Semi-Structured Data

School of Information Studies
Syracuse University

Semi-Structured Data

Well-formatted in a tag system

Passing data over the Internet

- HTML—describing data and appearance of Web pages
- XML—hierarchical tag system
- JSON—lighter-weight tag system

Features of Markup Languages

Tree-structured (hierarchical) format

Elements surrounded by opening and closing tags

Attributes embedded in open tags

<tag-name attr-name="attribute"> data </tag-name>

Basics of Web Scraping

To handle HTML and XML data

JSON data

- Social media of Twitter and Facebook
- Unicode issues
- Storing data in NoSQL database MongoDB

Obtaining Data

JSON from APIs—more structured

HTML—more difficult; use as last resort

Using Python libraries for HTML and XML

Selenium—more advanced Web scraping

Advanced option—using bots

HTML

School of Information Studies
Syracuse University

HTML

Hyper Text Markup Language

System for formatting Web pages

Uses tags

Sample HTML Document

```
<html>
<head>
   <title>Page title here</title>
</head>
<body>
   This is sample text...
   <!-- We use this syntax to write comments -->
   <p>This is text within a paragraph.</p>
   <em>I <strong>really</strong> mean that</em>
   <img src="smileyface.jpg" alt="Smiley face" >
</body>
</html>
```

Simple Sample

Uses white space for structure

Many put numerous tags on one line

Main purpose of HTML

- Interpreted by browser
- Do not always use ending tags

urllib

Request package to retrieve file from ftp server

Connect with web servers using http protocol

Use of request and response data types

XML, DOM, and Element Tree

School of Information Studies
Syracuse University

XML

Extended Markup Language

Format for data interchange

Design your own tag names

Sample XML Document

```
<?xml version='1.0' encoding='utf-8'?>
<feed xmlns='http://www.w3.org/2005/Atom' xml:lang='en'>
<CATALOG>
<CD>
<TITLE>Empire Burlesque</TITLE>
<ARTIST>Bob Dylan</ARTIST>
<COUNTRY>USA</COUNTRY>
<COMPANY>Columbia</COMPANY>
<PRICE>10.90</PRICE>
<YEAR>1985</YEAR>
</CD>
<CD>
<TITLE>Hide your heart</TITLE>
<ARTIST>Bonnie Tyler</ARTIST>
<COUNTRY>UK</COUNTRY>
<COMPANY>CBS Records</COMPANY>
<PRICE>9.90</PRICE>
<YEAR>1988</YEAR>
</CD>
</CATALOG>
</feed>
```

Simple Sample

Uses beginning and ending tags

- <foo>
- </foo>

Comments <!-- and -->

Predefined entities

- &lt less than
- &gt greater than
- &amp ampersand &
- &apos apostrophe ‘
- &quot quote “

DOM

Document Object Model

Used to parse the data

Converts entire text to a structure

Produces a node for each tag and its children

DOM Sample

```
>>> import urllib.request
>>> url = "http://feeds.bbci.co.uk/news/rss.xml"
>>> xmlstring = urllib.request.urlopen(url).read().decode('utf8')
>>> len(xmlstring)
35071
>>> xmlstring[:500]
'<?xml version="1.0" encoding="UTF-8"?>\n<?xml-stylesheet
title="XSL_formatting" type="text/xsl"
href="/shared/bsp/xsl/rss/nolsol.xsl"?>\n<rss
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:content="http://purl.org/rss/1.0/modules/content/"
xmlns:atom="http://www.w3.org/2005/Atom" version="2.0"
xmlns:media="http://search.yahoo.com/mrss/">\n    <channel>\n
<title><![CDATA[BBC News - Home]]></title>\n
<description><![CDATA[BBC News - Home]]></description>\n
<link>http://www.bbc.co'
```

Element Tree

Part of Python standard library

Main function is parse ()

Returns the tree structure

Attributes returned as Python dictionary

Element can be treated as a list