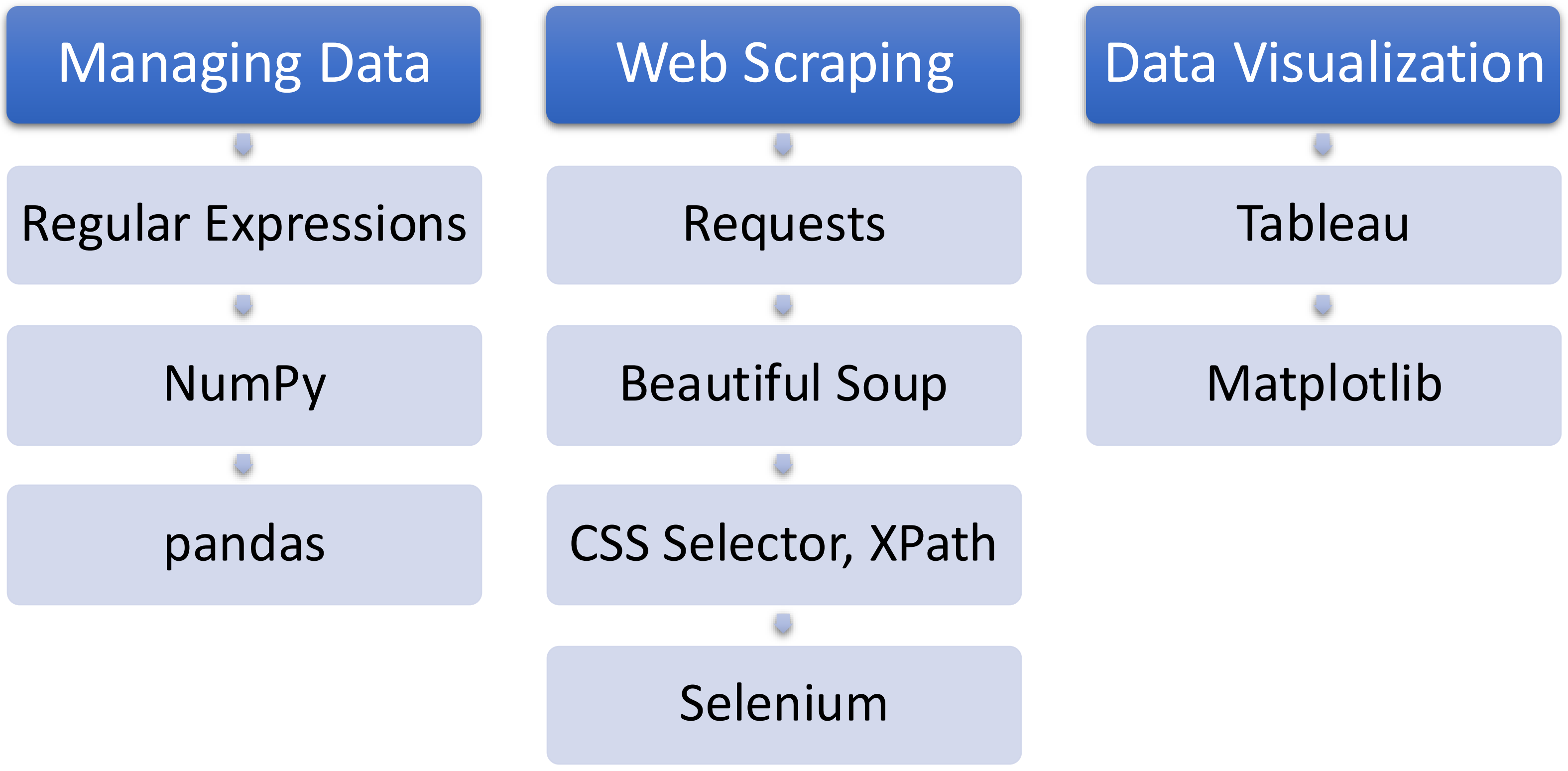
Web Scraping II

MSBA7001 Business Intelligence and Analytics
HKU Business School
The University of Hong Kong

Instructor: Dr. DING Chao

Course Roadmap



Agenda

- CSS Selectors
- XPath

CSS Selectors

What is CSS?

- CSS stands for **C**ascading **S**tyle **S**heets.
- It is a language that describes the style of HTML elements. For example: color, alignment, size, etc.

```
body {
  background-color: lightblue;
}

#city {
  color: white;
  text-align: center;
}
```

- CSS selectors are patterns used to select the element(s) on an HTML page.

