

# [301] Web 3

Tyler Caraza-Harter

# Learning Objectives Today

Use BeautifulSoup modules

- prettify, find\_all, find, get\_text

Learn about scraping

- Document Object Model
- extracting links
- robots.txt

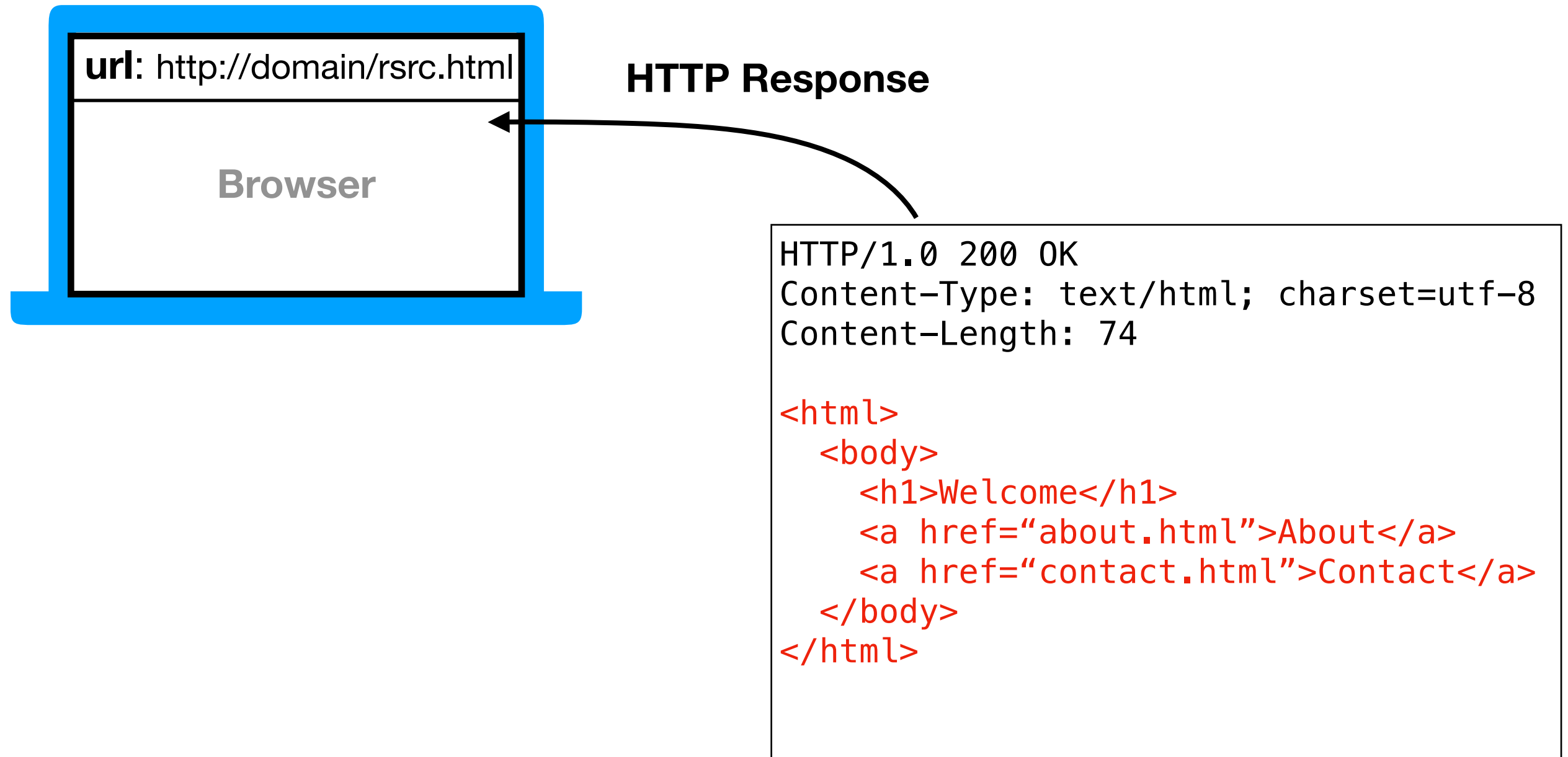
# Outline

Document Object Model

BeautifulSoup module

Scraping States from Wikipedia

# What does a web browser do when it gets some HTML in an HTTP response?



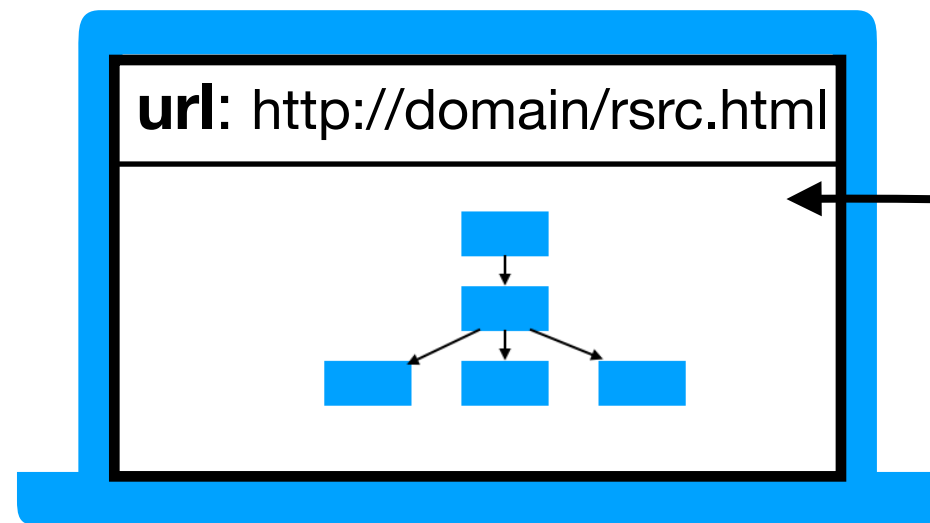
**url:** http://domain/rsrc.html

```
<html>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```

## HTTP Response

HTTP/1.0 200 OK  
Content-Type: text/html; charset=utf-8  
Content-Length: 74

```
<html>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```



## HTTP Response

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
```

```
<html>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```

before displaying a page, the  
browser uses HTML to generate  
a Document Object Model  
(DOM Tree)

url: http://domain/rsrc.html

HTTP Response

HTTP/1.0 200 OK  
Content-Type: text/html; charset=utf-8  
Content-Length: 74

```
<html>  
  <body>  
    <h1>Welcome</h1>  
    <a href="about.html">About</a>  
    <a href="contact.html">Contact</a>  
  </body>  
</html>
```

