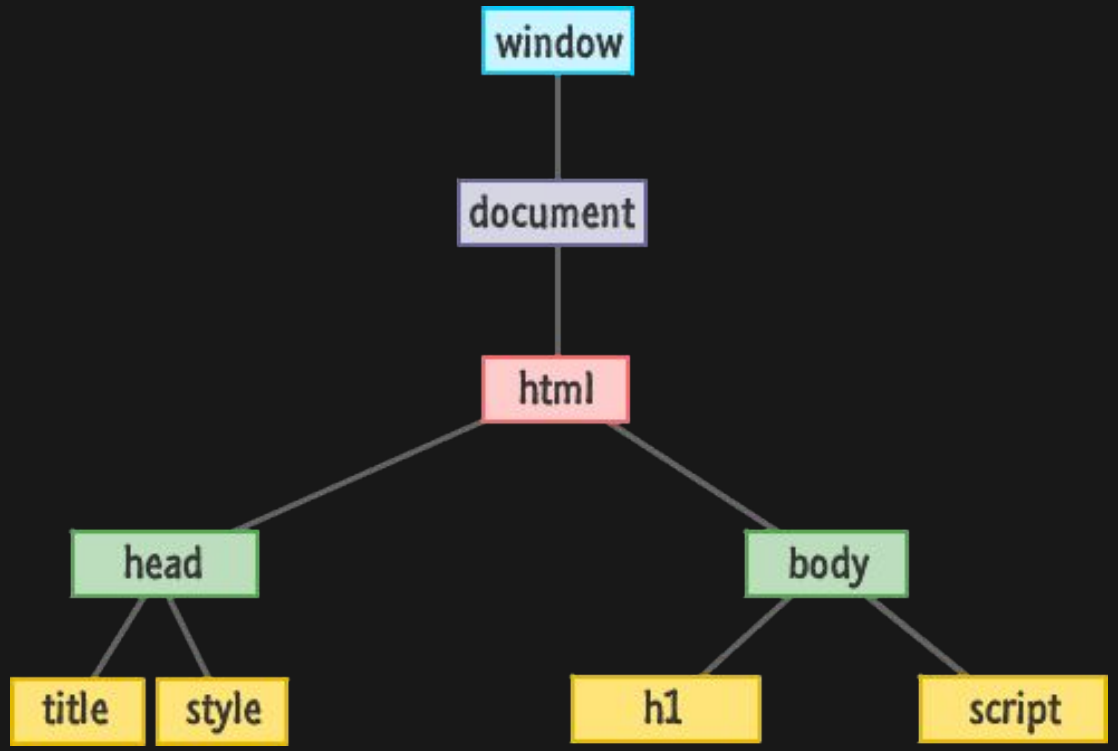


image from Kirupa.com

`BeautifulSoup` objects

- Tag
- NavigableString
- BeautifulSoup
- Comment

tag

- corresponds to HTML tag
- has a name
- has a dictionary of attributes

tag

```
soup = BeautifulSoup('<b id="boldest">Extremely bold</b>')
tag = soup.b
```

```
tag.name
# u'b'
tag['id']
# u'boldest'
tag.attrs
# {u'id': 'boldest'}
```

NavigableString

- a Python (Unicode) string, but with searching functionality

BeautifulSoup

- the entire document!

