4.6 Introduction to JSON with Python

Introduction to JSON

JSON (JavaScript Object Notation) is a text oriented data representation format which is mainly user for information exchange between web clients and web services/servers. JSON was inspired by JavaScript syntax, but today it is widely used in many application beyond JavaScript.

XML vs JSON

Both XML and JSON are structed data representation formats. Shown below is a comparison of the same data represented using XML and JSON.

```xml
<?xml version = "1.0?">
<motorvehicles>
        <vehicle>
                <registration_no>CBB1456</registration_no>
                <make>Toyota</make>
                <model>Premio</model>
        </vehicle>
        <owner>
                <first_name>Amal</first_name>
                <first_name>Perera</first_name>
                <nic>900324770V</nic>
        </owner>
<motorvehicles>
```

```json
{
  "motorvehicles": {
    "vehicle": {
      "registration_no": "CBB1456",
      "make": "Toyota",
      "model": "Premio"
    },
    "owner": {
      "first_name": "Amal",
      "last_name": "Perera",
      "nic": "900324770V"
    }
  }
}
```

JSON Syntax

- 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, etc.) are around syntactic elements are ignored
- No syntax for comments

JSON Data Types

- Number
- String
- Boolean
- Array
- Object
- Null

JSON with Python

JSON API is available in the Python standard library and it also provides implicit type conversion between JSON objects and Python data types.

- Implicit type conversion
  - JSON object → Python dict

- JSON array → Python list
- JSON string → Python unicode
- JSON number → Python int or long
- JSON true → Python True

JSON Serializing and Deserializing

Serializing refers to the process of converting a text content to another object formation, in this case from text to JSON. Deserializing is the reverse process.

json.dumps() function is used for JSON serializing and json.loads() function is used for deserializing a JSON object. And example is shown below.

```
>>>import json

>>>vehicle_entity = {  "motorvehicles": {      "vehicle": {      "registration_no": "CBB1456",      "make": "

>>>serialized = json.dumps(vehicle_entity)

>>>restored = json.loads(serialized)
```

We can customize serialized objects by providing serializing options as follows:

- encoding – character encoding
- indent – pretty-printing with indentation
- sort_keys – output sorted by key
- separator – tuple (item separator, key separator)

eg.

```
>>> json.dumps([1, 2, 3, {'4': 5, '6': 7}], separators=(',', ':'))

'[1,2,3,{"4":5,"6":7}]

>>> print(json.dumps({'4': 5, '6': 7}, sort_keys=True, indent=4))

{

    "4": 5,

    "6": 7

}
```

Similarities between XML and JSON

- Frequently used for data exchange
- Human readable
- Supported by most programming languages

Differences between XML and JSON

- JSON is lightweight than XML
- JSON is less verbose and simpler than XML
- XML provides more functionality

However, neither of them is superior and we need to select XML or JSON depending on the requirement.

GET IN TOUCH

University of Moratuwa
Centre for Open & Distance Learning
CODL
011 308 2787/8
011 265 0301 ext. 3850,3851
open@uom.lk

University Website
CODL Website