Parsing XML

WE'LL COVER THE FOLLOWING

Python can parse XML documents in several ways. It has traditional DOM and SAX parsers, but I will focus on a different library called ElementTree.

```
import xml.etree.ElementTree as etree    #①
tree = etree.parse('feed.xml')           #②
root = tree.getroot()                    #③
root                                     #④
#<Element {http://www.w3.org/2005/Atom}feed at cd1eb0>
```

① The ElementTree library is part of the Python standard library, in `xml.etree.ElementTree`.

② The primary entry point for the ElementTree library is the `parse()` function, which can take a filename or a file-like object. This function parses the entire document at once. If memory is tight, there are ways to parse an xml document incrementally instead.

③ The `parse()` function returns an object which represents the entire document. This is not the root element. To get a reference to the root element, call the `getroot()` method.

④ As expected, the root element is the `feed` element in the `http://www.w3.org/2005/Atom` namespace. The string representation of this object reinforces an important point: an xml element is a combination of its namespace and its tag name (also called the *local name*). Every element in this

namespace and its tag name (also called the *local name*). Every element in this

document is in the Atom namespace, so the root element is represented as `{http://www.w3.org/2005/Atom}feed`.

> *ElementTree represents xml elements as **{namespace}localname**. You'll see and use this format in multiple places in the ElementTree api.*

Elements Are Lists

In the ElementTree API, an element acts like a list. The items of the list are the element's children.

```
# continued from the previous example
import xml.etree.ElementTree as etree
tree = etree.parse('feed.xml')
root = tree.getroot()

print (root.tag )                          #①
#'{http://www.w3.org/2005/Atom}feed'

print (len(root))                          #②
#8

for child in root:                         #③
   print(child)                            #④

#<Element {http://www.w3.org/2005/Atom}title at e2b5d0>
#<Element {http://www.w3.org/2005/Atom}subtitle at e2b4e0>
#<Element {http://www.w3.org/2005/Atom}id at e2b6c0>
#<Element {http://www.w3.org/2005/Atom}updated at e2b6f0>
#<Element {http://www.w3.org/2005/Atom}link at e2b4b0>
#<Element {http://www.w3.org/2005/Atom}entry at e2b720>
#<Element {http://www.w3.org/2005/Atom}entry at e2b510>
#<Element {http://www.w3.org/2005/Atom}entry at e2b750>
```

① Continuing from the previous example, the root element is `{http://www.w3.org/2005/Atom}feed`.

② The "length" of the root element is the number of child elements.

③ You can use the element itself as an iterator to loop through all of its child elements.

④ As you can see from the output, there are indeed 8 child elements: all of the feed-level metadata (`title`, `subtitle`, `id`, `updated`, and `link`) followed by the three `entry` elements.

You may have guessed this already, but I want to point it out explicitly: the list of child elements only includes *direct* children. Each of the entry elements contain their own children, but those are not included in the list. They would be included in the list of each entry's children, but they are not included in the list of the feed's children. There are ways to find elements no matter how deeply nested they are; we'll look at two such ways later in this chapter.

Attributes Are Dictonaries

XML isn't just a collection of elements; each element can also have its own set of attributes. Once you have a reference to a specific element, you can easily get its attributes as a Python dictionary.

```
# continuing from the previous example
import xml.etree.ElementTree as etree         #①
tree = etree.parse('feed.xml')
root = tree.getroot()

print (root.attrib)                           #①
#{'{http://www.w3.org/XML/1998/namespace}lang': 'en'}

print (root[4])                               #②
#<Element {http://www.w3.org/2005/Atom}link at e181b0>

print (root[4].attrib)                        #③
#{'href': 'http://diveintomark.org/', 'type': 'text/html', 'rel': 'alternate'}

print (root[3] )                              #④
#<Element {http://www.w3.org/2005/Atom}updated at e2b4e0>

print (root[3].attrib)                        #⑤
#{}
```

① The `attrib` property is a dictionary of the element's attributes. The original markup here was `<feed xmlns='http://www.w3.org/2005/Atom' xml:lang='en'>`. The `xml:` prefix refers to a built-in namespace that every XML document can use without declaring it.

② The fifth child — `[4]` in a 0-based list — is the `link` element.

③ The `link` element has three attributes: `href`, `type`, and `rel`.

④ The fourth child — `[3]` in a 0-based list — is the `updated` element.

⑤ The `updated` element has no attributes, so its `.attrib` is just an empty dictionary.