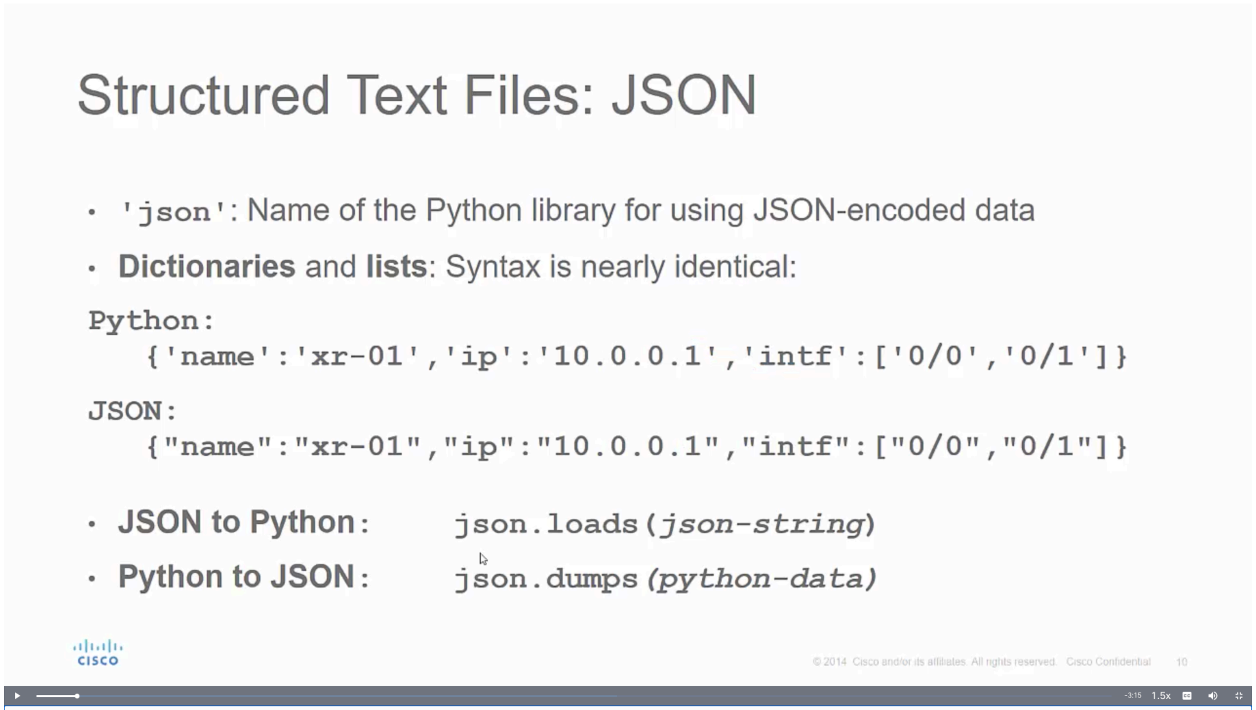
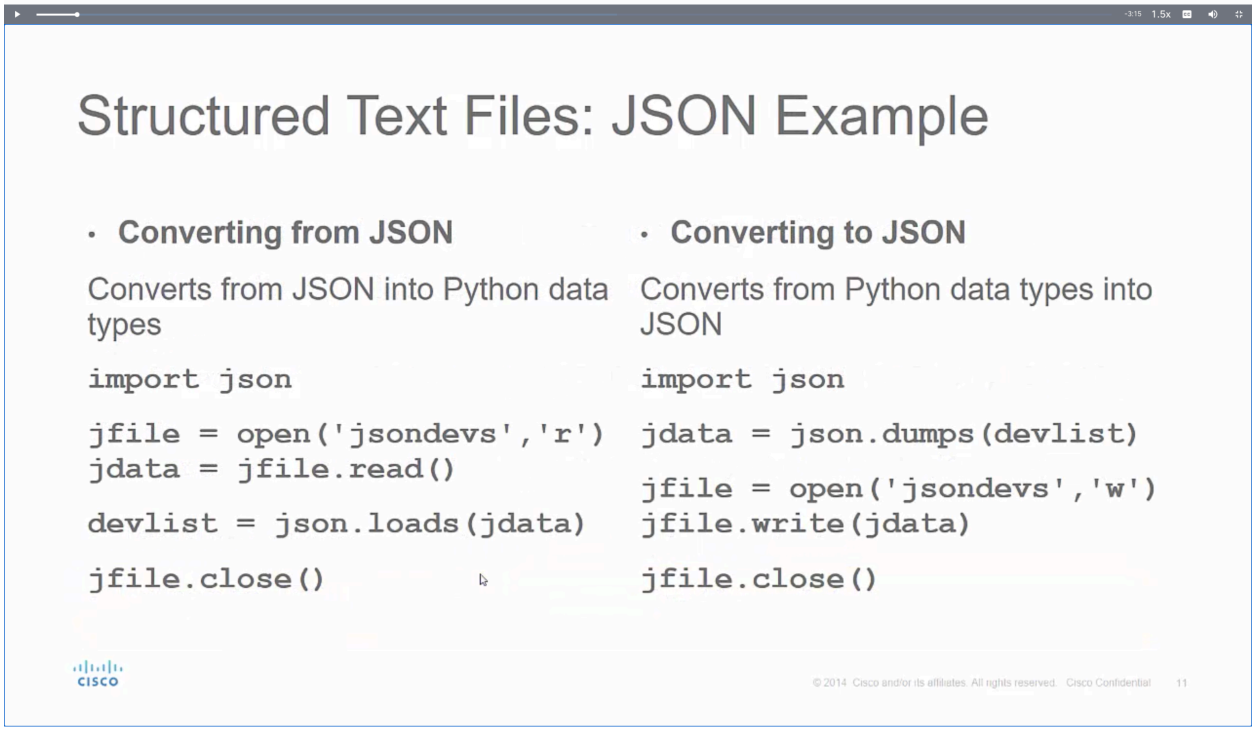


