13.1: eXtensible Markup Language - XML

XML looks very similar to HTML, but XML is more structured than HTML. Here is a sample of an XML document:

```
<person>
    <name>Chuck</name>
    <phone type="intl">
        +1 734 303 4456
    </phone>
    <email hide="yes"/>
</person>
```

Often it is helpful to think of an XML document as a tree structure where there is a top tag person and other tags such as phone are drawn as children of their parent nodes.

![A Tree Representation of XML](https://eng.libretexts.org/Bookshelves/Computer_Science/Book%3A_Python_for_Everybody_(Severance)/13%3A_Python_and_Web_Services/13.1%3A_eXtensible_Markup_Language_-_XML)
Parsing XML

Here is a simple application that parses some XML and extracts some data elements from the XML:

Code \(\PageIndex{1}\) (Python):

```python3
import xml.etree.ElementTree as ET

data = '''
<person>
    <name>Chuck</name>
    <phone type="intl">
        +1 734 303 4456
    </phone>
    <email hide="yes"/>
</person>'''

tree = ET.fromstring(data)
print('Name:', tree.find('name').text)
print('Attr:', tree.find('email').get('hide'))

# Code: http://www.py4e.com/code3/xml1.py
```

Calling `fromstring` converts the string representation of the XML into a "tree" of XML nodes. When the XML is in a tree, we have a series of methods we can call to extract portions of data from the XML.

The `find` function searches through the XML tree and retrieves a node that matches the specified tag. Each node can have some text, some attributes (like hide), and some "child" nodes. Each node can be the top of a tree of nodes.

| Name: Chuck | Attr: yes |

Using an XML parser such as `ElementTree` has the advantage that while the XML in this example is quite simple, it turns out there are many rules regarding valid XML and using `ElementTree` allows us to extract data from XML without worrying about the rules of XML syntax.