# Laboratory Tutorial 10: JSON

In this laboratory tutorial you will work with JSON and Python.

## Preamble

In the previous lab tutorial, you gained experience in creating XML elements, using external DTD to define the structure of an XML document, and validating both XML and DTD docs. Additionally, you have used python to write and read xml files.

In this lab tutorial you will:

- Represent data in JSON format
- Validate a JSON doc
- Convert a JSON string to Python object
- Convert a Python object to JSON string
- Read and write JSON files

You will need Jupyter notebook for Python 3.

## Exercise 10.1: JSON and Validation

### 1.

Refer to the following scenario:

Within a JSON object we have:

- An array variable – **student** – which contains 3 objects. For the first object we have 3 keys and 3 values respectively: id, name, age, 1234, JS, and 23. In terms of object 2 we have 3 keys and 3 values respectively: id, name, age, 2233, PK, and 20. For the last object, we have the following keys and values respectively: id, name, age, 2468, MN, and 19.
- An array variable – **lecturer** – which contains 3 objects. For the first object we have 3 keys and 3 values respectively: lno, name, room, 101, SG, and 106. For object 2 we have 3 keys and values respectively: lno, name, room, 103, KP, and 218. Finally, for the last object, we have the following keys and values respectively: lno, name, and room: 106, KS and 156.
- A key variable – **dept** – with an object which includes multiple keys and values respectively: id, name, location, CS101, Computer Science, and east campus.

- A key variable – **module** – with an object which includes multiple keys and values respectively: module1, module2, module3, INT1, DATA2, and HCI2.
- A single key – **rank** – with a value **20**.
- A single key – **Russell group** – with a Boolean value **true**.

*Task: Using a single text editor represent the above data in JSON format.*

Save the file as **ex1** with **.json** extension.

## 2.

Next, using the following validator check if your JSON file (ex1) is well formed. You are free to use your own validator.

https://jsonlint.com/

# Exercise 10.2: Reading JSON files in Python

**We are going to read the JSON file (that is ex1) which was created in Exercise 10.1.**

(a) Open Jupyter

(b) Next, we need to import JSON. Type the following:

```
import json
```

(c) The JSON file **(ex1)** needs to be opened and loaded. To do this type the following:

```
with open('ex1.json') as jsonfile:
    ex1=json.load(jsonfile)
```

(d) Now, type **ex1**. Click Run. The following should appear.

```
{'student': [{'id': 1234, 'name': 'JS', 'age': 23},
  {'id': 2233, 'name': 'PK', 'age': 20},
  {'id': 2468, 'name': 'MN', 'age': 19}],
 'lecturer': [{'lno': 101, 'name': 'SG', 'room': 106},
  {'lno': 103, 'name': 'KP', 'room': 218},
  {'lno': 106, 'name': 'KS', 'room': 156}],
 'dept': {'id': 'CS101',
  'name': 'Computer Science',
  'location': 'east campus'},
 'module': {'module1': 'INT1', 'module2': 'DATA2', 'module3': 'HCI2'},
 'rank': 20,
 'russell group': True}
```

Because we are working with Python, you can see that our JSON document is converted into Python dictionary (we will talk about dictionary in Exercise 10.5). For example, in JSON, the letter **t** in **true** is of lower case whereas in Python it is of uppercase.

(e) Now, to view the information for the key 'lecturer' type the following and then click **Run**.

```
ex1['lecturer']
```

(f) If you want to return only the first object of key 'lecturer' then you perform the following:

ex1['lecturer'][0]

(g) *Show the code to return only the room number of the key 'lecturer' within the second object.*

(h) *Show the code to search and return the name for each student.*

## Exercise 10.3: Manipulating JSON files in Python

**We will continue to use the file (ex1). We are going to make two changes.**

a. Let us update the value for key 'rank' to 19. To do that we need the following code.

ex1["rank"]=19

b. We will check whether the change made in a) has occurred or not. To do this simply type **ex1** and then click **Run**. Key 'rank' should now have a value of 19.

c. *Next, we need to delete the age for each student. Show the code to make this change.*

d. *Show the code to check whether the change made in (c) has occurred or not.*

## Exercise 10.4: Writing JSON files in Python

Having made the changes in Exercise 10.3, we will now write this file. The code is given below.

```
with open('ex1new.json','w') as newfile:
    json.dump(ex1,newfile)
```

Click **Run**. A file (ex1new) is created in the same location where your python file is resided. Please check.

# Exercise 10.5: CONVERTING JSON TO PYTHON

If we have a JSON string, it can be loaded into Python object using json.loads() method. In turn, you will get a Python dictionary. Consider the following JSON string:

```
a = “””
{
"store": "Tesco Express",
"id": "T12588",
"location": "Clifton Moore",
"payment methods": {
            "method1": "cash",
            "method2": "debit card",
            "method3": "credit card",
            "method4": "cheque",
            "method5": "tesco vouchers"
            },
"cash machine": null,
"24 hours": false
}“””
```

1. Open Jupyter
2. Import the JSON module:

   import json

3. Copy and paste the above JSON string
4. Next, we will parse the string into the load function:

   b=json.loads(a)

5. Type **b** and click **Run. We have now got a python dictionary.** How do we know that it is a python dictionary? - The letter **f** in word **False** is upper case, and **null** is replaced to **None**.

```
{'store': 'Tesco Express',
 'id': 'T12588',
 'location': 'Clifton Moore',
 'payment methods': {'method1': 'cash',
  'method2': 'debit card',
  'method3': 'credit card',
  'method4': 'cheques',
  'method5': 'tesco vouchers'},
 'cash machine': None,
 '24 hours': False}
```

6. Now, let us perform the following query. The result should be Tesco Express.

   b['store']

## Exercise 10.6: CONVERTING PYTHON TO JSON

In Exercise 10.5 we converted a JSON string into a Python object. Now, we are going to convert the **same** Python object back to JSON string. To do this we need the json.dumps() method.

(a) Type the following:

c=json.dumps(b)

(b) Next, type **c** and click **Run**. You should see your JSON string is back:

```
'{"store": "Tesco Express", "id": "T12588", "location": "Clifton Moore", "payment methods": {"method1": "cash", "meth
od2": "debit card", "method3": "credit card", "method4": "cheques", "method5": "tesco vouchers"}, "cash machine": nul
l, "24 hours": false}'
```

(c) Let us save this JSON value into a json file called **jsontopython**. *Show the code to do this.*

(d) From (c) the new file will be created in the same location where your Python file resides. Go and check.

## Summary and future work

In this tutorial you have learned how to represent data in JSON format and validate a JSON doc. Additionally, you have learned how to read, write, and manipulate files in Python. Finally, you have gained knowledge of converting JSON string to Python dictionary and vice versa.

## Further Reading

https://courses.cs.washington.edu/courses/cse154/14au/lectures/11-21/slides.pdf

https://www.academia.edu/19435559/JSON_Tutorial