Structured Text Files: JSON

- `'json'`: Name of the Python library for using JSON-encoded data
- **Dictionaries** and **lists**: Syntax is nearly identical:

```
Python:
    {'name':'xr-01','ip':'10.0.0.1','intf':['0/0','0/1']}
JSON:
    {"name":"xr-01","ip":"10.0.0.1","intf":["0/0","0/1"]}
```

- **JSON to Python**: `json.loads(`*`json-string`*`)`
- **Python to JSON**: `json.dumps(`*`python-data`*`)`



Structured Text Files: JSON Example

- **Converting from JSON**

Converts from JSON into Python data types

```
import json

jfile = open('jsondevs','r')
jdata = jfile.read()

devlist = json.loads(jdata)

jfile.close()
```

- **Converting to JSON**

Converts from Python data types into JSON

```
import json

jdata = json.dumps(devlist)

jfile = open('jsondevs','w')
jfile.write(jdata)

jfile.close()
```

The main JSON data structures, Object and Array, and very similar to the Python data structures of Dictionaries and Lists:

JSON Object and Python Dictionary:

- In JSON, an Object is a collection of name-value pairs, which are enclosed by curly brackets and separated by a colon: `{"name":"xr-01", "ip":"10.0.0.1"}`
- In Python, a Dictionary is a collection of name-value pairs, which are enclosed by curly brackets and separated by a colon: `{'name':'xr-01', 'ip':'10.30.30.1'}`

JSON Array and Python List:

- In JSON, an Array is a sequence of items of any type, which is enclosed by square brackets.
- In Python, a List is a sequence of items of any type, which is enclosed by square brackets.

You will convert from a JSON string into the appropriate Python structures using the `json.loads()` function. You will convert from a Python structure to a JSON string using the `json.dumps()` function.

Conversion is actually to and from a string. You will use a normal file read operation to read the JSON data from a file into a string, which you will then convert to Python data. And you will use a normal file write operation to write the JSON string to a text file.

The read operation works as follows:

```
json_file = open('json_devices', 'r') # just a normal file open
json_device_data = json_file.read()  # just a normal file read operation
devices_list = json.loads(json_device_data)  # convert to Python data structure
```

At this point, devices_list is in the familiar structure which will follow what was defined in the JSON file itself. If each piece of information in the file was specified as a JSON object, as follows:

```
    [
      {"name":"ios-01","os":"ios","ip":"10.30.30.1","user":"cisco","password":"c
      {"name":"ios-02","os":"ios","ip":"10.30.30.2","user":"cisco","password":"c
      {"name":"ios-03","os":"ios","ip":"10.30.30.3","user":"cisco","password":"c
    ]
```

The result will be a Python list (the list of devices) of dictionaries (the specific device information for each device).

If instead the data in the file for the device info was defined as an array, as follows:

```
    [
       ["ios-01","ios","10.30.30.1","cisco","cisco"],
       ["ios-02","ios","10.30.30.2","cisco","cisco"],
       ["ios-03","ios","10.30.30.3","cisco","cisco"]
    ]
```

The result would be a Python list (the list of devices) of lists (the lists of device information for each device).

Writing JSON data is similar but in reverse:

```
json_device_data = json.dumps(devices_out_list)
with open(json_devices_file, 'w') as json_file:
    json_file.write(json_device_data)
```

Reading and writing is very simple with Python, as the translations from one to the other are direct and straightforward. JSON has many advantages and is quite popular, and therefore being able to read and write JSON data is an important skill in developing Python applications.