Using lxml to read XML data

We will first need the *lxml* python package to be installed on your machine.

Under Windows, launch the command window and type the command *pip install lxml*.

Under OS X, in a terminal window and type the command *pip3 install lxml*

lxml provides us with various methods of dealing with XML data known as APIs (Application Programming Interfaces). The first way is the ElementTree API, which enables us to easily access XML data in a tree-like structure.

As with any other Python packages, you need to issue an import command to load a package:

In [ ]:

```
from lxml import etree
```

In order to load an XML file and to represent it as a tree in computer memory, you need to parse the XML file. The etree.parse() function parses the XML file that is passed in as a parameter. We first load in the file that you created in question 2.

In [ ]:

```
xmltree = etree.parse("royal2.xml")
```

The *parse()* function returns an XML *ElementTree* object, which represents the whole XML tree. Each node in the tree is translated into an *Element* object .

Use *getroot()* function of an *ElementTree* object to get the root element of the XML tree. You can print out the XML tag of an element using *tag* property.

In [ ]:

```
root = xmltree.getroot()
print (root.tag)
```

Traversing the XML Tree

The following sections describe various methods for traversing the XML tree

To obtain a list all of the children of an element, you can iterate over the XML *Element* itself:

In [ ]:

```
for e in root:
   print (e.tag)
```

You can use indexing to access the children of an element:

In [ ]:

```
oldest_prince = root[0]
print (oldest_prince.get("title"))
```

The *find()* method returns only the first matching child.

In [ ]:

```
the_first_child_with_prince_tag = root.find("prince")
print (the_first_child_with_prince_tag.get('title'))
```

The *iterchildren()* function allows you to iterate over children with a particular tag:

In [ ]:

```
for child in root.iterchildren(tag="prince"):
    print (child.get('title'))
```

There is also a *iterdescendants()* function to iterate all descendants of a particular node.

Exercise 5a)

Using the *royal2.xml*:

i) Write Python code to get the title property of queen's grandsons.

ii) Write Python code to get the full title of the only princess in the family tree.

Accessing XML attributes

You can access the XML attributes of an element using the *get()* method or *attrib* properties of an element.

In [ ]:

```
print (root.attrib)
print (root.get("title"))
```

Accessing XML text

Let's now use another sample of XML data. Consider the file book.xml Salinger, J. D. English 1951-07-16 Little, Brown and Company 0-316-76953-3 A story about a few important days in the life of Holden Caulfield

This XML looks different to the *royal2.xml* in that it has some text content within each element. To access the text content of an element (text between start and end tag), use *text* properties of that element

In [ ]:

```
from lxml import etree
xmltree = etree.parse('book.xml')
root = xmltree.getroot()
for child in root:
    print (child.tag + ": " + child.text)
```

Building XML data

Let's go back to the *book.xml* example above. As usual, use *lxml* library to parse the XML and get the root of the tree:

In [ ]:

```
from lxml import etree
xmltree = etree.parse('book.xml')
root = xmltree.getroot()
```

To create a new XML element, use *etree.Element()* function:

In [ ]:

```
new_element = etree.Element('genre')
new_element.text = 'Novel'
root.append(new_element)
etree.tostring(root[-1],pretty_print=True,encoding='unicode')   # the last ele
ment, the newly appended element
```

Tips: You can create a totally a new XML tree by constructing the root element:

In [ ]:

```
root = etree.Element('book')
```

You can also create new element using *SubElement()* function:

In [ ]:

```
new_element = etree.SubElement(root, "price")
new_element.text = '23.95'
for e in root: # check whether the new element is added
    print (e.tag)
```

Use *insert()* to insert a new element at a specific location:

In [ ]:

```
root.insert(1,etree.Element("country"))
root[1].text = "United States"
etree.tostring(root[1],pretty_print=True,encoding='unicode')
```

Serialising XML data (printing as web content or writing into a file)

You can get the whole XML string by calling *etree.tostring()* with the root of the tree as the first parameter:

In [ ]:

```
output = etree.tostring(root, pretty_print=True, encoding="UTF-8")
for e in root:
   print (e.tag)
```

In [ ]:

```
open('output.xml','wb').write(output)
```

Exercise 5b)

Write Python code to load in the file "book.xml", change the ISBN to "Unknown" and then write out the file to "book-new.xml"

This is the end of the notebook. Now return to question 6) in the exercises sheet.