Parsing XML with lxml.objectify

The lxml module has a module called **objectify** that can turn XML documents into Python objects. I find "objectified" XML documents very easy to work with and I hope you will too. You may need to jump through a hoop or two to install it as **pip** doesn't work with lxml on Windows. Be sure to go to the Python Package index and look for a version that's been made for your version of Python. Also note that the latest pre-built installer for lxml only supports Python 3.2 (at the time of writing), so if you have a newer version of Python, you may have some difficulty getting lxml installed for your version.

Anyway, once you have it installed, we can start going over this wonderful piece of XML again:

```
<?xml version="1.0" ?>
<zAppointments reminder="15">
    <appointment>
        <begin>1181251680</begin>
        <uid>040000008200E000</uid>
        <alarmTime>1181572063</alarmTime>
        <state></state>
        <location></location>
        <duration>1800</duration>
        <subject>Bring pizza home</subject>
    </appointment>
    <appointment>
        <begin>1234360800</begin>
        <duration>1800</duration>
        <subject>Check MS Office website for updates</subject>
        <location></location>
        <uid>604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800</uid>
        <state>dismissed</state>
  </appointment>
</zAppointments>
```

Now we need to write some code that can parse and modify the XML. Let's take a look at this little demo that shows a bunch of the neat abilities that objectify provides.

```
from lxml import etree, objectify

def parseXML(xmlFile):

    """Parse the XML file"""
    with open(xmlFile) as f:
        xml = f.read()

    root = objectify.fromstring(xml)

    # returns attributes in element node as dict
    attrib = root.attrib

    # how to extract element data
    begin = root.appointment.begin
    uid = root.appointment.uid

    # loop over elements and print their tags and text
    for appt in root.getchildren():
        for e in appt.getchildren():
            print("%s => %s" % (e.tag, e.text))
        print()

    # how to change an element's text
    root.appointment.begin = "something else"
    print(root.appointment.begin)

    # how to add a new element
    root.appointment.new_element = "new data"

    # remove the py:pytype stuff
    objectify.deannotate(root)
    etree.cleanup_namespaces(root)
    obj_xml = etree.tostring(root, pretty_print=True)
    print(obj_xml)

    # save your xml
    with open("new.xml", "w") as f:
        f.write(obj_xml)

if __name__ == "__main__":
    f = r'path\to\sample.xml'
    parseXML(f)
```

The code is pretty well commented, but we'll spend a little time going over it anyway. First we pass it our sample XML file and **objectify** it. If you want to get access to a tag's attributes, use the **attrib** property. It will return a dictionary of the attributes of the tag. To get to sub-tag elements, you just use dot notation. As you can see, to get to the **begin** tag's value, we can just do something like this:

```
begin = root.appointment.begin
```

One thing to be aware of is if the value happens to have leading zeroes, the

returned value may have them truncated. If that is important to you, then you should use the following syntax instead:

```
begin = root.appointment.begin.text
```

If you need to iterate over the children elements, you can use the **iterchildren** method. You may have to use a nested for loop structure to get everything. Changing an element's value is as simple as just assigning it a new value.

```
root.appointment.new_element = "new data"
```

Now we're ready to learn how to create XML using **lxml.objectify**.