Week 6 – XML Demo

Here is a simple example of an xml file that might come from an RSS feed. The first line is an XML declaration that contains information about the xml version and the character encoding. The second line defines the root element and gives the name space and language as attributes.

```
<?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>

Every XML document has one root tag, and the structure of the data is given by a set of elements that consist of hierarchical start and end tags of the form:

<foo> </foo>

or if the element's tag has no actual content (perhaps only attributes), it can combine the start and end:

<foo/>

One of the uses of XML is for documents returned by RSS feeds. As an example, I used the RSS feed from the BBC news. Our goal is to retrieve information about current news stories from the XML document returned from this URL.

```
>>> 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'

ElementTree

The ElementTree package is part of the Python standard library. Its main function is the parse() function, which wants to take a file or "file-like object" and return the tree structure. Since we have a string, we can turn it into a stream, which is like a file in that there is a read() function that gives its contents.

```
>>> import xml.etree.ElementTree as etree
>>> import io
>>> xmlfile = io.StringIO(xmlstring)
>>> tree = etree.parse(xmlfile)
>>> type(tree)
<class 'xml.etree.ElementTree.ElementTree'>
```

What we have is the entire tree structure of the document. We can access the document structure by getting the root node, or element, and then traversing the tree by looking at each element and then getting its children.

```
>>> root = tree.getroot()
```

```
>>> type(root)
<class 'xml.etree.ElementTree.Element'>
```

From each element, you can get the tag, the attributes, and the content as follows:

```
>>> root.tag
'rss'
>>> root.attrib
{'version': '2.0'}
>>> root.text
'  \n  '
```

Note that the **attributes are returned as a Python dictionary,** where each tag name is a key in the dictionary and the value is the dictionary value. The contents given by the 'text' is a string, in this case consisting of some spaces and a line feed.

Each **element can also be treated as a list by iterating over it, and that gives the list of children.** As a list, the root element only has one child:

```
>>> len(root)
1
>>> for child in root:
...     print(child)
<Element 'channel' at 0x10175f5e8>
```

Now we can start traversing the tree by getting the child node, which is the first and only element of the root list. We can look at its information.

```
>>> firstchild = root[0]
>>> type(firstchild)
<class 'xml.etree.ElementTree.Element'>
>>> firstchild.tag
'channel'
>>> firstchild.attrib
{}
>>> firstchild.text
' \n    '
>>> len(firstchild)
66
```

We can continue this process by looking at the first of these children:

```
>>> firstgrandchild = firstchild[0]
>>> firstgrandchild.tag
'title'
>>> firstgrandchild.attrib
{}
>>> firstgrandchild.text
'BBC News - Home'
```

This is the title tag of the channel tag. But looking at the file, we can see that what we want are not all of the children of the channel tag but all the ones with the tag 'item'. So instead of traversing the tree, we can use the findall function to get all of the item tags that are direct children of the channel tag. We'll look at the first of these:

```
>>> itemlist = firstchild.findall('item')
```

```
>>> len(itemlist)
45
>>> firstitem = itemlist[0]
>>> firstitem.attrib
{}
>>> firstitem.text
' \n     '
>>> len(firstitem)
6
>>> for element in firstitem:
    print(element.tag, element.attrib, element.text)

title {} Trump signs new travel ban directive
description {} Iraq is removed from the list of countries in
the revised measure, which takes effect on 16 March.
link {} http://www.bbc.co.uk/news/world-us-canada-
39183153
guid {'isPermaLink': 'true'}
http://www.bbc.co.uk/news/world-us-canada-39183153
pubDate {} Mon, 06 Mar 2017 21:30:30 GMT
{http://search.yahoo.com/mrss/}thumbnail {'height': '549',
'width': '976', 'url':
'http://c.files.bbci.co.uk/12FEE/production/_94960877_c6p
7ywcxeaas38r.jpg'} None
```

So now we see what we need. We can use each of the elements corresponding to an item tag as a list, where the first child on the list is the title, the second is the description, and so on.

Although we don't need these in our program, another useful function is the find() function, which returns the first item that matches the tag.

```
>>> firstitem = firstchild.find('item')
>>> firstitem
<Element 'item' at 0x10176f048>
```

And with the findall function, there is additional notation, called the query language, in which you can give path information to guide the search. If we want to search the entire subtree of all the children, we add '//' to our search tag. Here is how to find all the title tags in the entire document:

```
>>> alltitles = tree.findall('.//title')
>>> len(alltitles)
47
>>> for title in alltitles[:6]:
...     print(title.text)
...
BBC News - Home
BBC News - Home
Trump signs new travel ban directive
Vauxhall deal prompts cautious optimism from UK government
US 'may split families that cross border'
'Stark' photos show bus station drug use
```

In addition to the capabilities of accessing information in XML documents, the Python packages such as ElementTree allow you to create new XML documents.