Reading XML with xml.etree.ElementTree

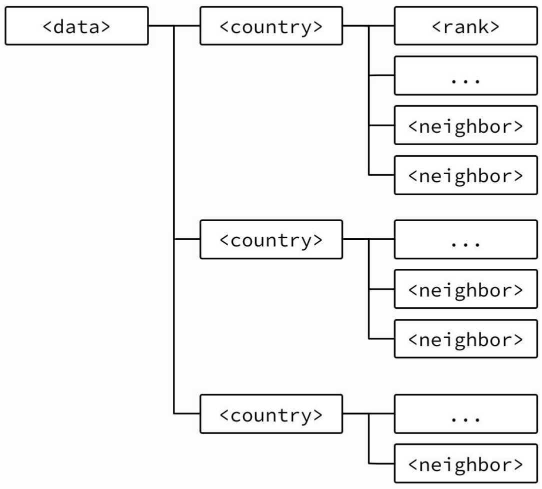
Jessica's notes from Sep 21, 2018 on
https://docs.python.org/3.6/library/xml.etree.elementtree.html

Python's built-in `xml.etree.ElementTree` module can read XML documents into an `ElementTree` object.
An `ElementTree` contains a recursive data structure of text, dictionaries, and lists.

XML document

```xml
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

Data structure



Downloading and parsing XML

Some online APIs format data as XML. The `ElementTree` module can read XML from these APIs.

For example, `ElementTree.fromstring` can parse a webpage's XML that was downloaded using the `requests` library (shown to the right).

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

import requests


response = requests.get(...)
root = ET.fromstring(response.content)
```

Using Elements

`Element` is the primary class used to navigate the parsed XML document.

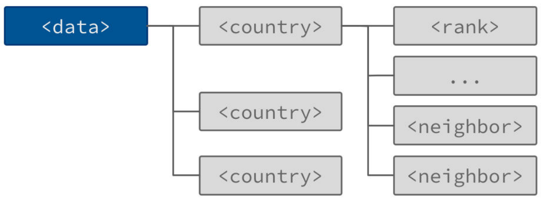
`Elements` have a few attributes and methods to access the data contained in the data structure. Some useful attributes and methods for accessing data are shown to the right.

Below are examples of the values these attributes have for the XML document shown above.

`Element e`

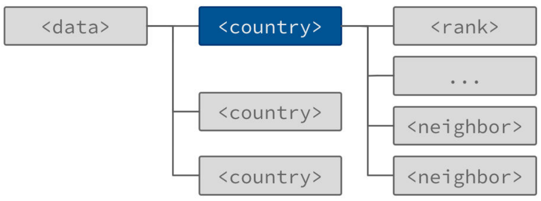
Attribute	≈XML	Type
`e.tag`	`<country>`	`string`
`e.attrib`	`name="Panama"`	`{string: string}`
`e.text`	`<>text</>`	`string`
`e.tail`	`<></>tail`	`string`
`list(e)`	`<><></></>`	`[element]`

`root = ET.fromstring(...)`



`root.tag`	`'data'`
`root.attrib`	`{}`
`root.text`	`(whitespace)`
`root.tail`	`(whitespace)`
`list(root)`	`[<Element 'country'>, <Element 'country'>, <Element 'country'>]`

`child = root[0]`



`child.tag`	`'country'`
`child.attrib`	`{'name': 'Liechtenstein'}`
`child.text`	`(whitespace)`
`child.tail`	`(whitespace)`
`list(child)`	`[<Element 'rank'>, <Element 'year'>, <Element 'gdppc'>, <Element 'neighbor'>, <Element 'neighbor'>]`

Elements

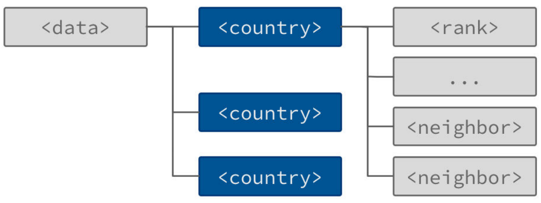
`Elements` also have methods to search the structure to find interesting `Elements`.

Below are a few examples of the values these methods return for the above XML document and `Elements` shown.

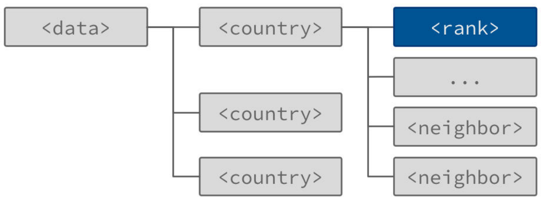
See documentation for full function signature and more methods.

`e.findall(match)`	List `Elements` that are direct children of `e` with tag or path `match`.
`e.find(match)`	First `Element` that is a direct child of `e` with tag or path `match`.
`e.iter(tag)`	Recursively iterate over the sub-tree of `e` filtering for tag `tag`.

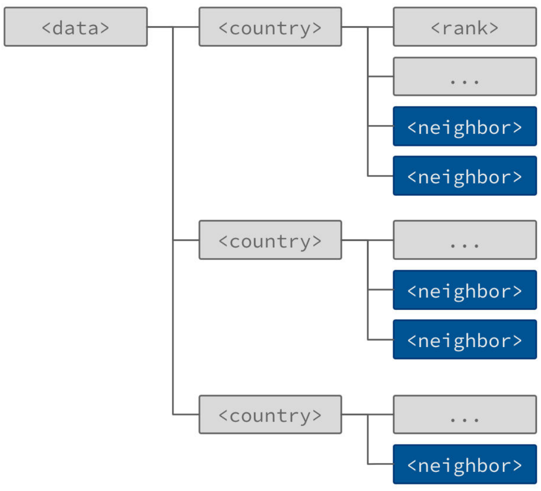
`root.findall('country')`



`child.find('neighbor')`
or
`root.find('country/neighbor')`



`root.iter('neighbor')`
or*
`root.findall('country/neighbor')`



*In this example, these return the same `Elements` but work differently. `root.iter` returns any `Element` with the `neighbor` tag. `root.iter` only returns `Elements` that are grandchildren of `root` and children of `Elements` with tag `country`.