34-structured

April 29, 2016

1 Structured Language Tools

- tools to r/w structured languages

2 JSON

- JSON format very similar to python lists and dicts, and javascript
- JSON used extensively in internet protcols
- docs

In [2]:
```
# convert python to JSON string

import json

data = ['foo', {'bar': ('baz', None, 1.0, 2)}]
js = json.dumps(data)
js
```

Out[2]:
```
'["foo", {"bar": ["baz", null, 1.0, 2]}]'
```

In [3]:
```
# can do 'pretty printing'

print(json.dumps(data, sort_keys=True, indent=4))
```

```
[
    "foo",
    {
        "bar": [
            "baz",
            null,
            1.0,
            2
        ]
    }
]
```

In [4]:
```
# convert JSON back to Python

json.loads(js)
```

Out[4]:
```
['foo', {'bar': ['baz', None, 1.0, 2]}]
```

3 XML parser

- doc

```
In [4]: 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>

        '''

In [5]: import xml.etree.ElementTree as ET

        root = ET.fromstring(xml)

In [6]: root.tag

Out[6]: 'data'

In [7]: for c in root:
            print(c, c.items(), c.find('rank').text)

<Element 'country' at 0x105e387c8> [('name', 'Liechtenstein')] 1
<Element 'country' at 0x1058e69a8> [('name', 'Singapore')] 4
<Element 'country' at 0x1058e6b38> [('name', 'Panama')] 68

In [8]: [a,b,c] = list(root)

In [9]: [a.items(), b.items(), c.items()]

Out[9]: [[('name', 'Liechtenstein')], [('name', 'Singapore')], [('name', 'Panama')]]

In [24]: [a.find('year').text, b.find('neighbor'), c.find('rank').text]

Out[24]: ['2008', <Element 'neighbor' at 0x1058e6ae8>, '68']
```

4 HTML parser

- interesting technique
  - define methods for things you care about
- doc

```
In [7]: from html.parser import HTMLParser

        class MyHTMLParser(HTMLParser):
            def handle_starttag(self, tag, attrs):
                print("Encountered a start tag:", tag)
            def handle_endtag(self, tag):
                print("Encountered an end tag :", tag)
            def handle_data(self, data):
                print("Encountered some data  :", data)

        parser = MyHTMLParser()
        parser.feed('<html><head><title>Test</title></head>'
                    '<body><h1>Parse me!</h1></body></html>')
```

```
Encountered a start tag: html
Encountered a start tag: head
Encountered a start tag: title
Encountered some data  : Test
Encountered an end tag : title
Encountered an end tag : head
Encountered a start tag: body
Encountered a start tag: h1
Encountered some data  : Parse me!
Encountered an end tag : h1
Encountered an end tag : body
Encountered an end tag : html
```

```
In [ ]:
```