html

body

h1

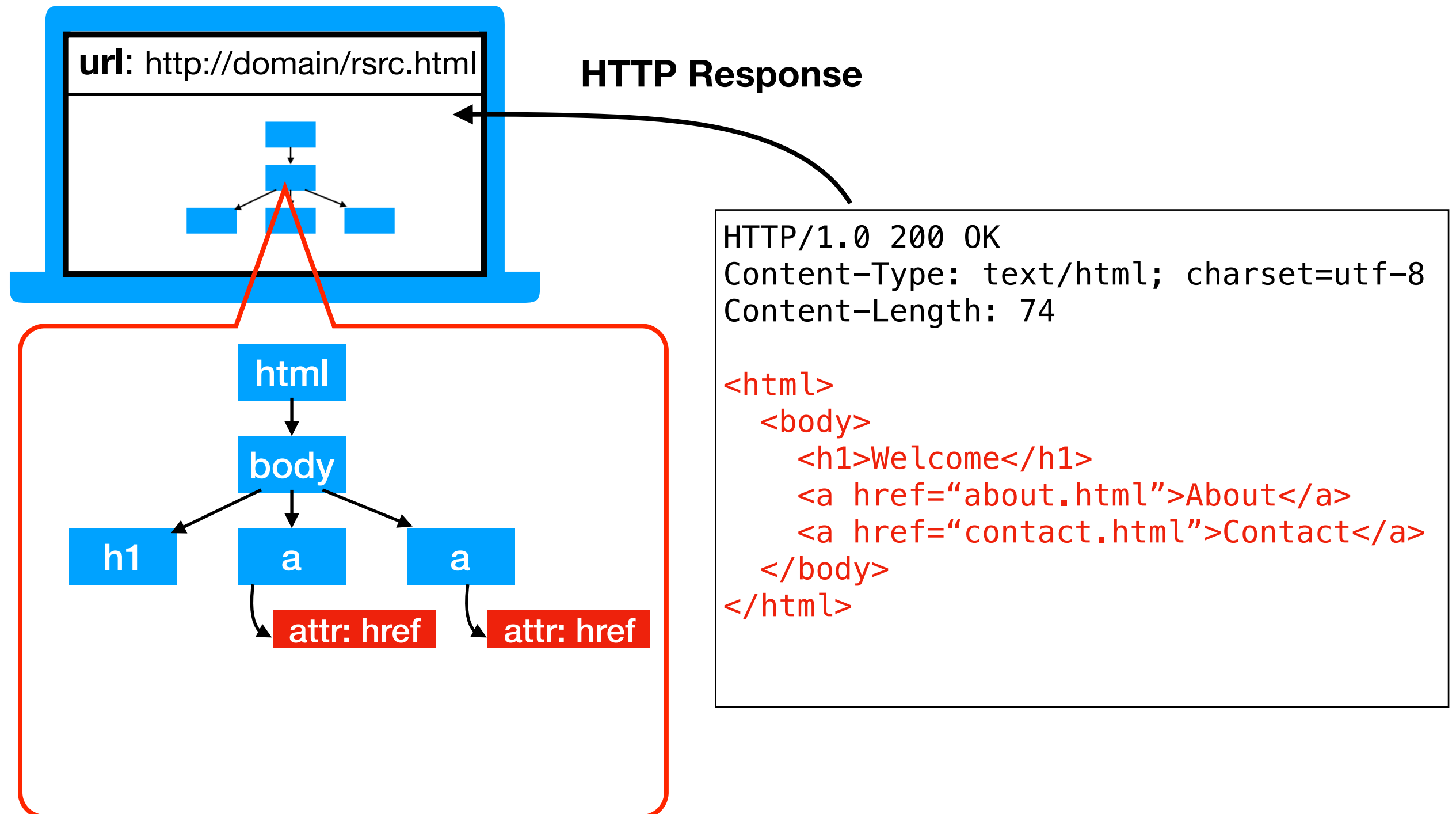
a

a

vocab: elements

# Elements may contain

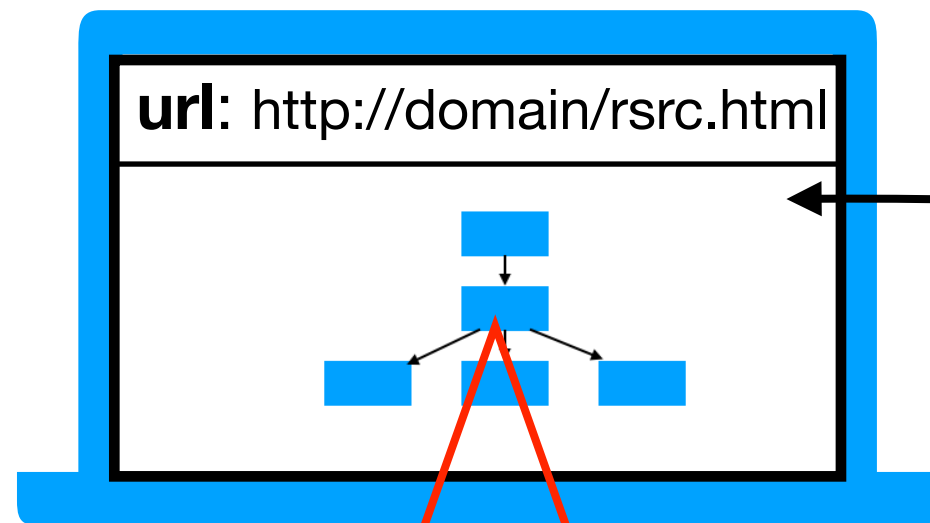
- attributes





## Elements may contain

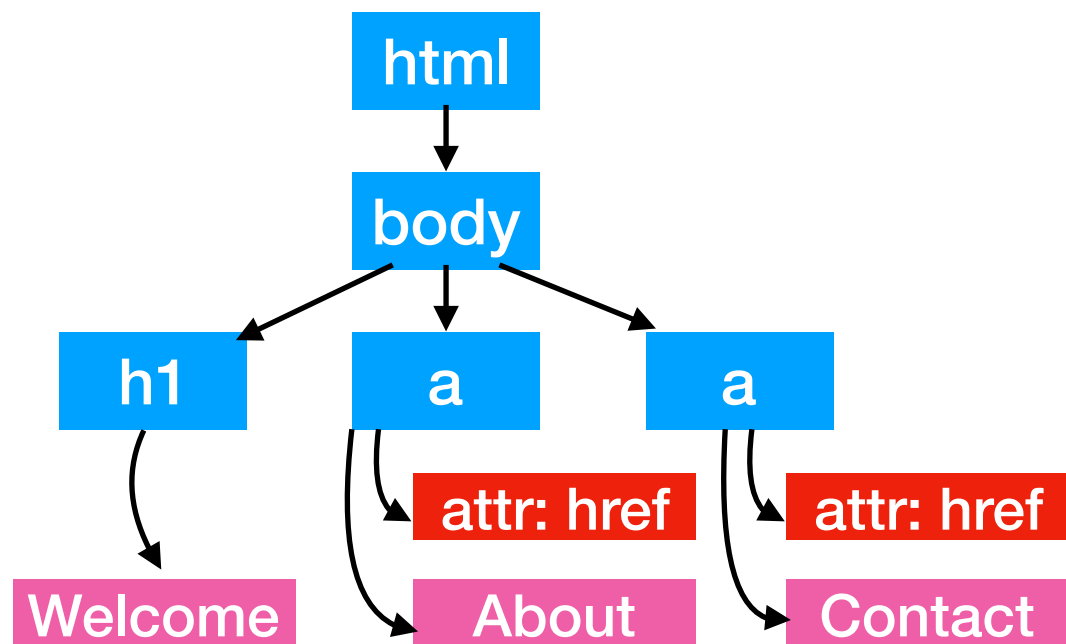
- attributes
- text



### HTTP Response

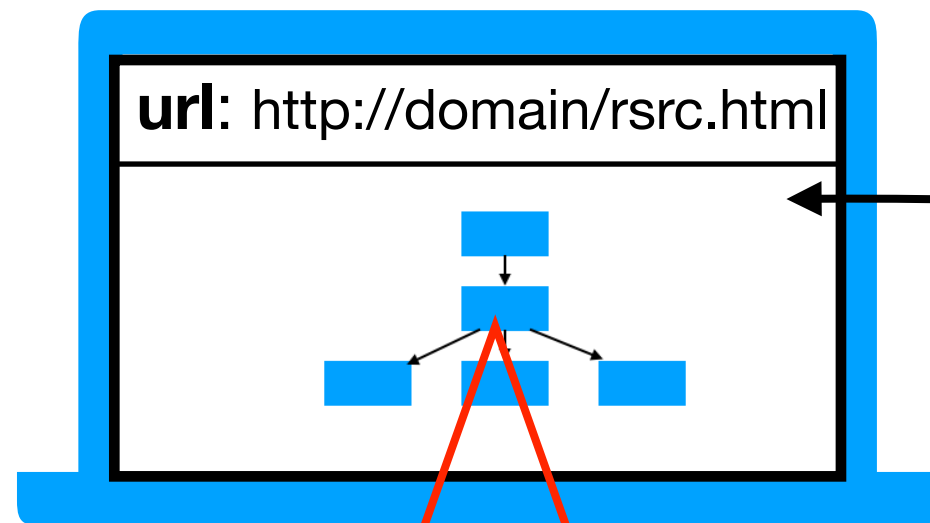
HTTP/1.0 200 OK  
Content-Type: text/html; charset=utf-8  
Content-Length: 74

```
<html>  
  <body>  
    <h1>Welcome</h1>  
    <a href="about.html">About</a>  
    <a href="contact.html">Contact</a>  
  </body>  
</html>
```



## Elements may contain

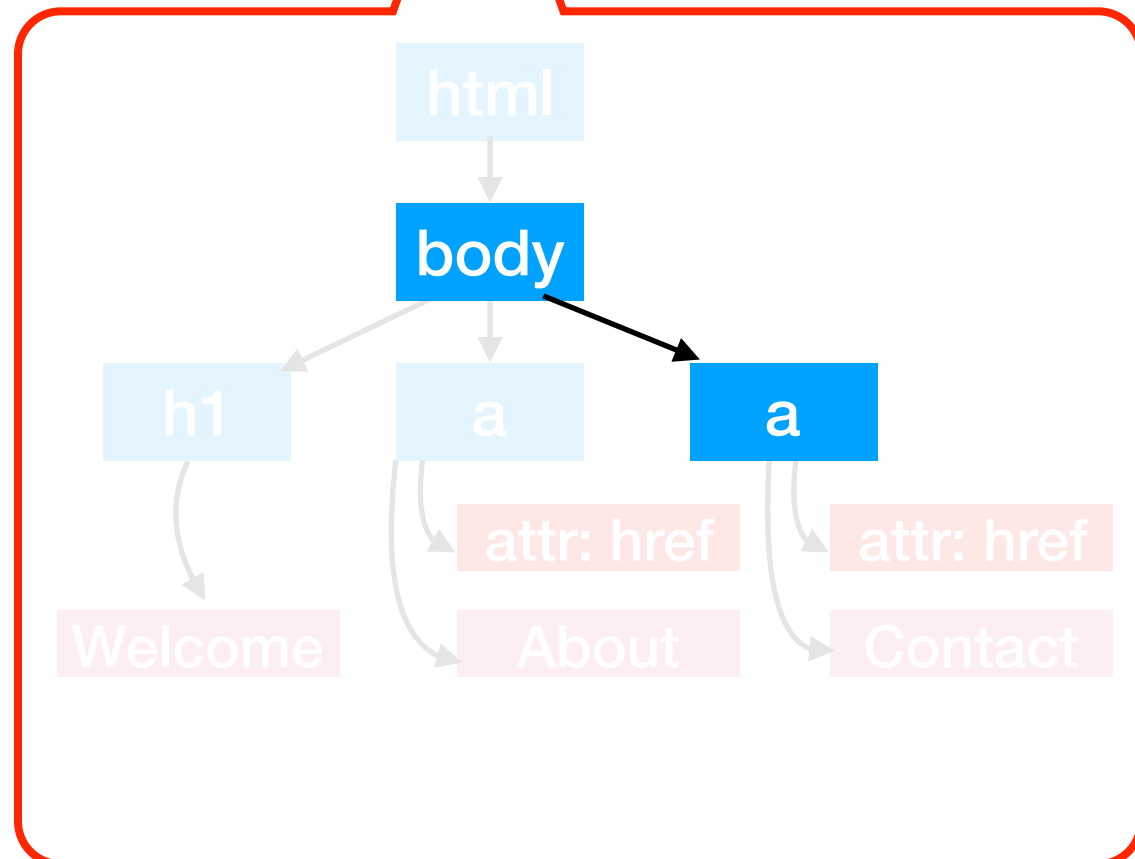
- attributes
- text
- other elements



### HTTP Response

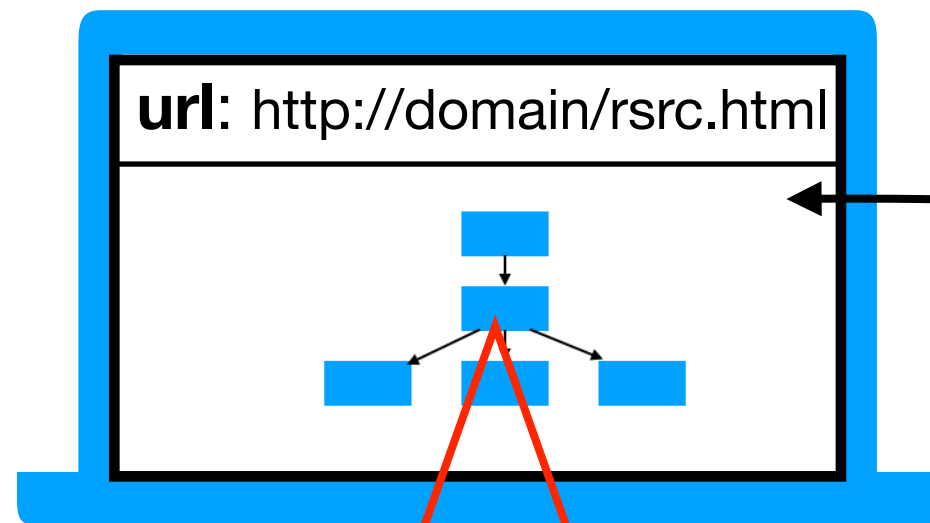
HTTP/1.0 200 OK  
Content-Type: text/html; charset=utf-8  
Content-Length: 74

```
<html>  
  <body>  
    <h1>Welcome</h1>  
    <a href="about.html">About</a>  
    <a href="contact.html">Contact</a>  
  </body>  
</html>
```



