XML and python

Corpus Linguistics

HAP/LAP

lxml.etree

Module to work with XML within python.
Can do many things:

- Parse files.
- Analyze structure.
- Xpath queries.
- Create XML documents.

Hello world

- `lxml.etree` is widely used
- 'Hello world':

```
from lxml import etree # load module

parser = etree.XMLParser(remove_blank_text=True) # create parser
tree = etree.parse("input.xml", parser) # parse document intro 'tree'
root = tree.getroot() # document element

print(etree.tostring(root, pretty_print=True,
                     encoding="unicode")) # print tree
```

Parser

Create a parse to analyze input documents

```
from lxml import etree # load module

parser = etree.XMLParser(remove_blank_text=True) # discard whitespace nodes
tree = etree.parse(filename, parser)
```

Many options:

- `dtd_validation`: validate against a DTD
- `load_dtd`: use DTD for parsing
- `recover`: try hard to parse through broken XML
- `remove_blank_text`: discard blank text nodes
- `remove_comments`: discard comments
- `remove_pis`: discard processing instructions
- `strip_cdata` - replace CDATA sections by normal text content (on by default)

Elements

- Create elements with `etree.Element` or `etree.SubElement`

```
from lxml import etree # load module

sent = etree.Element("sent") # create element
print(sent.tag) # prints 'sent'
child1 = etree.SubElement(sent, "wf") # add one child
child2 = etree.SubElement(sent, "wf") # add another child
print(etree.tostring(sent, pretty_print=True)) # print element as a string

# test if a variable is an element

if etree.iselement(sent):
    print('Yes')
```

- Obtain elements from documents

```
parser = etree.XMLParser() # create parser
tree = etree.parse(filename, parser)
root = tree.getroot() # document element
```

Elements: accessing children

- Use `len` to know number of children
- get children by index

```
print(len(sent)) # 2
sent[0] # first children
sent[1] # second children
sent.append(c) # append element c as child of sent
del sent[0] # remove first element
```

- use iterators for traversing tree

```
# traverse children
for child in sent:
    print(child.tag)
# given an element 'child'
child.getparent() # parent element
child.getprevious() # previous sibling
child.getnext() # next sibling
...
```

Textual elements

- textual elements

```
from lxml import etree # load module
sent = etree.Element("sent")
sent.text = "Some text" # set textual element as child
print (sent.text) # prints 'Some text'
print(etree.tostring(sent, pretty_print=True)) # <sent>Some text</sent>
```

More traversal

- Given an element object (*etree.Element*)
  - `find(tag_or_path)`: Finds the **first** matching subelement, by tag name or path.
  - `findtext(string)`: Finds text for the **first** matching subelement, by tag name or path.
  - `findall(tag_or_path)`: Returns **all** matching children, by tag name or path.

```
from lxml import etree # load module
node = etree.fromstring("<a><b>bum</b><b>ear</b><c/></a>")
node.findtext("b") # returns 'bum'
node.findtext("c") # returns ''
```

Attributes

- Attributes
  - `get(name)`: get value of attribute 'name'
  - `set(name, value)`: set attribute 'name' with value 'value'
  - `keys()`: return attribute keys as a list
  - `items()`: return a list of `(name, value)` tuples

```
from lxml import etree # load module

 # create element with 'pos' and 'subc' attributes
word = etree.Element("wf", pos="N", subc="NNP")
word.get("pos") #  'N'
word.get("lemma") #  None
word.set("lemma", "public") # create attribute 'lemma'

sorted_keys = sorted(word.keys()) # ['lemma', 'pos', 'subc']

for name,value in word.items():
    print(f"{name} : {value}")
```

XPath queries

- Use
  - `document.xpath(`*expression*`)`
  - `element.xpath(`*expression*`)`
- Result is a list (nodeset).

Example: XPath queries

Example

Get all tokens from s3aw.xml

```
#!/usr/bin/python3
from lxml import etree
parser = etree.XMLParser(remove_blank_text=True) # discard whitespace nodes
tree = etree.parse("s3aw.xml", parser) # parse document
# using the tree root as current node, execute the XPath query "//wf"
for wf_elem in tree.xpath("//wf"):
    text = wf_elem.text
    print(text)
```

Example: XPath queries

Example

Get all tokens from `s3aw.xml`

Another option:

```
#!/usr/bin/python3
from lxml import etree
parser = etree.XMLParser(remove_blank_text=True) # discard whitespace nodes
tree = etree.parse("s3aw.xml", parser) # parse document
for wftext_elem in tree.xpath("//wf/text()"):
    print(wftext_elem)
```

Example: XPath queries

Example

Get all noun forms from `s3aw.xml`

Another option:

```
#!/usr/bin/python3
from lxml import etree
parser = etree.XMLParser(remove_blank_text=True) # discard whitespace nodes
tree = etree.parse("s3aw.xml", parser) # parse document
for wf_elem in tree.xpath("//token[@pos='n']/wf"):
    print(wf_elem.text)
```

Example: XPath queries

Example

Get all definitions from `EH.xml`

```
#!/usr/bin/python3
from lxml import etree
parser = etree.XMLParser(remove_blank_text=True) # discard whitespace nodes
tree = etree.parse("EH.xml", parser) # parse document
# using the tree root as current node, execute the XPath query "//def"
for def_elem in tree.xpath("//def"):
    print(def_elem.text)
```

Example: nested XPath queries

- First iterate over all entries
- In a second loop, do something for each entry
  - The second xpath expression is executed under the `entry` variable

```
#!/usr/bin/python3

import sys
from lxml import etree

parser = etree.XMLParser(remove_blank_text=True) # discard whitespace nodes
tree = etree.parse("EH.xml", parser) # parse document
# using the tree root as current node, execute the XPath query "//entry"
for entry in tree.xpath("//entry"):
    # using entry as current node, execute query ".//def"
    for def in entry.xpath(".//def")
        print (def.text)
```

Example: XPath queries

Example

Get tokens and ids (tabulated output) from `s3aw.xml`

- given a token element (in variable `token_elem`), how to access the <wf> element?
  - `wf = token_elem.xpath("wf")[0]`
  - there are more options (`find`, `findtext`, etc)

```
#!/usr/bin/python3
from lxml import etree
parser = etree.XMLParser(remove_blank_text=True) # discard whitespace nodes
tree = etree.parse("s3aw.xml", parser) # parse document
for token_elem in tree.xpath("//token"):
    id = token_elem.get("id")
    wf = token_elem.xpath("wf")[0] # first element of nodeset
    print(id, wf.text)
```

Creating documents

- Use `etree.ElementTree` for creating documents from scratch
  - Needs root element as parameter

```
import sys # for stdout
from lxml import etree # load module
page = etree.Element("html") # create root element
doc = etree.ElementTree(page) # create document
# use etree.SubElement() to attach children to elements
header = etree.SubElement(page, "head")
body = etree.SubElement(page, "body")
# new element below header element
title = etree.SubElement(header, "title")
title.text = "The page title" # assign textual content
# create elements with attributes (<link rel='stylesheet' type='text/css'>)
link = etree.SubElement(header, "link", rel="stylesheet", type = "text/css")
# write document
print(etree.tostring(doc, encoding="unicode"))
```