CSS Selectors

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element	div, p	Selects all <div> elements and all <p> elements
element element	div p	Selects all <p> elements inside <div> elements
element>element	div > p	Selects all <p> elements where the parent is a <div> element
element+element	div + p	Selects all <p> elements that are placed immediately after <div> elements
element1~element2	p ~ ul	Selects every <ul> element that are preceded by a <p> element

https://www.w3schools.com/cssref/trysel.asp

CSS Selectors

Selector	Example	Example description
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent

Select HTML Elements with CSS Selectors

- BeautifulSoup supports the most commonly-used CSS selectors.
- Just pass a string into the `select` method of a tag object or the soup object itself.
- The output is a list object.

```
soup.select("title")
```

```
[<title>The King’s story</title>]
```

```
soup.select("p:nth-of-type(3)")
```

```
[<p class="story">...</p>]
```

XPath

XML

- XML stands for eXtensible Markup Language.
- It is designed to store and transport data, especially over the Internet.

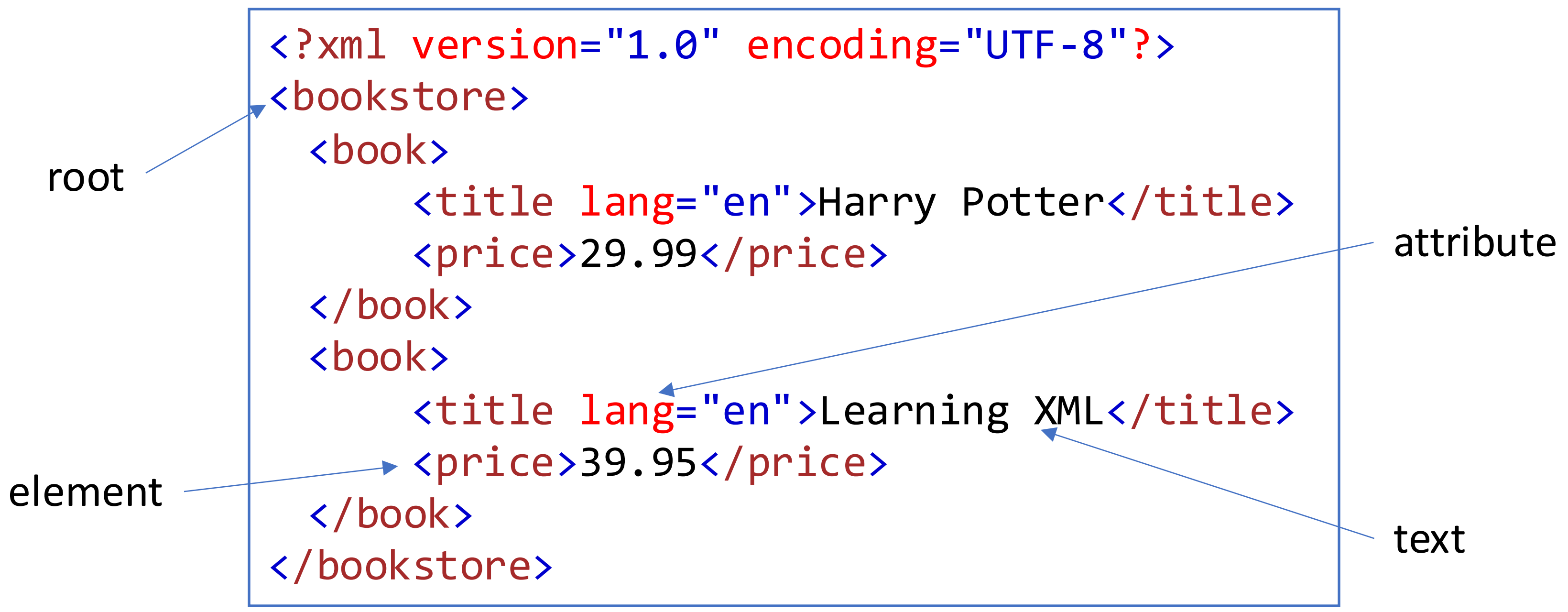
```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <body>Don't forget me this weekend!</body>
</note>
```

- The structure looks similar to HTML.
- HTML focuses on data presentation, while XML is more about data interchange and storage. HTML tags are predefined, whereas XML tags are user-defined.