## Elements may contain

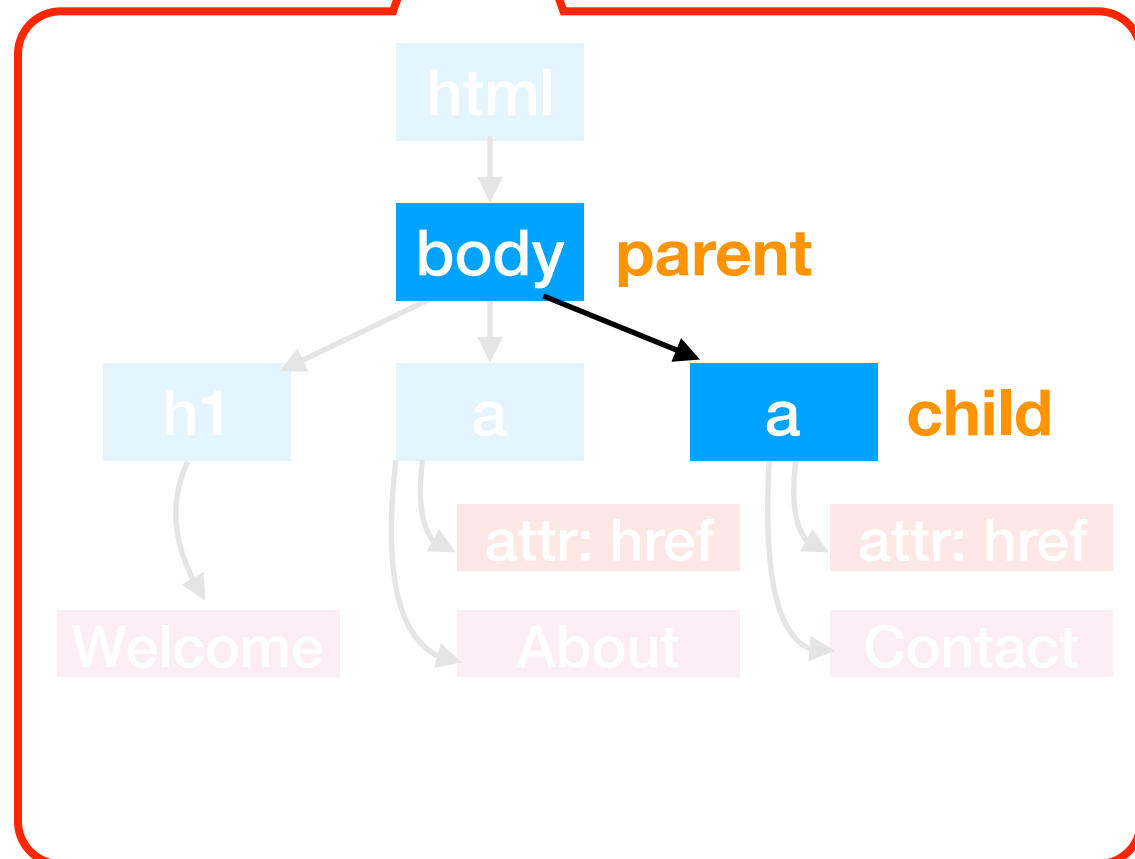
- attributes
- text
- other elements



HTTP Response

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
```

```
<html>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```



## Elements may contain

- attributes
- text
- other elements



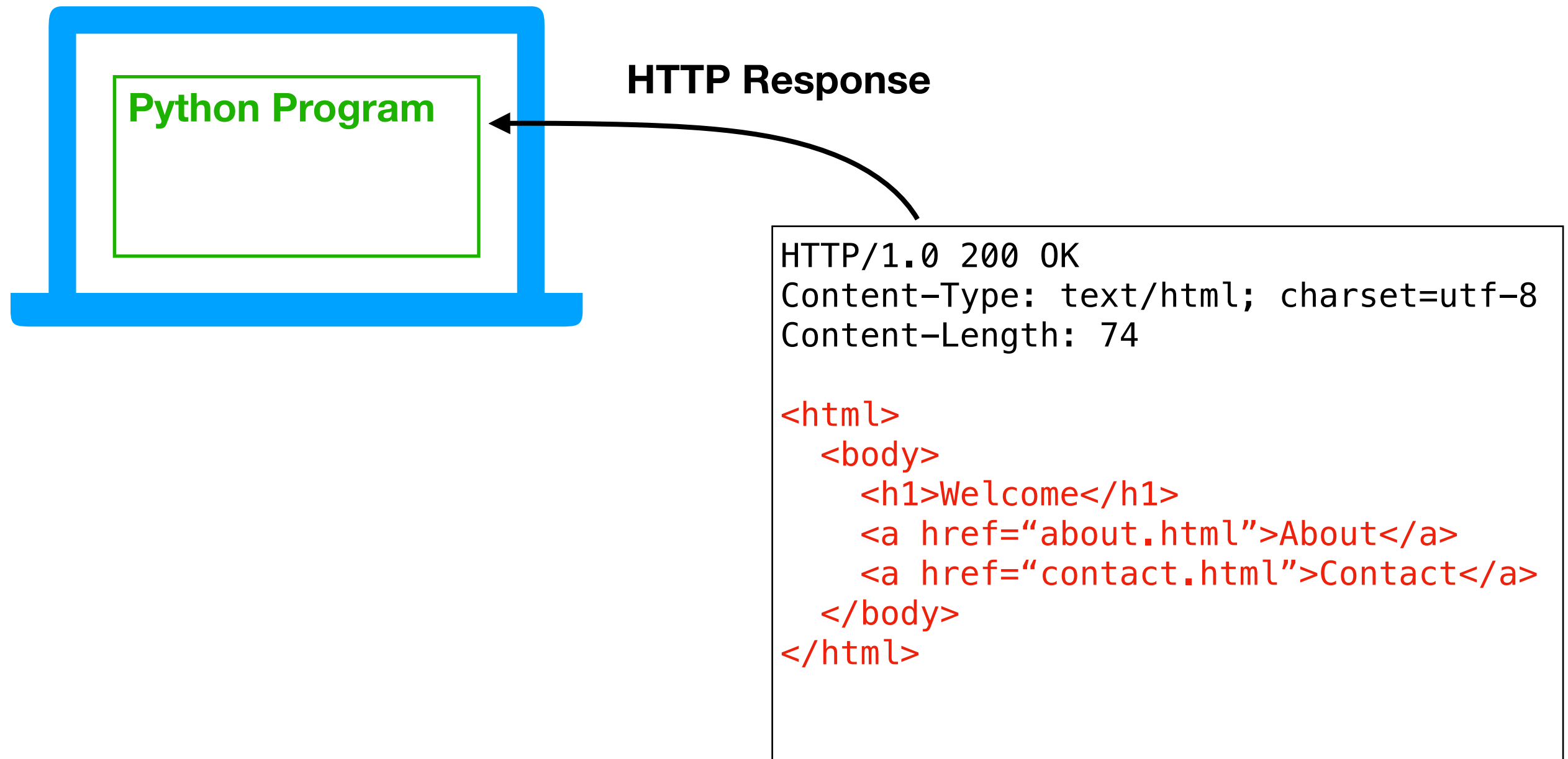
HTTP Response

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
```

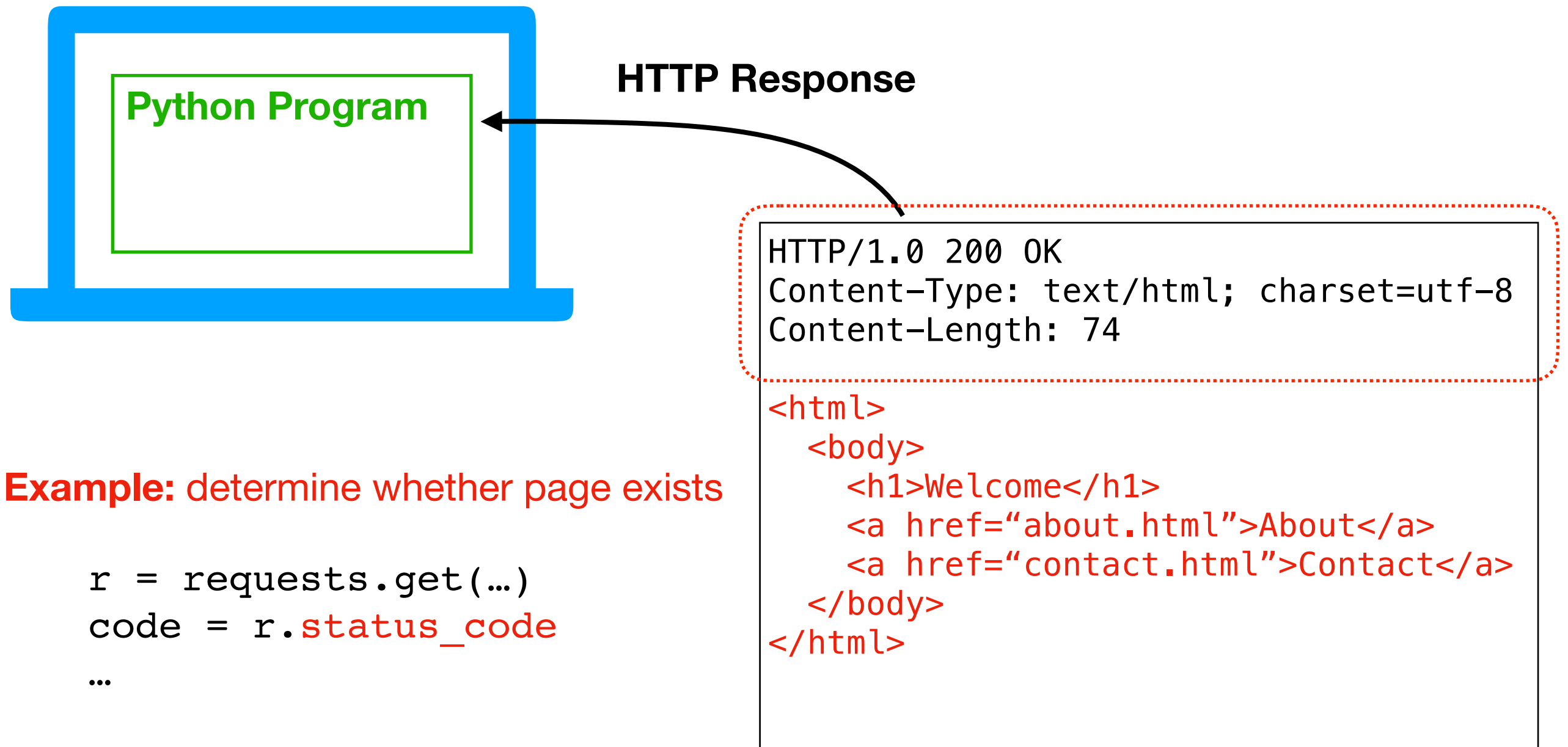
```
<html>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```

browser renders (displays)  
the DOM tree

Python program gets back the same info  
as a web browser (HTTP and HTML)

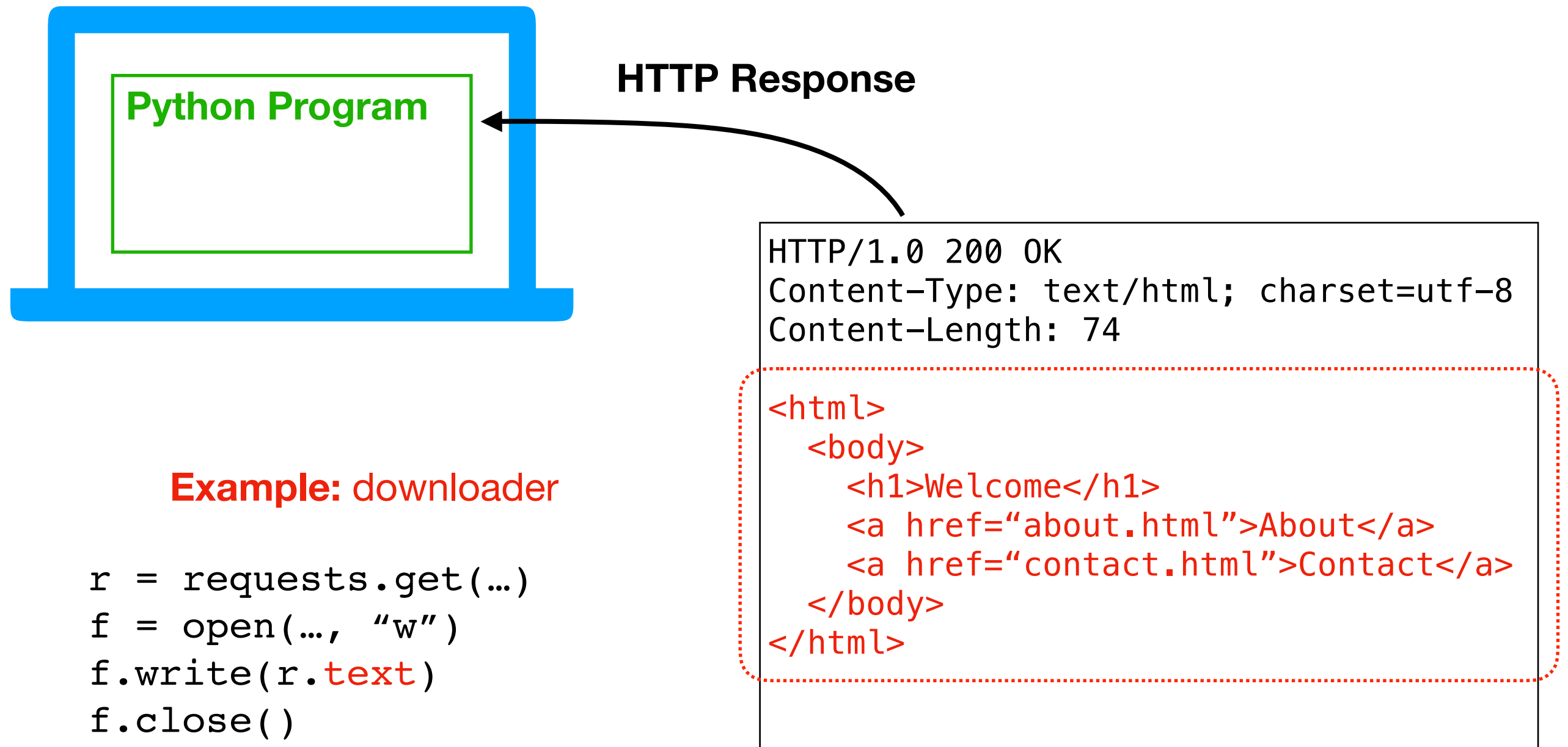


Depending on application, we may want to use:  
**1. HTTP information**



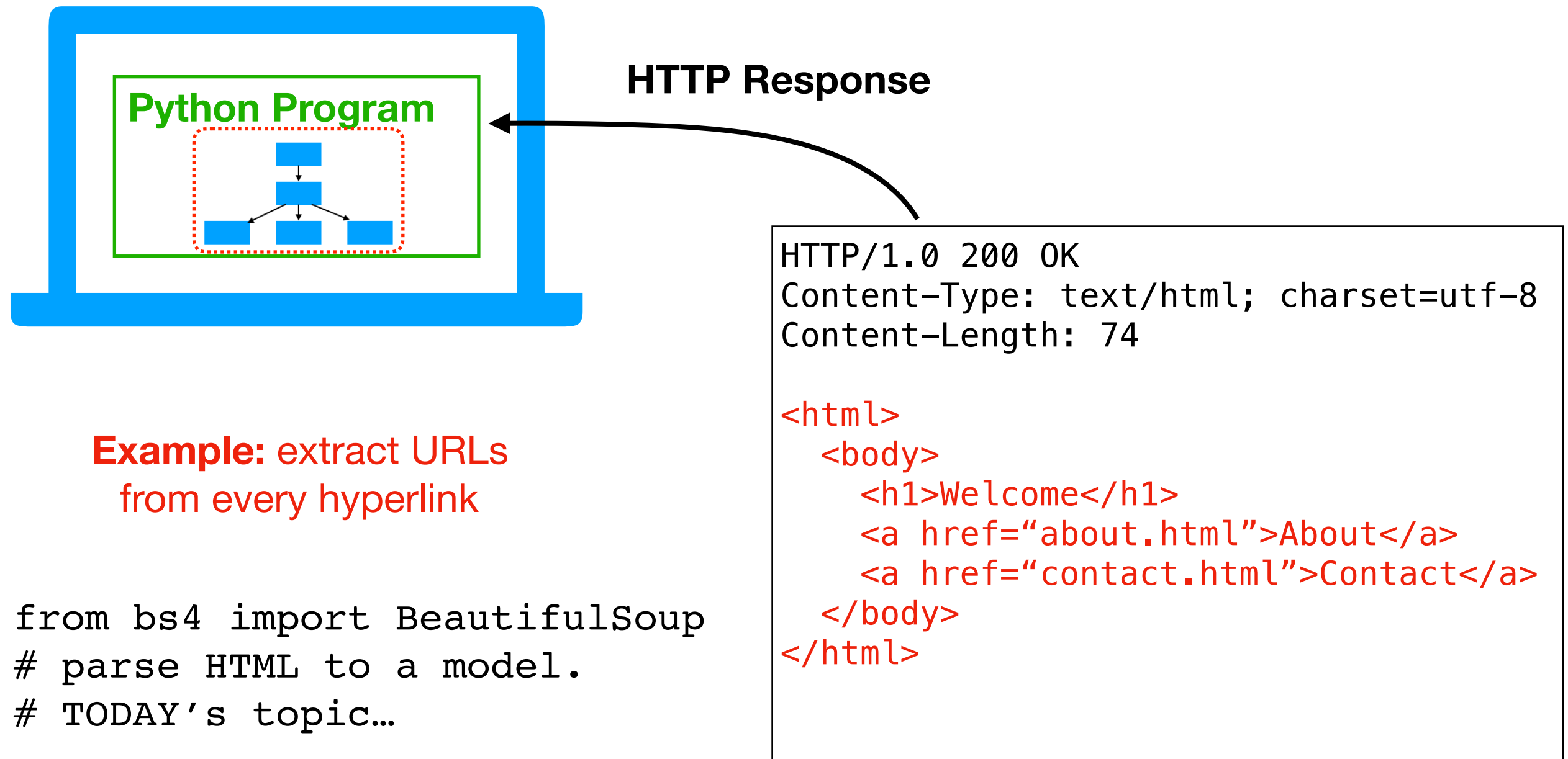
Depending on application, we may want to use:

