XML and JSON in Python

Zdeněk Žabokrtský, Rudolf Rosa

December 5, 2018

Charles Univeristy in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

XML in Python

- the two standard approaches for XML processing are supported in the standard library:
  - `xml.dom.*` – a standard DOM API
  - `xml.sax.*` – a standard SAX API
- but there's `xml.etree.ElementTree` (ET for short)
  - a lightweight Pythonic API
  - supports both DOM-like (but ET faster than DOM) and SAX-like processing (i.e. event-based, streaming i.e. all-in-memory)
  - fast C implementation used by default whenever possible in Python3 (no need for xml.etree.cElementTree as in Python 2)

Credit: The following slides are based on an ElementTree intro by Eli Bendersky.

ET: loading an XML doc

loading from a file:

```
import xml.etree.ElementTree as ET
tree = ET.ElementTree(file='sample.xml')
```

or from a string:

```
root = ET.fromstring('<my><own><xmlcontent/></own></my>')
```

ET: traversing the tree

```
root = tree.getroot()

for child in root:
    print(child.tag, child.attrib, child.text)

for descendant in root.iter():
    ....
```

Time for exercise

write a recursive function `drawxmltree` that visualizes the tree structure of an XML element by space indentation (one element per line, only tag displayed, two-space indentation per level), so that

```
drawxmltree(ET.fromstring('<root><child><grandchild/><grandchild/><grandchild/><child/></root
```

results in

```
root
  child
    grandchild
    grandchild
  child
```

ET: accessing attribute values

```
root = ET.fromstring('<koren id="x15" name="John"/>')
for attr in root.attrib:
    print(attr+"="+root.attrib[attr])
```

ET: simple searching

```
for elem in tree.iter(tag='surname'):
    ....
```

ET: complex searching using XPath

```
for elem in tree.iterfind('*/section/figure[@id="f15"]'):
    ....
```

ET: creating+storing an XML doc

```
root = ET.Element('root')
newelem = ET.SubElement(root, 'data')
ET.dump(root)
```

JSON

- JavaScript Object Notation
- a simple text-oriented format for data exchange between a browser and a server
- inspired by JavaScript object literal syntax, but nowadays used well beyond the JavaScript world
- became one of the most popular data exchange formats in the last years

XML vs. JSON – a first glimpse

```xml
<?xml version="1.0"?>
<book id="123">
  <title>Object Thinking</title>
  <author>David West</author>
  <published>
    <by>Microsoft Press</by>
    <year>2004</year>
  </published>
</book>
```

```json
{
  "id": 123,
  "title": "Object Thinking",
  "author": "David West",
  "published": {
    "by": "Microsoft Press",
    "year": 2004
  }
}
```

JSON – a quick syntax tour

- data – hierarchical structures
- curly braces hold objects
  - name and value separated by colon
  - name-value pairs separated by comma
- square brackets hold arrays
  - values separated by comma
- whitespaces (space, tab, LF, CR) around syntactic elements ignored
- BOM not allowed
- no syntax for comments

JSON – data types

- number
- string
- boolean
- array
- object
- null

JSON in Python

- `json` – JSON API in available the standard library
- API similar to that of `pickle`

`json`: Implicit type conversions

- A JSON object goes to Python `dict`
- a JSON array goes to Python `list`
- a JSON string goes to Python `unicode`
- a JSON number goes to Python `int` or `long`
- a JSON true goes to Python `True`
- etc.

and vice versa.

json: serializing/deserializing

```
import json

named_entity = {"form":"Bob", "type":"firstname", span:[0,1,2]}

serialized = json.dumps(named_entity)

restored = json.loads(serialized)
```

`json`: selected serialization options

There's some space for customizing the serialization (within the limits given by the JSON spec):

- `encoding` – the character encoding (utf-8 by default)
- `indent` – pretty-printing with the specified indent level for object members
- `sort_keys` – output of dictionaries sorted lexicographically by key
- `separator` – tuple `(item_sep, key_sep)`

XML vs. JSON – similarities

- both XML and JSON are frequently used for data interchange
- both formats are human readable (if designed properly)
- both are currently supported by many programming languages

XML vs. JSON – differences

- as usual, we face the trade-off of simplicity against expressiveness
- with some over-simplification: JSON is a lightweight cousin of XML
- JSON is slightly less verbose and simpler (and faster) to parse...
- ..., but currently there's more functionality associated with the XML standard: namespaces, referencing, validations schemes, stylesheet transformations, query languages etc.
- so threre's no clear superiority of one against the other
- your final choice should depend on what you really need (and, of course, on the system context)

XML vs. JSON – can we estimate future from history?

- In 1990s, XML was introduced as a considerably simplified descendant of SGML.
- But 20 years later SGML is still everywhere around, incarnated basically in every web page.
- However, does XML have such a killer app now?

References I