XML Nodes

- XML documents are treated as trees of nodes.
- Common nodes include element, attribute, text, comment, and root nodes.

https://www.w3schools.com/xml/default.asp



```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book>
      <title lang="en">Harry Potter</title>
      <price>29.99</price>
  </book>
  <book>
      <title lang="en">Learning XML</title>
      <price>39.95</price>
  </book>
</bookstore>
```

Work with XML Documents

- For local XML documents, we could simply read them by pandas or use BeautifulSoup to navigate the nodes.

```
pd.read_xml('filepath.xml')
```

```
raw = open('filepath.xml', 'r').read()
soup = BeautifulSoup(raw, 'xml')
```

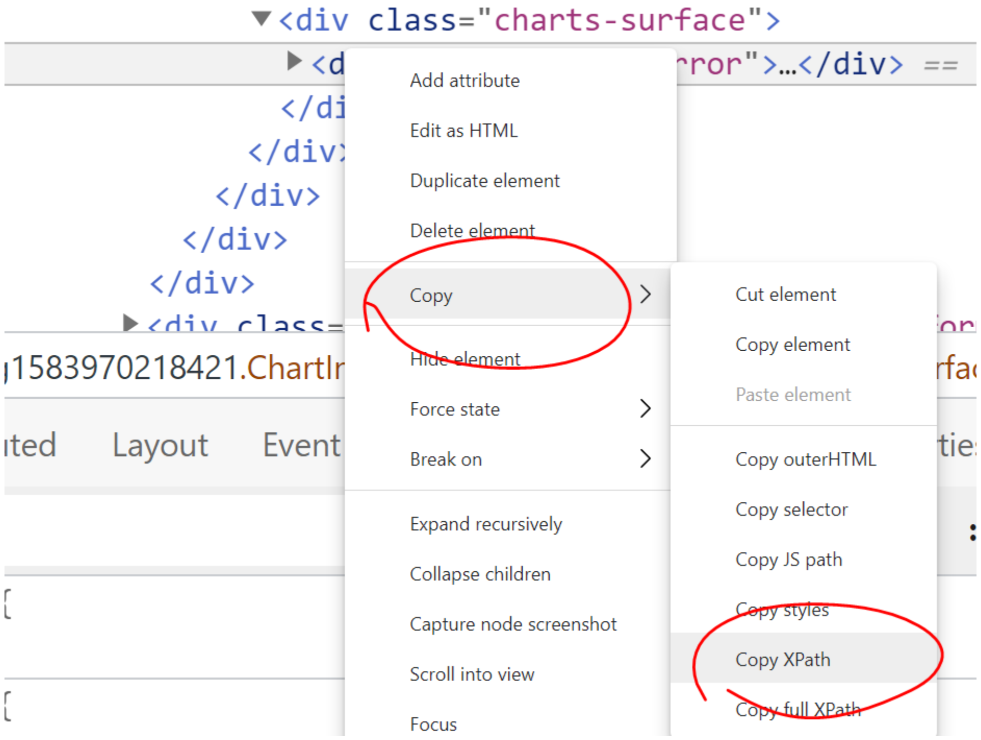
- For online XML documents, we could work with them like HTML pages.

```
url = 'http://xxxxxxx/xxx.xml'
raw = requests.get(url).text
soup = BeautifulSoup(raw, 'xml')
```

- However, these approaches have limitations. For advanced navigation, we use XPath.

What is XPath?

- XPath stands for **XML Path Language**.
- It uses “path like” syntax to navigate nodes in an XML document. Just like CSS selectors to HTML pages.
- XPath can also be used to navigate HTML elements.



```
▼<div class="charts-surface">
  ▶<d        ror">…</div> ==
    </di
   </div>
  </div>
 </div>
</div>
▶<div class=
```

XPath Syntax

- XPath uses path expressions to select nodes.

Example	Example description
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore Note: If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

XPath Syntax

- Predicates are used to find a specific node or a node that contains a specific value. Include them in square brackets. This is a key advantage of XPath over CSS selectors.

Example	Example description
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

XPath Syntax

- * is a wildcard that matches with any node.
- | is used to select several paths.

Example	Example description
/bookstore/*	Selects all the child element nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have at least one attribute of any kind
//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all the title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

Select HTML Elements with XPath

- BeautifulSoup by default doesn't support XPath.
- We need to convert the soup object to an etree object by using the lxml module.

Parse the HTML content of the page.
Argument must be a string.

```
from bs4 import BeautifulSoup
from lxml import etree
import requests

source = requests.get(url).text
soup = BeautifulSoup(source, "html.parser")
dom = etree.HTML(str(soup))
print(dom.xpath('//*[@id="link4"]')[0].text)
```

Chao

Returns a list of matches elements

CSS Selectors vs. XPath

- CSS selectors are primarily used for styling and locating elements, while XPath provides more advanced features and flexibility.
- CSS selectors have a simpler and more readable syntax compared to XPath, which makes them easier for testers to construct and maintain.
- XPath & CSS cheatsheet: https://quickref.me/xpath.html

	CSS Selectors	XPath
Advantages	Faster, simpler syntax	DOM navigation, flexibility, compatibility
Use Cases	Styling web pages, locating elements	Web scraping, parsing HTML