



FULL SAIL
UNIVERSITY

XML and Python

Using Extensible Information

Design Patterns for Web Programming
Web Design & Development Bachelor of Science Degree



What is XML?

Its meta!

- eXtensible Markup Language
- Meta Language
 - Used to create languages like XHTML
- Organized storage
- Platform independent



What XML looks like

Trust me.. you've seen it's like before.

- Very similar to HTML
- Has a root element
- Has tag pairs
- Tags must be nested
- Tags can have attributes
- Tag names are arbitrary



What XML looks like

Root it out!

Has a root element:

```
<root>
```

```
</root>
```

```
<songs>
```

```
</songs>
```



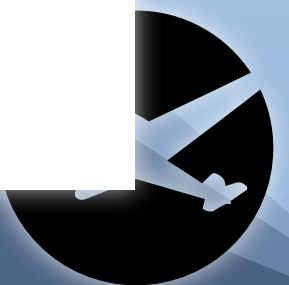
What XML looks like

Root it out!

Has elements within root to store values:

```
<root>
  <element>value</element>
  <element>value</element>
  <element>value</element>
  <element>value</element>
</root>
```

```
<songs>
  <song>In the Flesh</song>
  <song>Another Brick in the
Wall</song>
  <song>Empty Spaces</song>
  <song>Comfortably Numb</
song>
</songs>
```





What XML looks like

Root it out!

Can nest elements within elements:

```
<root>
  <element>
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
  <element>
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
  <element>
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
  <element>
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
</root>
```

```
<songs>
  <song>
    <title>In the Flesh</title>
    <length>3:16</length>
  </song>
  <song>
    <title>Another Brick in the
    Wall</title>
    <length>3:59</length>
  </song>
  <song>
    <title>Empty Spaces</title>
  </song>
  <song>
    <title>Comfortably Numb</title>
  </song>
</songs>
```



What XML looks like

Root it out!

Elements can have attributes:

```
<root>
  <element attribute="value">
    <sub>value</sub>
    <nsb>value</nsb>
  </element>
  <element>
    <sub>value</sub>
    <nsb>value</nsb>
  </element>
  <element>
    <sub>value</sub>
    <nsb>value</nsb>
  </element>
  <element>
    <sub>value</sub>
    <nsb>value</nsb>
  </element>
</root>
```

```
<songs>
  <song writer="Roger Waters">
    <title>In the Flesh</title>
    <length>3:16</length>
  </song>
  <song>
    <title>Another Brick in the
    Wall</title>
    <length>3:59</length>
  </song>
  <song>
    <title>Empty Spaces</title>
  </song>
  <song>
    <title>Comfortably Numb</title>
  </song>
</songs>
```



Namespaces

Differentiating between different tags

- Namespaces allow for additional differentiation between tags
- `<table>` and `<table>` might mean different things in different contexts
- Solution:
 - `<setup:table>` `<data:table>`
- Namespace link





What XML looks like

Space for more names

Adding Namespaces

```
<root xmlns:new="namespace link">
  <element attribute="value">
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
  <element>
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
  <element>
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
  <element>
    <sub>value</sub>
    <nsub>value</nsub>
  </element>
  <new:sub>value</sub>
</root>
```

```
<songs s:xmlns="musicplayer.com/xmlns">
  <title>Song List</title>
  <song writer="Roger Waters">
    <s:title>In the Flesh</title>
    <length>3:16</length>
  </song>
  <song>
    <s:title>Another Brick in the
    Wall</title>
    <length>3:59</length>
  </song>
  <song>
    <s:title>Empty Spaces</title>
  </song>
  <song>
    <s:title>Comfortably Numb</title>
  </song>
</songs>
```



Traversing XML in Python

Turning strings of XML into Python Objects

- Many libraries out there for doing this:
 - etree
 - xml.dom.minidom
 - lxml



Using minidom

Turning strings of XML into Python Objects

● Get XML from file:

- `xml.dom.minidom.parse()`

● or from string

- `xml.dom.minidom.parseString()`

● Getting element

- `xml.documentElement.tagName`



Minidom cont'd

Is it cold in here, or is it just me?

- Accessing attributes:
 - `t.attributes['name'].value`
- Accessing values of elements
 - `xmlDoc.getElementsByTagName('name').nodeValue`
 - `xmlDoc.getElementsByTagName('name')[0].firstChild.nodeValue`



Using etree

Turning strings of XML into Python Objects

○ Get XML from file:

- `ET.parse()` or `ET.fromstring()`

○ Getting element

- `root.tag` or `findall()`

○ Getting attribute

- `root.attrib`

○ Getting root

- `xml.getroot()`