1. HTTP information
2. **raw HTML (or JSON, CSV, etc)**



Depending on application, we may want to use:

1. HTTP information
2. raw HTML (or JSON, CSV, etc)
3. **model of HTML document**





# Outline

Document Object Model

BeautifulSoup module

Scraping States from Wikipedia

# BeautifulSoup module

## Purpose

- convert HTML (downloaded from the web or otherwise) to a model of **elements**, **attributes**, and **text**
- simple functions for searching for elements for a particular type (e.g., find all “a” tags to extract all hyperlinks)

## Installation

- comes with Anaconda
- otherwise run this:

```
pip install beautifulsoup4
```

## Using it

- just import:

```
from bs4 import BeautifulSoup
```

# Parsing HTML

new type

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

## Items

- x
- y
- z

# Parsing HTML

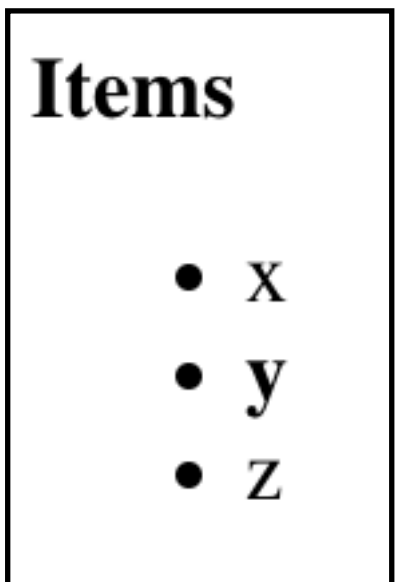
```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```



this could have come from anywhere:

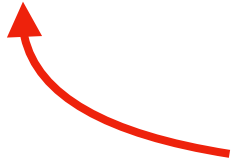
- hardcoded string
- something from requests GET
- loaded from local file



# Parsing HTML

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```



we'll always use this  
(other strings parse  
other formats)

## Items

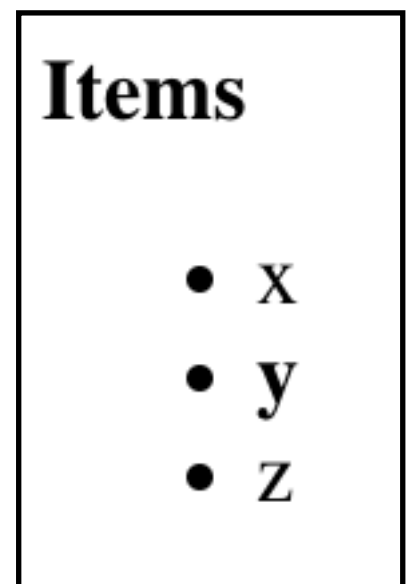
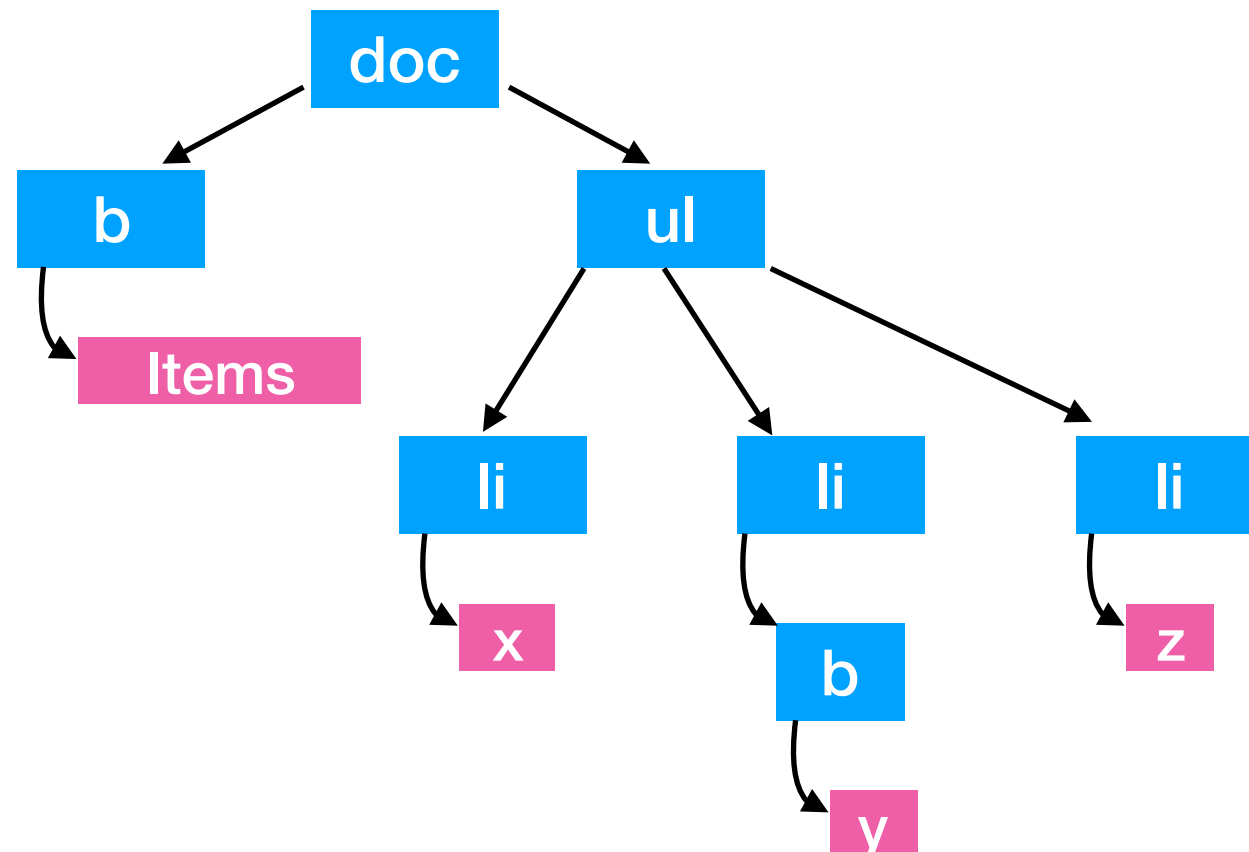
- x
- y
- z

# Parsing HTML

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

document object that  
we can easily analyze

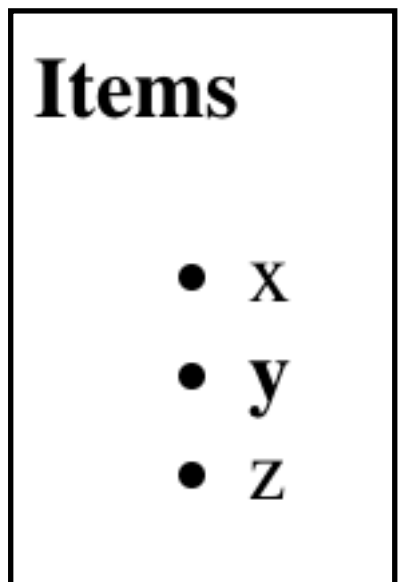


# Parsing HTML

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
print(doc.prettyfy())
```



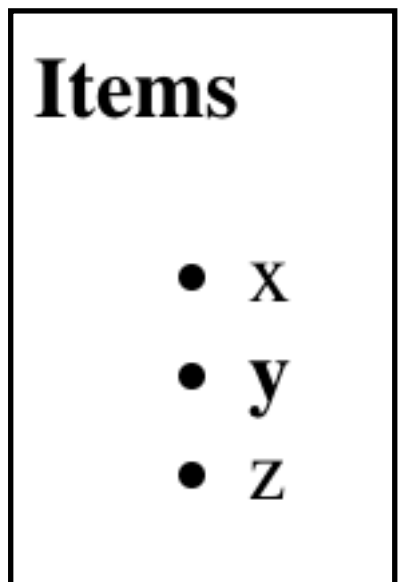
# Parsing HTML

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
print(doc.prettyfy())
```

```
<b>  
  Items  
</b>  
<ul>  
  <li>  
    x  
  </li>  
  <li>  
    <b>  
      y  
    </b>  
  </li>  
  <li>  
    z  
  </li>  
</ul>
```



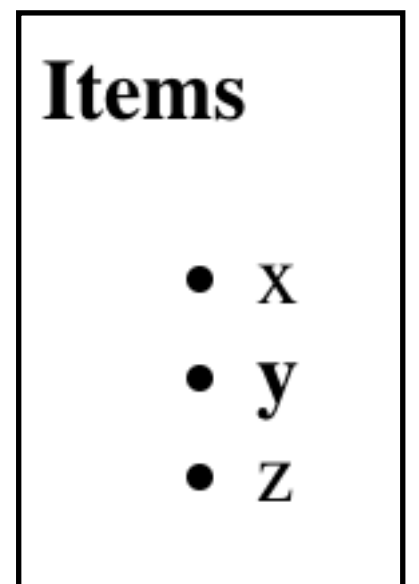
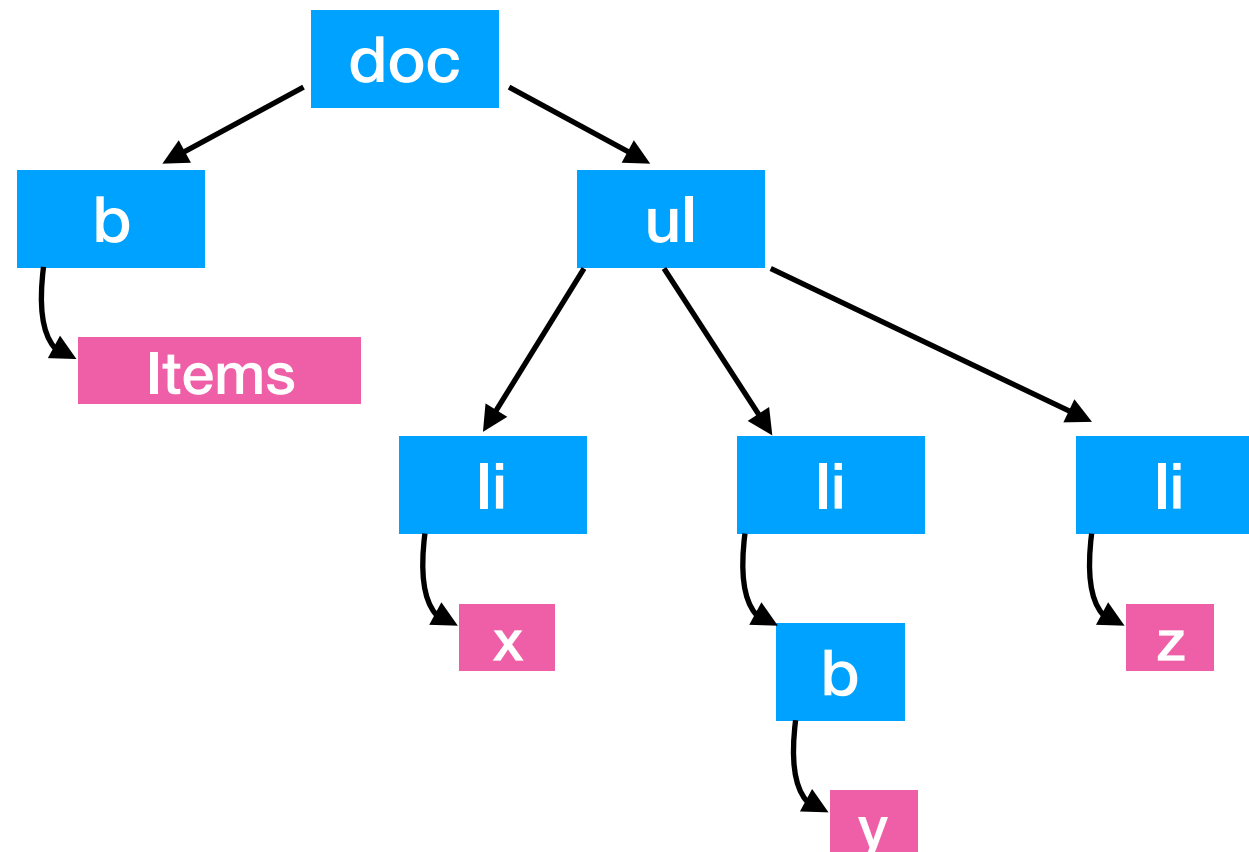


# Searching for Elements

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("li")  
print(len(elements))
```