Comment

- NavigableString but with special formatting

navigating the tree
&
searching the tree

navigating the tree

&

searching the tree

navigating the tree

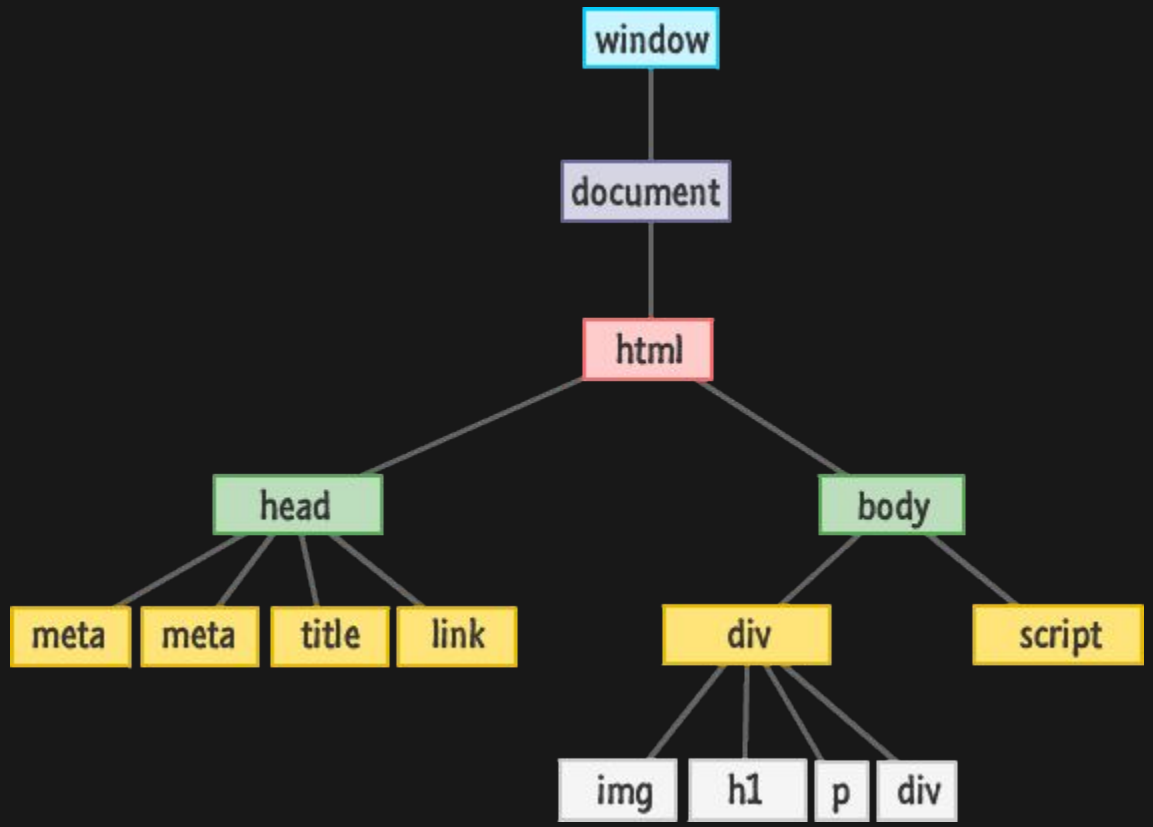
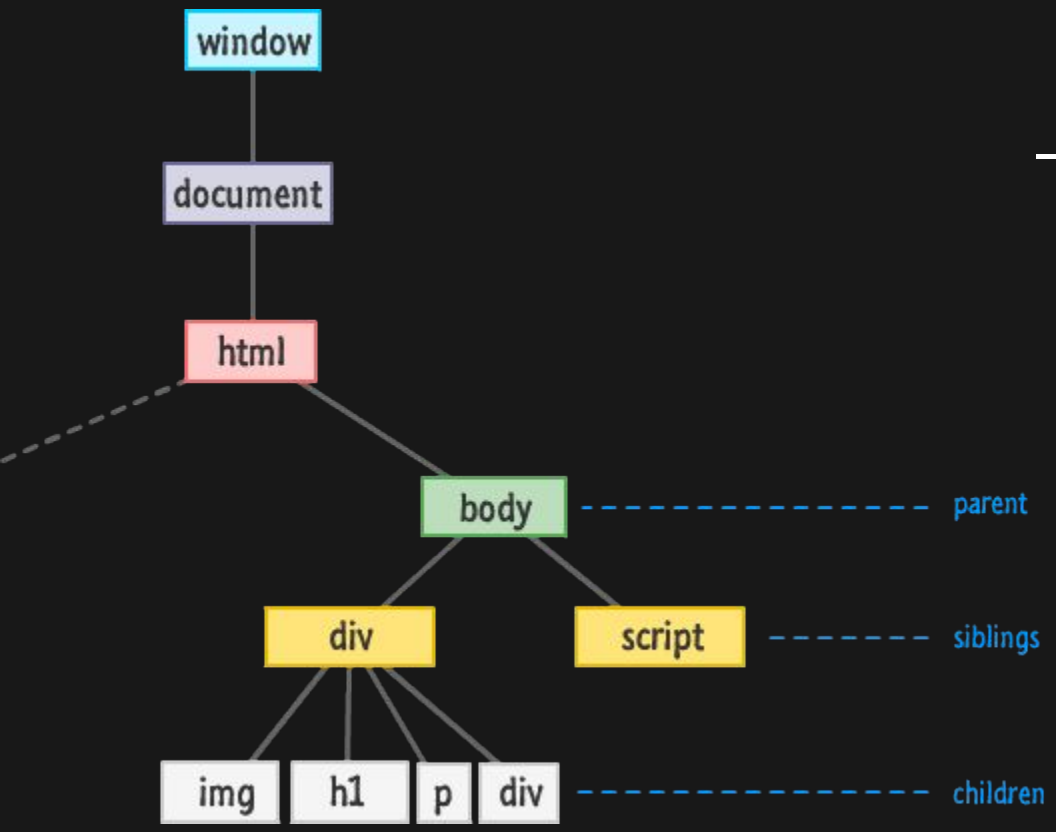


