Manipulating XML

All Versions

Improvements requested:

Examples

Opening and reading using an ElementTree

Import the ElementTree object, open the relevant .xml file and get the root tag:

```
import xml.etree.ElementTree as ET
tree = ET.parse("yourXMLfile.xml")
root = tree.getroot()
```

There are a few ways to search through the tree. First is by iteration:

```
for child in root:
    print(child.tag, child.attrib)
```

Otherwise you can reference specific locations like a list:

```
print(root[0][1].text)
```

To search for specific tags by name, use the `.find` or `.findall` :

```
print(root.findall("myTag"))
print(root[0].find("myOtherTag"))
```

Create and Build XML Documents

Import Element Tree module

```
import xml.etree.ElementTree as ET
```

Element() function is used to create XML elements

```
p=ET.Element('parent')
```

SubElement() function used to create sub-elements to a give element

```
c = ET.SubElement(p, 'child1')
```

dump() function is used to dump xml elements.

```
ET.dump(p)
# Output will be like this
#<parent><child1 /></parent>
```

If you want to save to a file create a xml tree with ElementTree() function and to save to a file use write() method

```
tree = ET.ElementTree(p)
tree.write("output.xml")
```

Comment() function is used to insert comments in xml file.

```
comment = ET.Comment('user comment')
p.append(comment) #this comment will be appended to parent element
```

Modifying an XML File

Import Element Tree module and open xml file, get an xml element

```
import xml.etree.ElementTree as ET
tree = ET.parse('sample.xml')
```

```
root=tree.getroot()
element = root[0] #get first child of root element
```

Element object can be manipulated by changing its fields, adding and modifying attributes, adding and removing children

```
element.set('attribute_name', 'attribute_value') #set the attribute to xml element
element.text="string_text"
```

If you want to remove an element use Element.remove() method

```
root.remove(element)
```

ElementTree.write() method used to output xml object to xml files.

```
tree.write('output.xml')
```

Opening and reading large XML files using iterparse (incremental parsing)

Sometimes we don't want to load the entire XML file in order to get the information we need. In these instances, being able to incrementally load the relevant sections and then delete them when we are finished is useful. With the iterparse function you can edit the element tree that is stored while parsing the XML.

Import the ElementTree object:

```
import xml.etree.ElementTree as ET
```

Open the .xml file and iterate over all the elements:

```
for event, elem in ET.iterparse("yourXMLfile.xml"):
    ... do something ...
```

Alternatively, we can only look for specific events, such as start/end tags or namespaces. If this option is omitted (as above), only "end" events are returned:

```
events=("start", "end", "start-ns", "end-ns")
for event, elem in ET.iterparse("yourXMLfile.xml", events=events):
    ... do something ...
```

Here is the complete example showing how to clear elements from the in-memory tree when we are finished with them:

```
for event, elem in ET.iterparse("yourXMLfile.xml", events=("start","end")):
    if elem.tag == "record_tag" and event == "end":
        print elem.text
        elem.clear()
    ... do something else ...
```

Searching the XML with XPath

Starting with version 2.7 ElementTree has a better support for XPath queries. XPath is a syntax to enable you to navigate through an xml like SQL is used to search through a database. Both find and findall functions support XPath. The xml below will be used for this example

```
 <Catalog>
    <Books>
        <Book id="1" price="7.95">
            <Title>Do Androids Dream of Electric Sheep?</Title>
            <Author>Philip K. Dick</Author>
        </Book>
        <Book id="5" price="5.95">
            <Title>The Colour of Magic</Title>
            <Author>Terry Pratchett</Author>
        </Book>
        <Book id="7" price="6.95">
            <Title>The Eye of The World</Title>
            <Author>Robert Jordan</Author>
        </Book>
    </Books>
</Catalog>
```

Searching for all books:

```
import xml.etree.cElementTree as ET
```

```
tree = ET.parse('sample.xml')
tree.findall('Books/Book')
```

Searching for the book with title = 'The Colour of Magic':

```
tree.find("Books/Book[Title='The Colour of Magic']")
# always use '' in the right side of the comparison
```

Searching for the book with id = 5:

```
tree.find("Books/Book[@id='5']")
# searches with xml attributes must have '@' before the name
```

Search for the second book:

```
tree.find("Books/Book[2]")
# indexes starts at 1, not 0
```

Search for the last book:

```
tree.find("Books/Book[last()]")
# 'last' is the only xpath function allowed in ElementTree
```

Search for all authors:

```
tree.findall(".//Author")
#searches with // must use a relative path
```

Syntax

Parameters

Remarks

Not all elements of the XML input will end up as elements of the parsed tree. Currently, this module skips over any XML comments, processing instructions, and document type declarations in the input. Nevertheless, trees built using this module's API rather than parsing from XML text can have comments and processing instructions in them; they will be included when generating XML output.