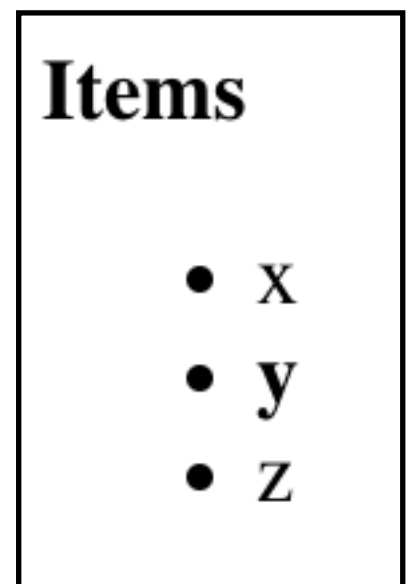
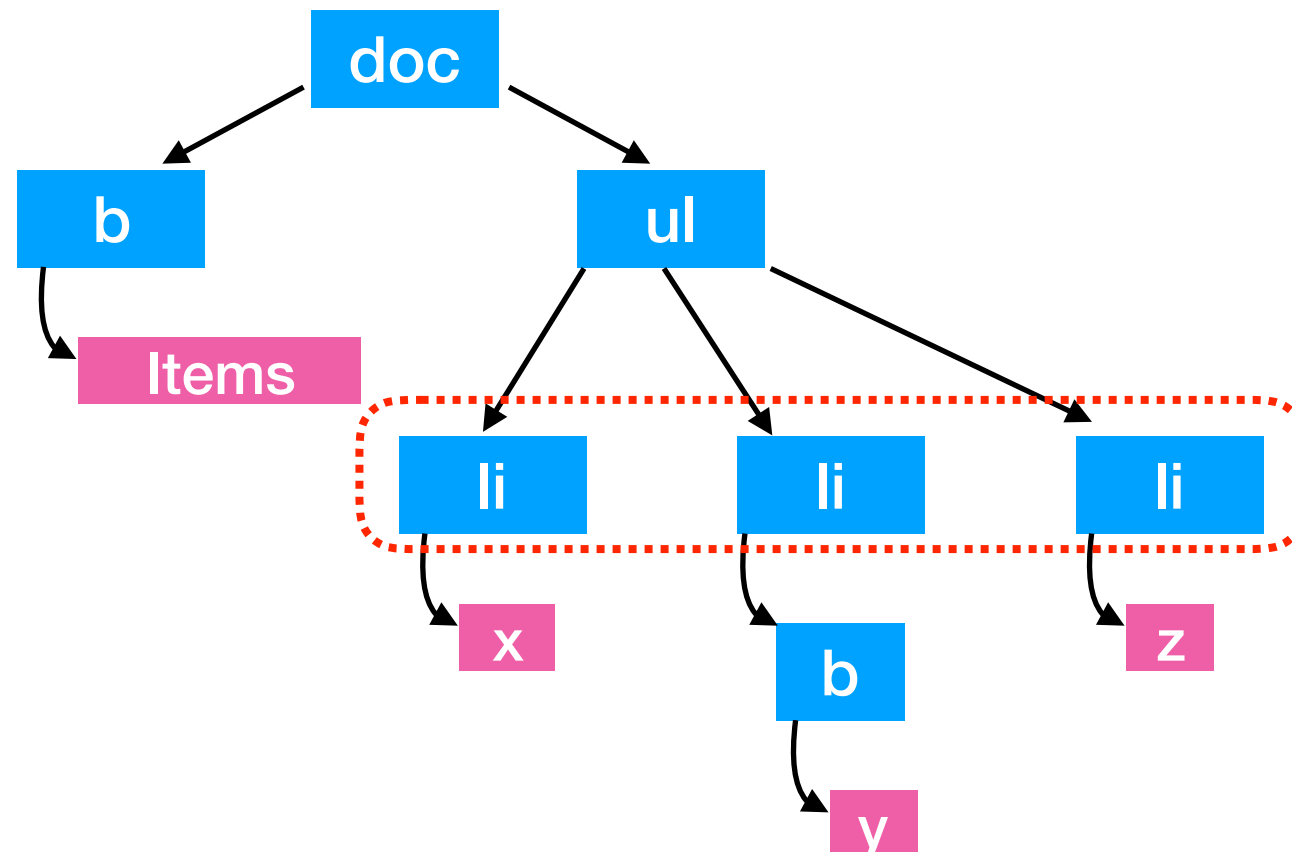


# Searching for Elements

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("li")  
print(len(elements))
```

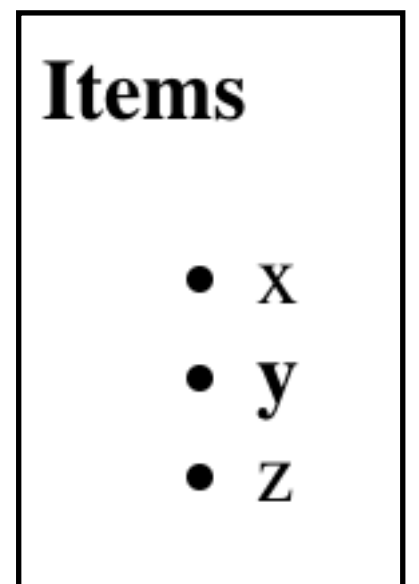
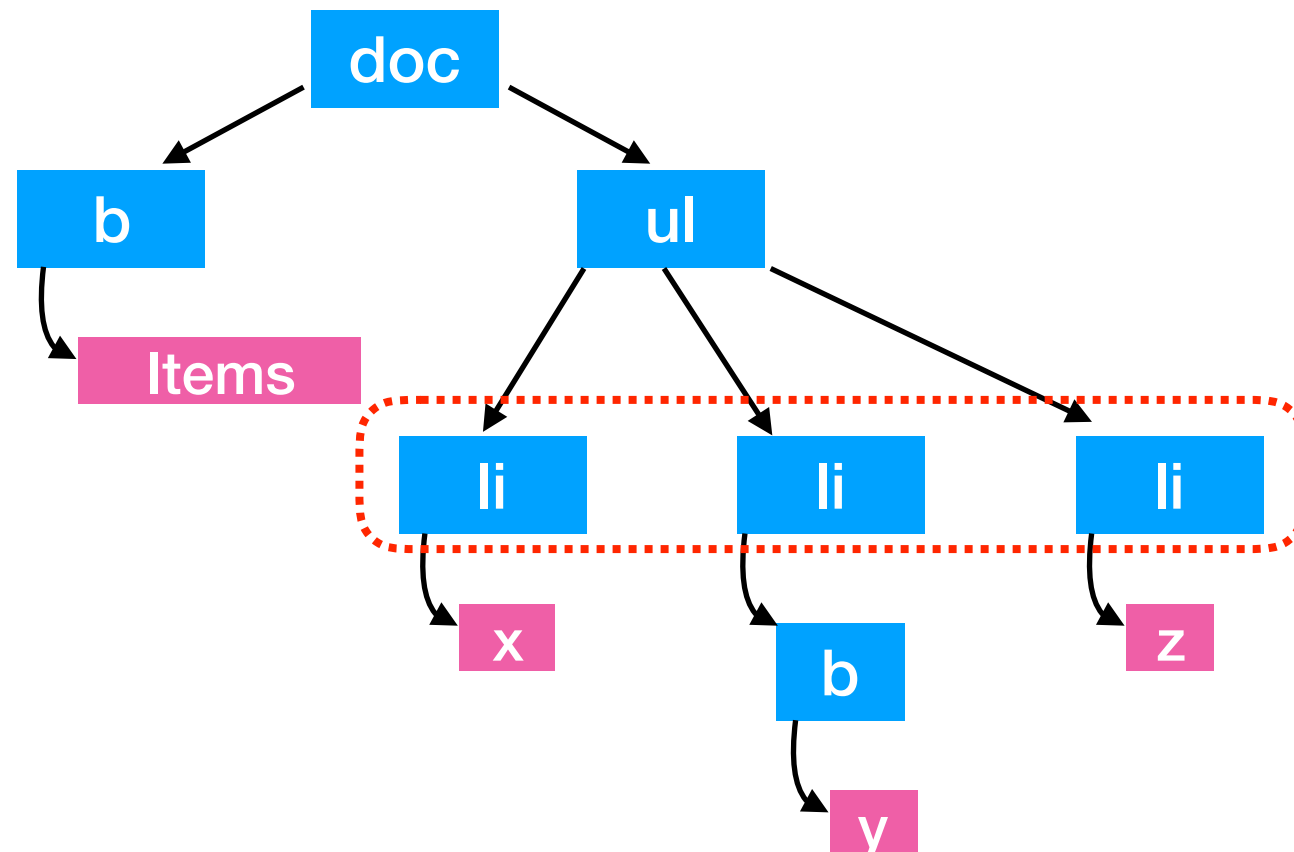


# Searching for Elements

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("li")    list of three elements  
print(len(elements))             prints 3
```

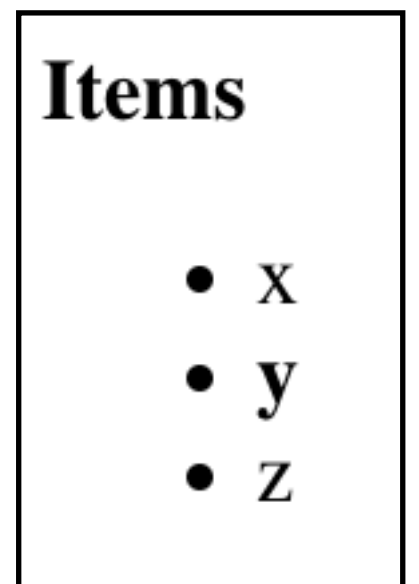
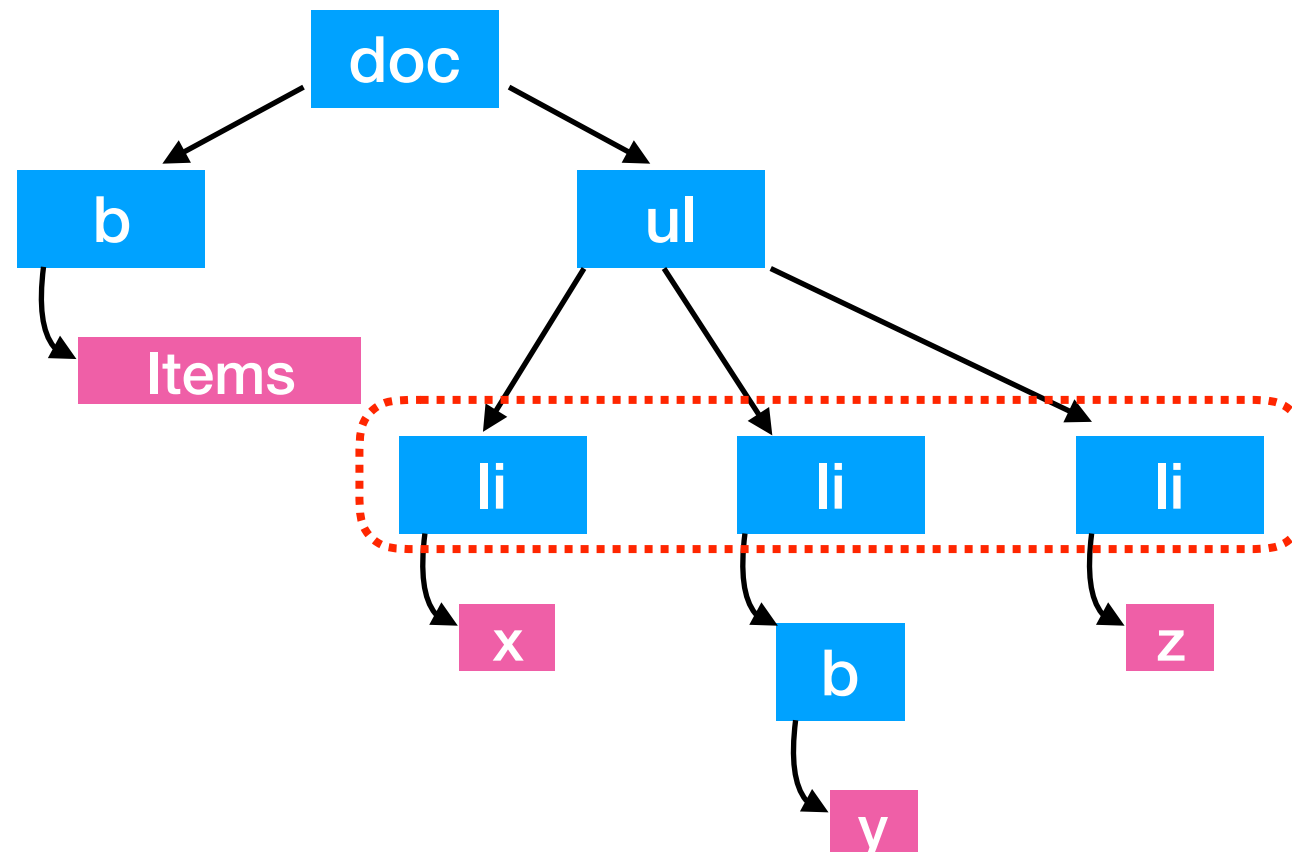