image from Kirupa.com

navigating the tree



- `BeautifulSoup` provides functions to move along this tree

image from Kirupa.com

navigating the tree

```
<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; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
            <a class="sister" href="http://example.com/tillie" id="link2">Tillie</a>
            ; and they lived at the bottom of a well.
        </p>
        <p class="story">
            ...
        </p>
    </body>
</html>
```

navigating the tree

```
<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; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
            <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
            ; and th
        </p>
        <p class="st
            ...
        </p>
    </body>
</html>
```

using tags

```
soup.head
# <head><title>The Dormouse's story</title></head>
soup.body.b
# <b>The Dormouse's story</b>
```

navigating the tree

```
<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; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class=
            <a class=
            ; and th
        </p>
        <p class="st
            ...
        </p>
    </body>
</html>
```

using contents

```
head_tag = soup.head
head_tag
# <head><title>The Dormouse's story</title></head>
head_tag.contents
# [<title>The Dormouse's story</title>]
```

navigating the tree

```
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
        <p class="title">
            <b>The Dormouse's story</b>
        </p>
        <p class="st
            Once upo
            <a class
            <a class
            <a class
            ; and th
        </p>
        <p class="st
            ...
        </p>
    </body>
</html>
```

using contents

```
head_tag.contents
# [<title>The Dormouse's story</title>]
title_tag = head_tag.contents[0]
title_tag
# <title>The Dormouse's story</title>
title_tag.contents
# [u'The Dormouse's story']
```

navigating the tree

```
<html>
    <head>
        <title>The Dormouse's story</title>
    </head>
    <body>
        <p class="title">
            <b>The Dormouse's story</b>
        </p>
        <p class="story">
            Once upo
            <a class
            <a class
            <a class
            ; and th
        </p>
        <p class="st
            ...
        </p>
    </body>
</html>
```

using descendants

```
head_tag
# <head><title>The Dormouse's story</title></head>
for child in head_tag.descendants:
    print(child)
# <title>The Dormouse's story</title>
# The Dormouse's story
```

navigating the tree

```
<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; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
            <a class="sister" href="ht
            ; and they lived at the bo
        </p>
        <p class="story">
            ...
        </p>
    </body>
</html>
```

using string

```
title_tag
# <title>The Dormouse's story</title>
title_tag.string
# u'The Dormouse's story'
```

navigating the tree

`.contents` and `.children`

`.parent` and `.parents`

navigating the tree

`.next_sibling` & `.next_siblings`

`.next_element` & `.next_elements`

navigating the tree

`.previous_sibling` & `.previous_siblings`

`.previous_element` & `.previous_elements`

navigating the tree

&

searching the tree

searching the tree

```
<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; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
            <a clas
            ; and t
        </p>
        <p class="s
            ...
        </p>
    </body>
</html>
```

tags & find_all

```
soup.find_all("title")
# [<title>The Dormouse's story</title>]
soup.find_all("p", "title")
# [<p class="title"><b>The Dormouse's story</b></p>]
```

searching the tree

```
soup.find_all("a")
# [<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie"
id="link2">Lacie</a>,
#  <a class="sister"
href="http://example.com/tillie"
id="link3">Tillie</a>]
```

```
<html>
    <head>
        <title>The D
    </head>
    <body>
        <p class="ti
            <b>The D
        </p>
        <p class="story">
            Once upon a time there were three little sisters; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
            <a class="sister" href="http://example.com/tillie" id="link2">Tillie</a>
            ; and they lived at the bottom of a well.
        </p>
        <p class="story">
```

searching the tree

```
soup.find_all(["a", "b"])
# [<b>The Dormouse's story</b>,
# [<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie"
id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie"]
id="link3">Tillie</a>]
```

```
<html>
    <head>
        <title>The
    </head>
    <body>
        <p class="
            <b>The
        </p>
        <p class="
            Once upon a time there were three little sisters; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
            <a class="sister" href="http://example.com/tillie" id="link2">Tillie</a>
            ; and they lived at the bottom of a well.
        </p>
        <p class="story">
```

searching the tree

```
<html>
    <head>
        <title>The D
    </head>
    <body>
        <p class="ti
            <b>The D
        </p>
        <p class="story">
            Once upon a time there were three little sisters; and their names were
            <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
            <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
            <a class="sister" href="http://example.com/tillie" id="link2">Tillie</a>
            ; and they lived at the bottom of a well.
        </p>
        <p class="story">
            ...
        </p>
    </body>
</html>
```

```
soup.find(id="link1")
# [<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>]
```

using attributes

custom search by function

```
def has_class_but_no_id(tag):
    return tag.has_attr('class') and not
tag.has_attr('id')

soup.find_all(has_class_but_no_id)
# [<p class="title"><b>The Dormouse's story</b></p>,
#  <p class="story">Once upon a time there were...</p>,
#  <p class="story">...</p>]
```

search by attributes

```
soup.find_all("a", class_="sister")
# [<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie"
id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie"
id="link3">Tillie</a>]
```

search by string

```
soup.find_all(string="Elsie")
# [u'Elsie']
```

limit search quantity

```
soup.find_all("a", limit=2)
# [ <a class="sister" href="http://example.com/lacie"
id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie"
id="link3">Tillie</a>]
```

searching the tree

```
.find_parent()

.find_next_sibling()

.find_previous_sibling()

.find_next()
```

search using `.select()`

```
soup.select("title")
# [<title>The Dormouse's story</title>]
soup.select("p:nth-of-type(3)")
# [<p class="story">...</p>]
soup.select("html head title")
# [<title>The Dormouse's story</title>
```

more with the tree

- modifying the tree
- make the soup ~beautiful~

what do I do with all this information?

- output your findings somewhere!
- write to a CSV
- write to a database
- create your own website
- data analysis

general scraping limitations

- badly formatted HTML code
- authentication systems
- session based systems (cookies)
- blocking bulk access
- legal barriers (copyright, database rights)

...also, web scraping sometimes walks a fuzzy ethical line...

A government site containing public data offers no API to access this data. Nothing on the site indicates that the agency operating it has a problem with scraping.

A private site containing private data that was assembled through the hard work of its owners offers an API to access its data, but such access requires payment, and you don't wish to pay.

what is the ethical line for scraping data?
is there an ethical line?

experts recommend...

- use APIs when possible
- respect Terms & Conditions
- respect robots.txt
- be gentle, avoid distributed denial of service (DDoS)
- identify yourself
- only save data you need
- respect the content you keep
- create new value from information

you try!

https://tinyurl.com/scrape50

1. scrape alice.html using scrape_alice.py
2. play with miniwiki.py
3. scrape https://www.thecrimson.com/