# Searching for Elements

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("li")    list of three elements  
print(len(elements))             prints 3
```



# Extracting Text

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("li")  
print(len(elements))
```

```
for e in elements:  
    print(e.get_text())
```

**Prints:**

x  
y  
z

**Items**

- x
- y
- z

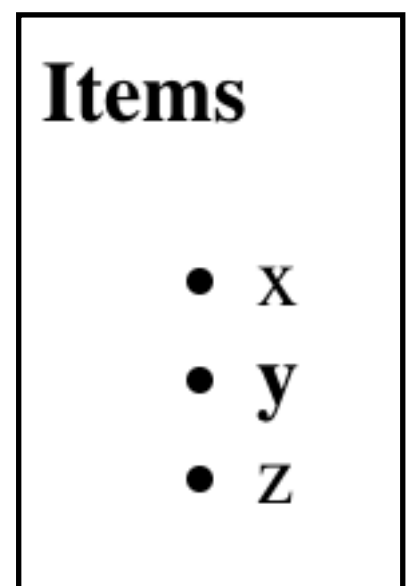
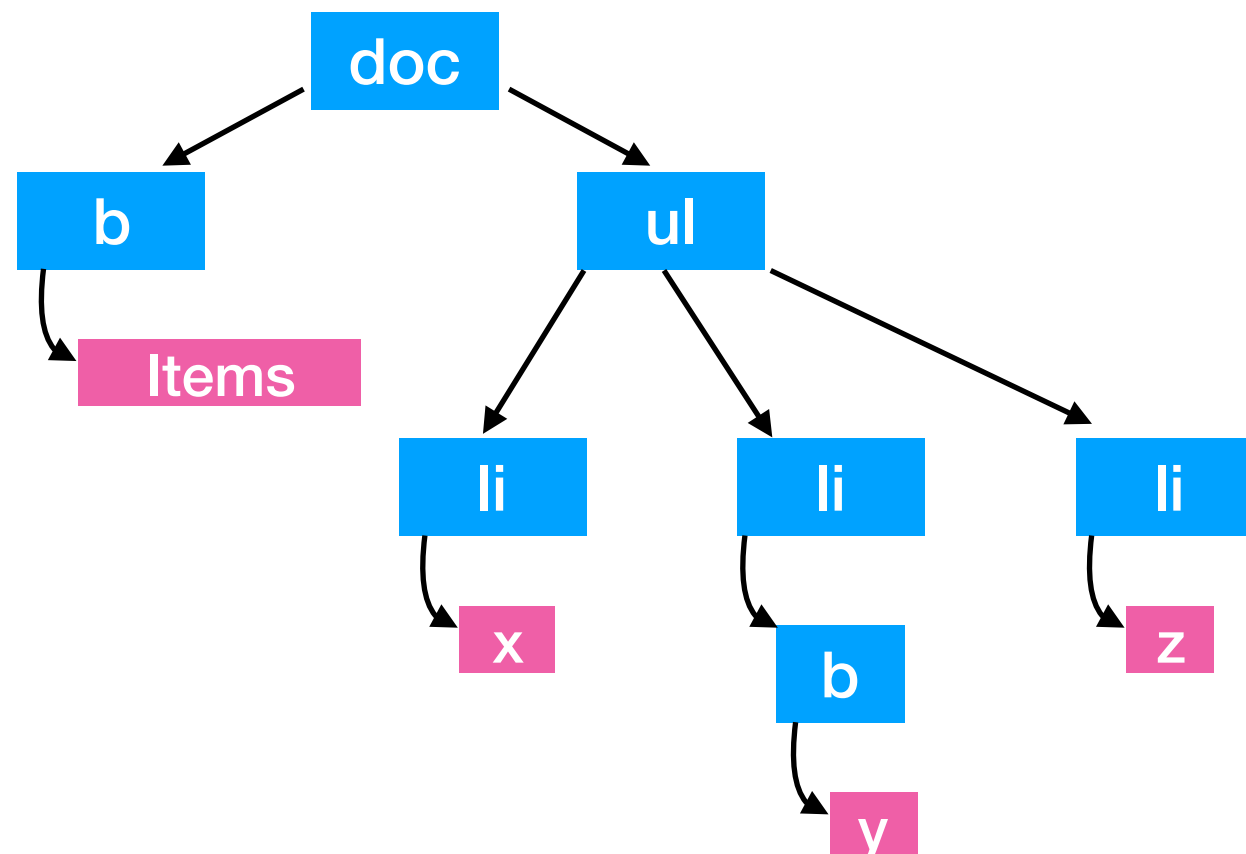
# Parsing HTML

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("b")  
print(len(elements))
```

now look for all bold elements



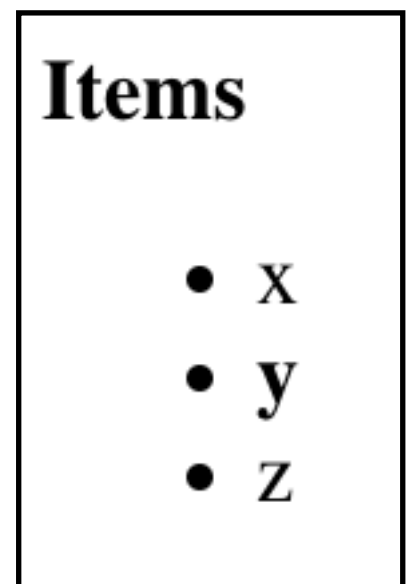
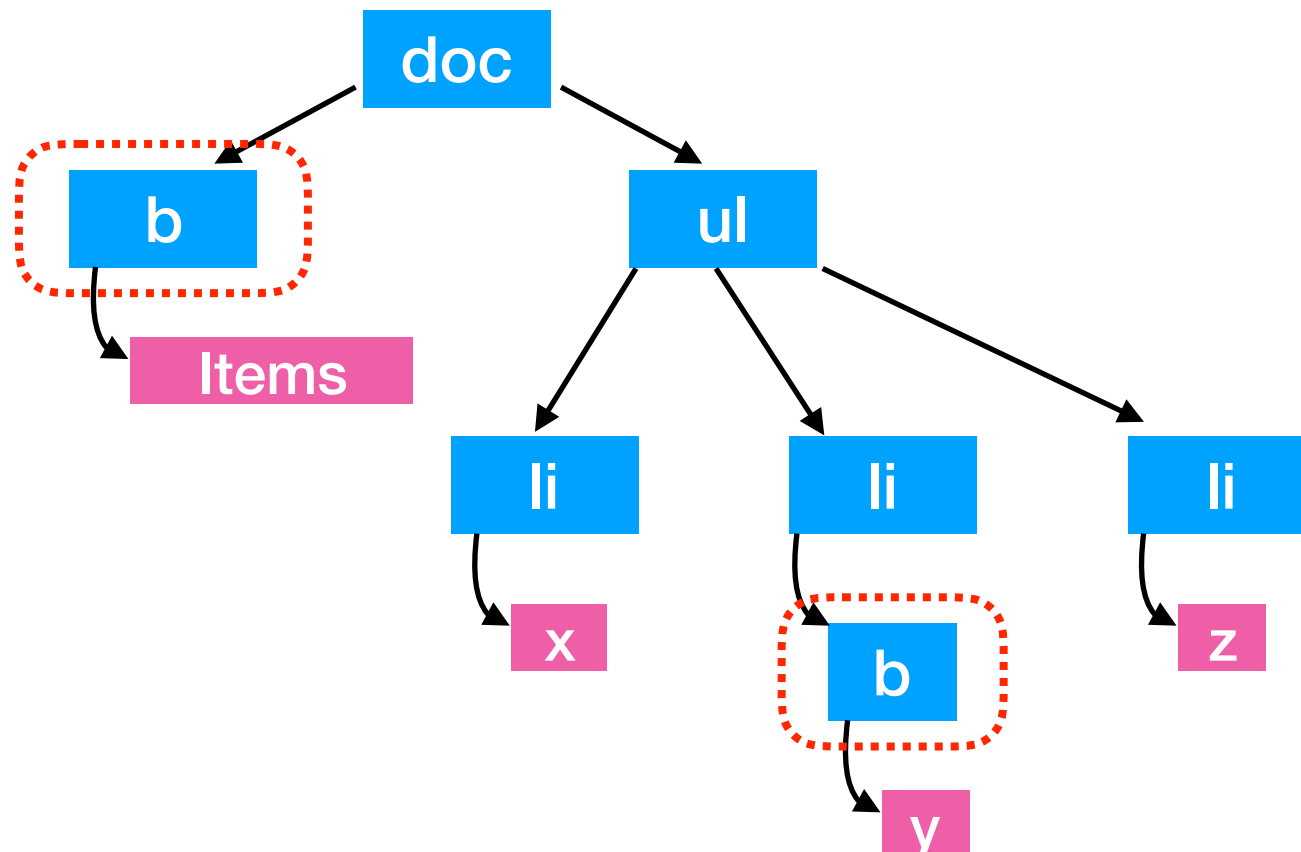
# Searching for Elements

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("b")  
print(len(elements))
```

now look for all bold elements

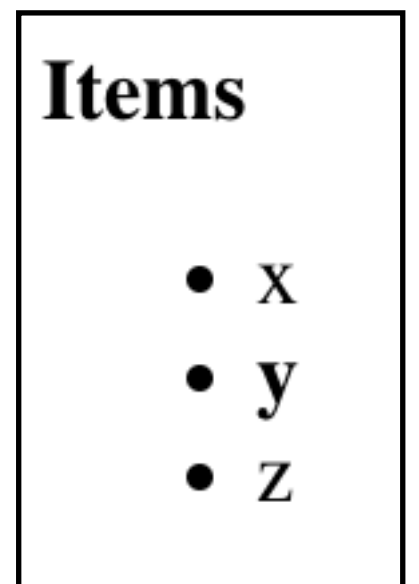
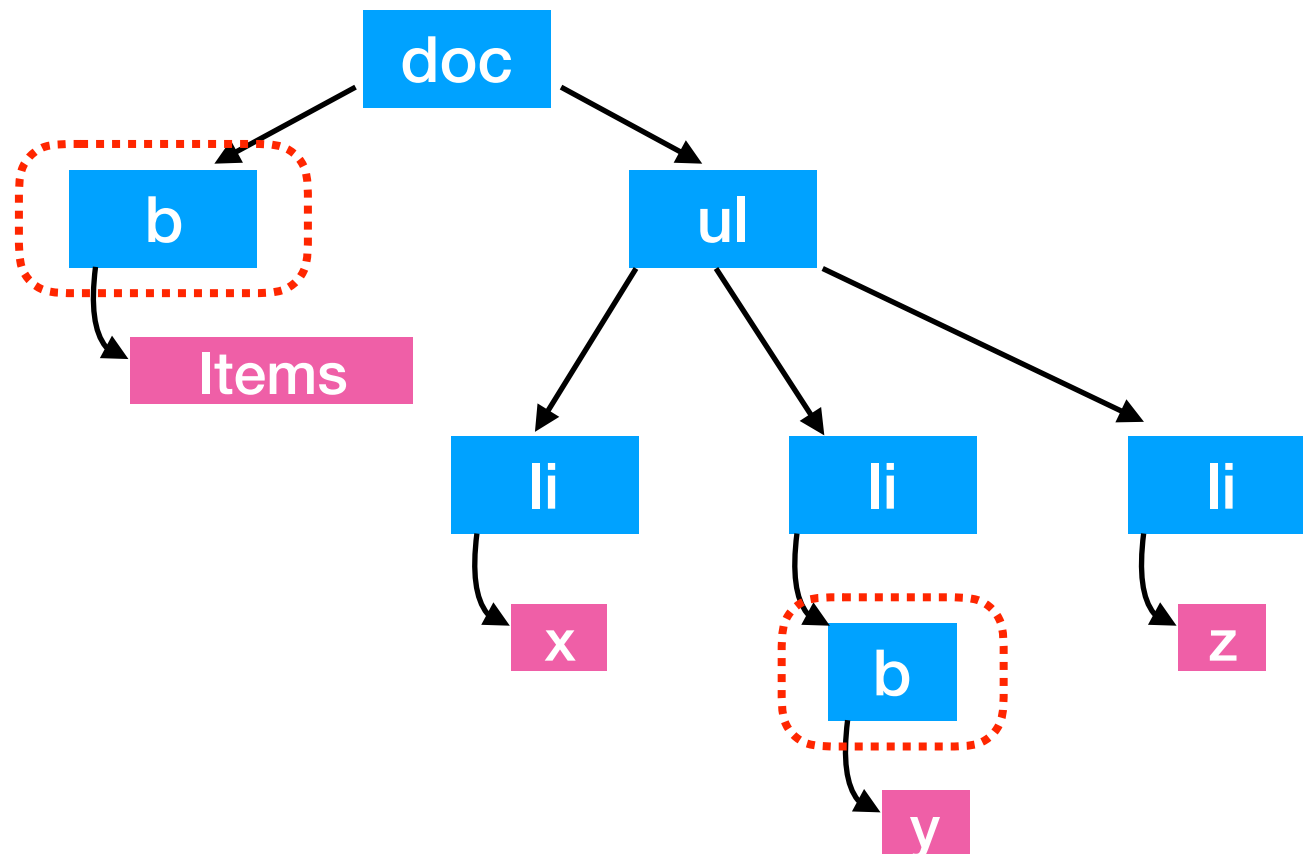


# Searching for Elements

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
elements = doc.find_all("b")    list of two elements  
print(len(elements))           prints 2
```





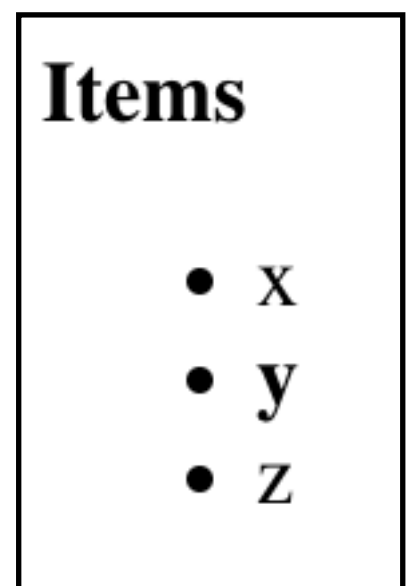
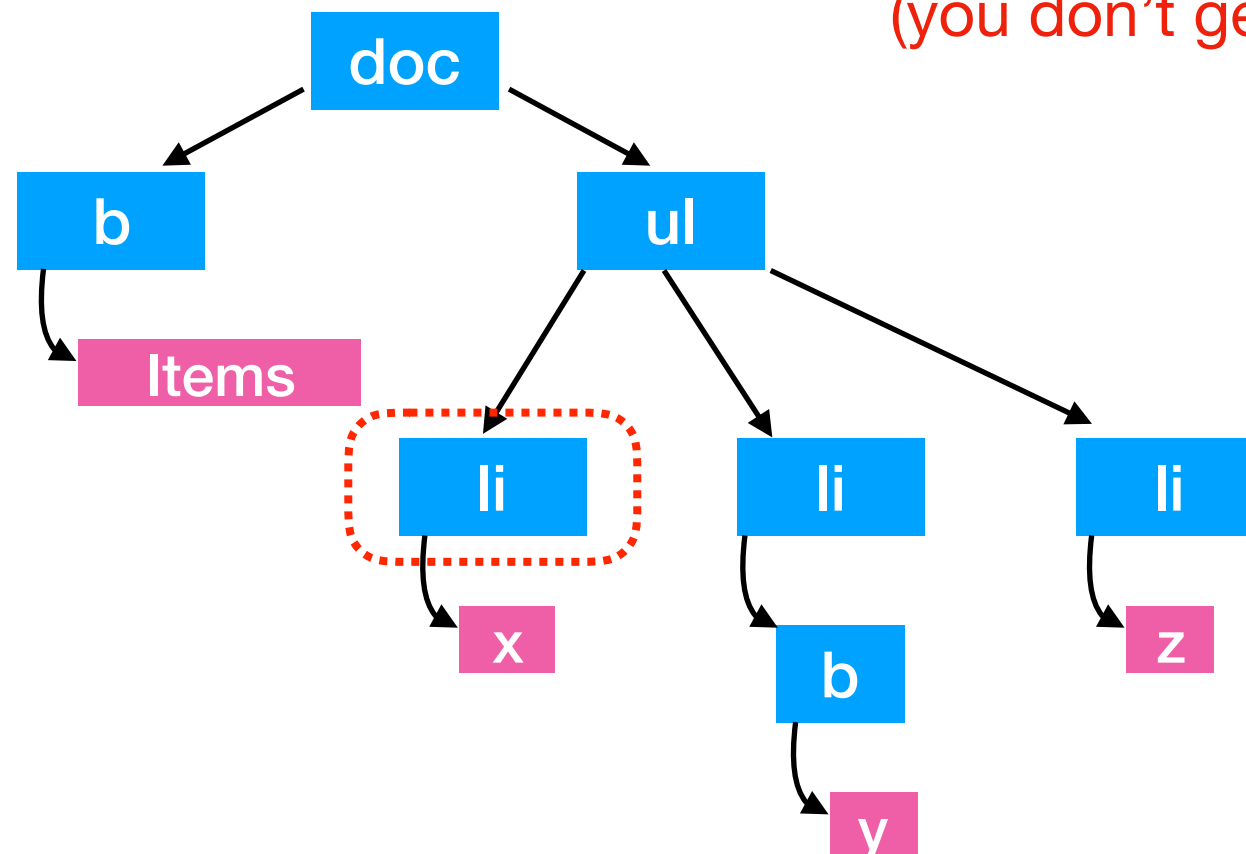
# Find One

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
li = doc.find("li")  
assert(li != None)
```

find just grabs the first one  
(you don't get a list)

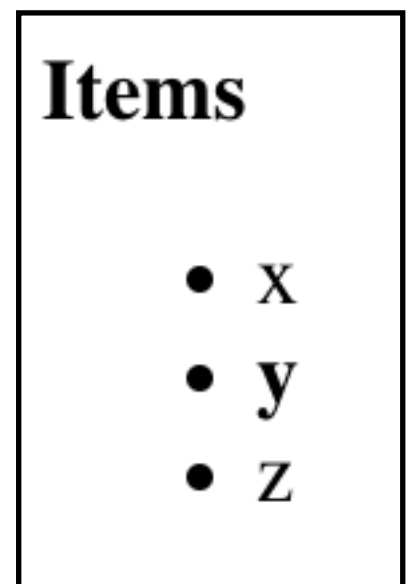
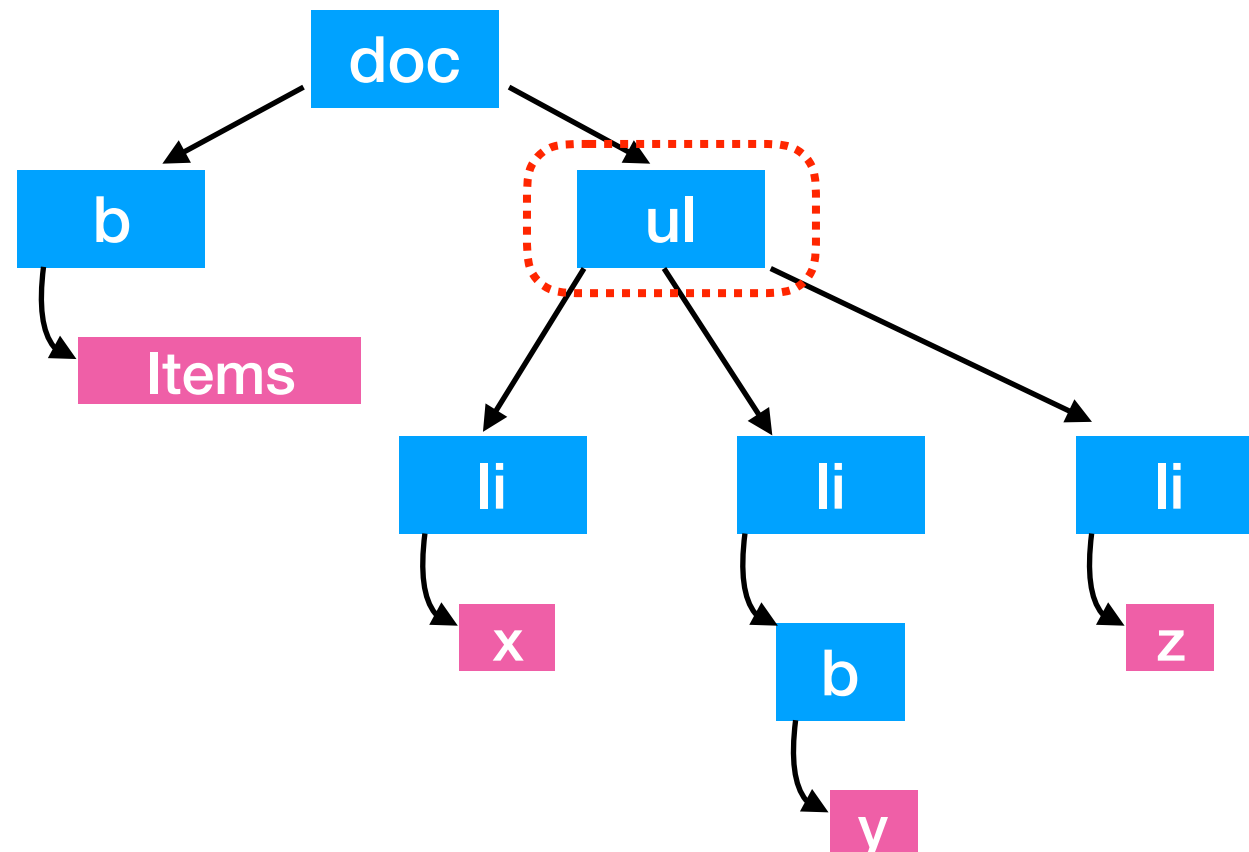


# Find One

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
ul = doc.find("ul")  
assert(ul != None)
```



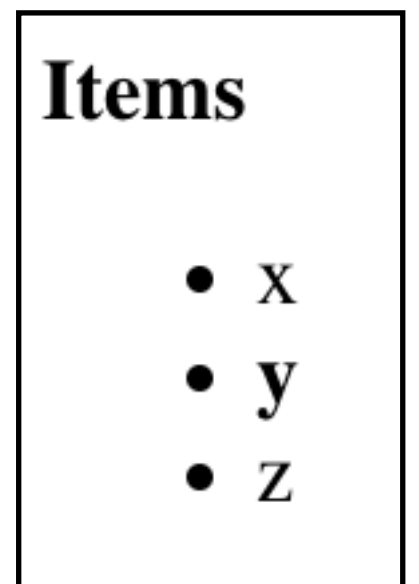
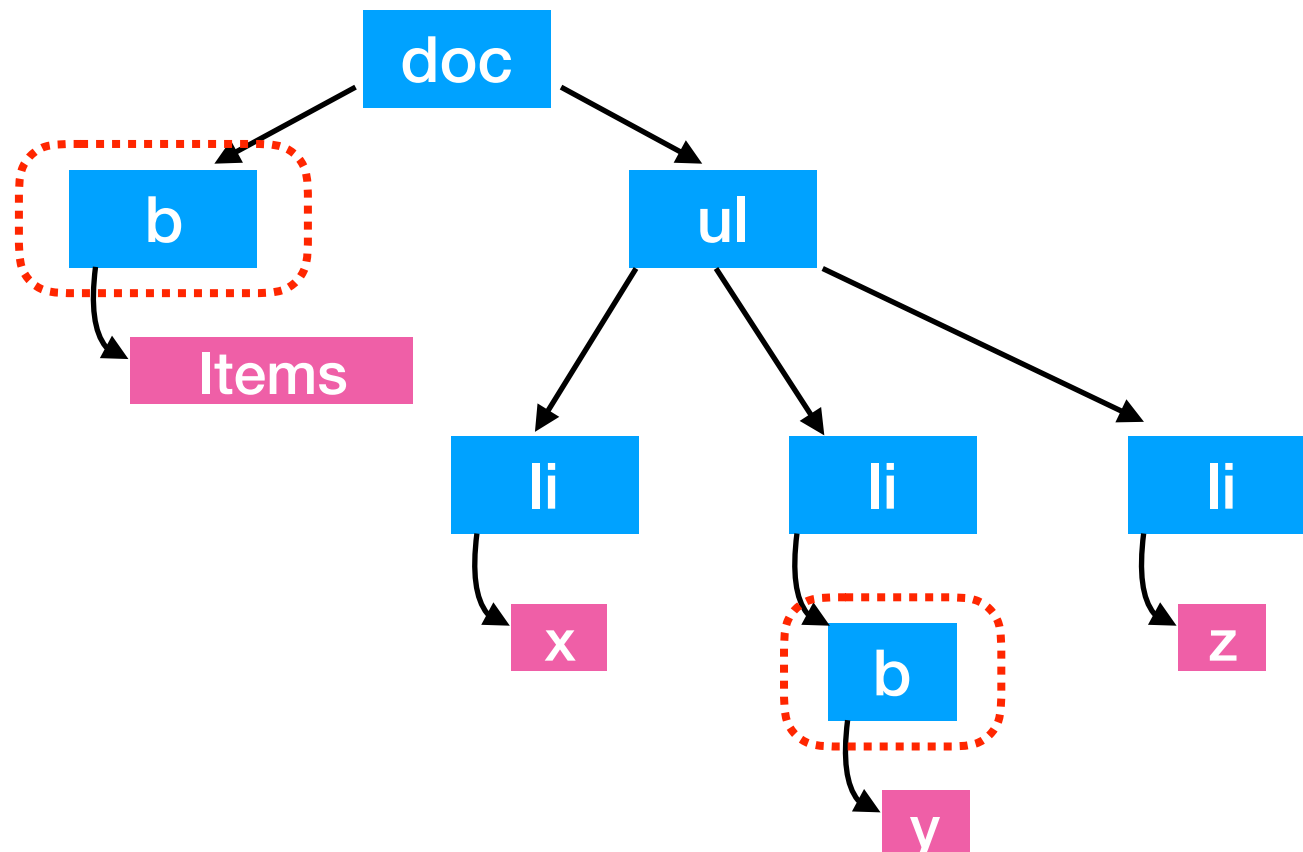
# Wide Search

```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
ul = doc.find("ul")
```

```
doc.find_all("b")
```

*find all bold text in the document*

# Narrower Search

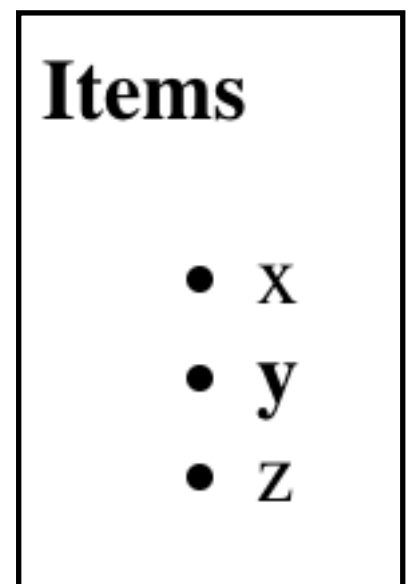
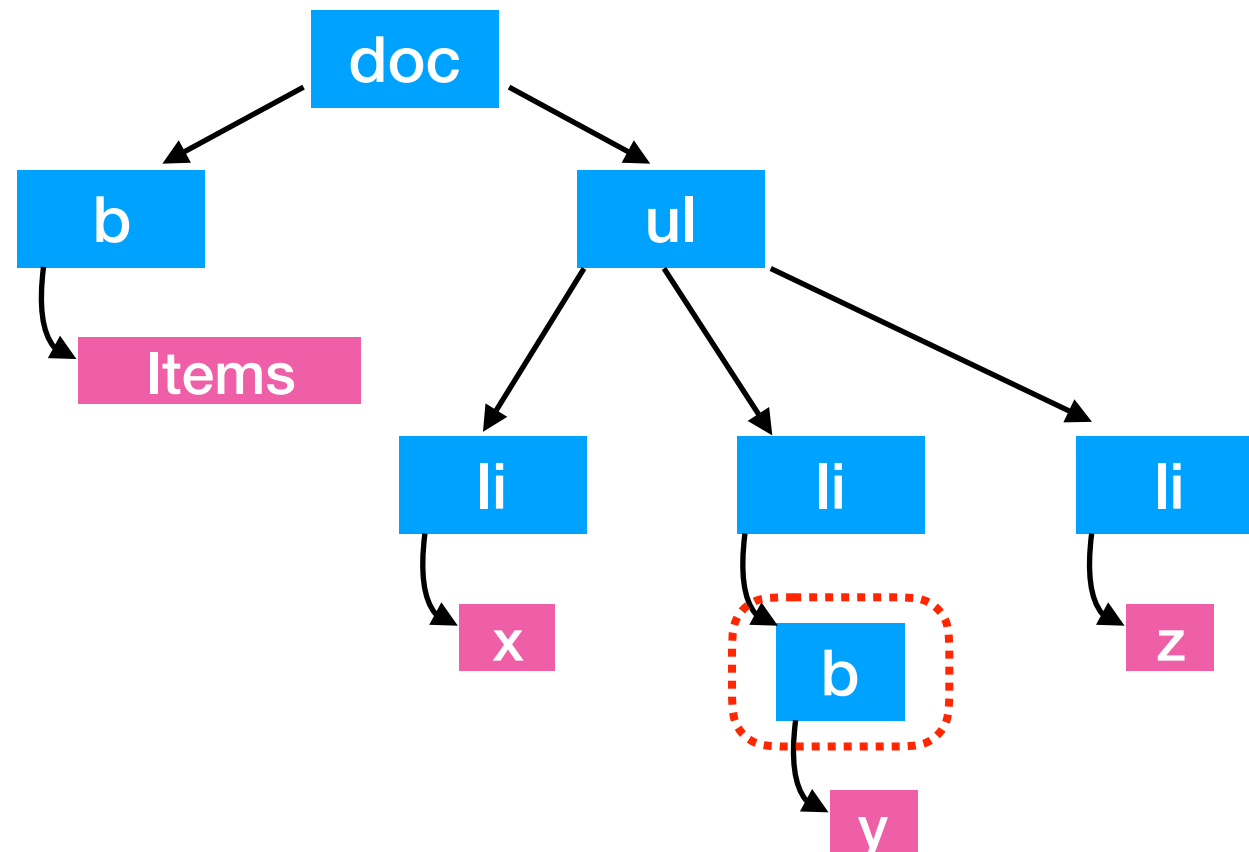
```
from bs4 import BeautifulSoup
```

```
html = "<b>Items</b><ul><li>x</li><li><b>y</b></li><li>z</li></ul>"  
doc = BeautifulSoup(html, "html.parser")
```

```
ul = doc.find("ul")
```

```
ul.find_all("b")
```

*find all bold text in the unordered list*



# Inspecting an Element

Remember! **Elements** may contain:

- attributes
- text
- other **elements**

# Inspecting an Element

Remember! **Elements** may contain:

- attributes
- text
- other **elements**

---

[what you see]

[please click here](#)

# Inspecting an Element

Remember! **Elements** may contain:

- attributes
- text
- other **elements**

---

[what you see]

[please click here](#)

---

[HTML]

```
<a href="schedule.html"><i>please</i> click <b>here</b></a>
```

# Inspecting an Element

Remember! **Elements** may contain:

- attributes
- text
- other **elements**

---

[what you see]

[please click here](#)

---

[HTML]

```
<a href="schedule.html"><i>please</i> click <b>here</b></a>
```

---

[Python]

```
link = doc.find("a")
```



# Inspecting an Element

Remember! **Elements** may contain:

- attributes
- text
- other **elements**

---

[what you see]

[\*please\* click \*\*here\*\*](#)

---

[HTML]

`<a href="schedule.html"><i>please</i>click<b>here</b></a>`

---

[Python]

```
link = doc.find("a")
list(link.children)
```

**Result:**

italic element	click text	bold element
----------------	------------	--------------

(list)

# Inspecting an Element

Remember! **Elements** may contain:

- attributes
- text
- other **elements**

---

[what you see]

[please click here](#)

---

[HTML]

```
<a href="schedule.html"><i>please</i> click <b>here</b></a>
```

---

[Python]

```
link = doc.find("a")
link.get_text()
```

**Result:** please click here  
(str)

# Inspecting an Element

Remember! **Elements** may contain:

- attributes
- text
- other **elements**

---

[what you see]

[please click here](#)

---

[HTML]

```
<a href="schedule.html"><i>please</i> click <b>here</b></a>
```

---

[Python]

```
link = doc.find("a")
link.attrs
```

**Result:** {'href': 'schedule.html'}  
(dict)

# Outline

Document Object Model

BeautifulSoup module

Scraping States from Wikipedia

# Demo Stage 1: Extract Links from Wikipedia

Goal: scrape links to all articles about US states from a table on a wiki page (check this: <https://simple.wikipedia.org/robots.txt>)

## Input:

- [https://simple.wikipedia.org/wiki/List\\_of\\_U.S.\\_states](https://simple.wikipedia.org/wiki/List_of_U.S._states)

## Output:

- <https://simple.wikipedia.org/wiki/Alabama>
- <https://simple.wikipedia.org/wiki/Alaska>
- etc

### List of U.S. states

From Wikipedia, the free encyclopedia

A **U.S. state** is one of the [states](#) of the [United States of America](#). Four states (Kentucky, Massachusetts, Pennsylvania, and the twenty-first, 1959.

The states are labeled with their [U.S. postal abbreviations](#), their founding date and [capitals](#).

Sl no. ↕	Abbreviations ↕	State Name ↕	Capital ↕	Became a State ↕
1	AL	<a href="#">Alabama</a>	<a href="#">Montgomery</a>	December 14, 1819
2	AK	<a href="#">Alaska</a>	<a href="#">Juneau</a>	January 3, 1959
3	AZ	<a href="#">Arizona</a>	<a href="#">Phoenix</a>	February 14, 1912
4	AR	<a href="#">Arkansas</a>	<a href="#">Little Rock</a>	June 15, 1836

# Demo Stage 2: Download State Pages

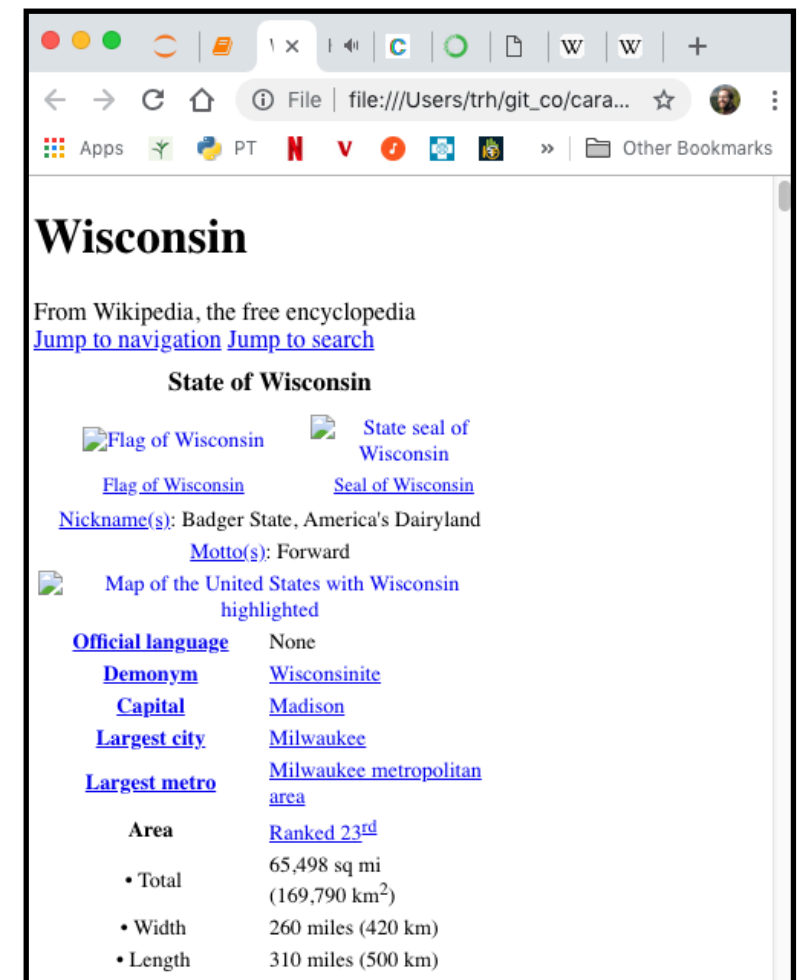
Goal: download all Wiki pages for the states

## Input:

- Links generated in stage 1:
- <https://simple.wikipedia.org/wiki/Alabama>
- <https://simple.wikipedia.org/wiki/Alaska>
- etc

## Output Files:

- Alabama.html
- Alaska.html
- etc



# Demo Stage 3: Biggest City

Goal: find the biggest city in each state

## Input:

- HTML files generated from stage 2
- Alabama.html
- Alaska.html
- etc

## Output:

- dictionary mapping states to largest cities

```
Out[7]: {'Alabama': 'Birmingham',  
        'Alaska': 'Anchorage',  
        'Arizona': 'Phoenix',  
        'Arkansas': 'Little Rock',  
        'California': 'Los Angeles',  
        'Colorado': 'Denver',  
        'Connecticut': 'Bridgeport',  
        'Delaware': 'Wilmington',  
        'Florida': 'Jacksonville',  
        'Georgia': 'Atlanta',  
        'Hawaii': 'Honolulu',  
        'Idaho': 'Boise',  
        'Illinois': 'Chicago',  
        'Indiana': 'Indianapolis',  
        'Iowa': 'Des Moines',  
        'Kansas': 'Wichita[3]',  
        'Kentucky': 'Louisville',  
        'Louisiana': 'New Orleans[2][3][4]',  
        'Maine': 'Portland',
```