**JSON:**

JSON stands for **J**ava**S**cript **O**bject **N**otation.

* JSON is a lightweight data-interchange format
* JSON is plain text written in JavaScript object notation
* JSON is **Language independent.**
* JSON is simpler to read and write when compared to XML.

JSON is a **text format** for storing and transporting data. It transports data from client to server and vice versa.

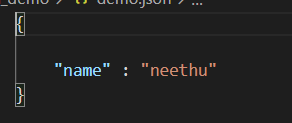
**Storing Data:**

JSON makes it possible to store **JavaScript objects** as text.

**JSON Syntax Rules:**

JSON syntax is derived from JavaScript object notation syntax:

* Data is in name/value pairs
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays
* The JSON format is almost identical to JavaScript objects.
* In JSON, *keys* must be strings, written with double quotes.



**JSON Values:**

In **JSON**, *values* must be one of the following data types:

* a string
* a number
* an object
* an array
* a boolean
* null
* Dates and datetimes are best stored as strings in a format that is widely used.

**JSON VS XML:**

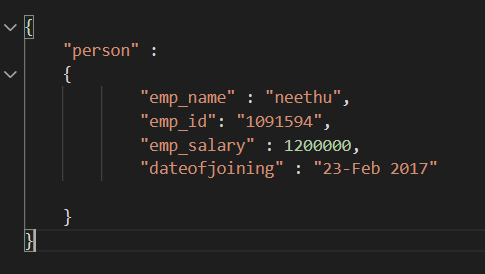
Both JSON and XML can be used to receive data from a web server.

**Why json is better than xml?**

XML is much more difficult to parse than JSON.  
JSON is parsed into a ready-to-use JavaScript object.

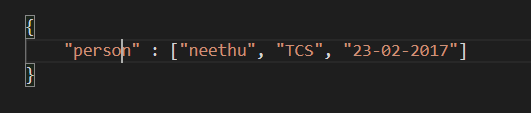
**JSON Objects:**

* Curly braces hold objects.



**JSON Arrays:**

* Square brackets hold arrays. (It is an ordered collection of values)



**Parsing functions:**

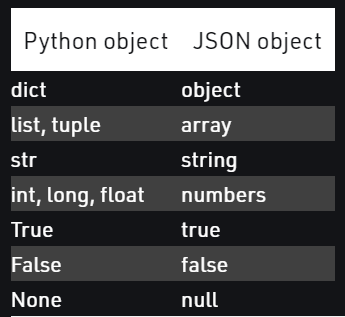
* Functions are not allowed in JSON.
* If you need to include a function, write it as a string. You can convert it back into a function later.

**Parsing dates:**

* Date objects are not allowed in JSON.
* If you need to include a date, write it as a string. You can convert it back into a date object later.

**Python JSON:**

* Python supports JSON objects by inbuilt function called json.
* This package provides all the necessary tools for working with JSON Objects including parsing, serializing, deserializing, and many more.
* The text in JSON is done through quoted-string which contains the value in key-value mapping within {}.
* It is like the dictionary in Python.
* JSON shows an API like users of Standard Library marshal and pickle modules and Python natively supports JSON features.
* JSON supports primitive types, like strings, numbers, nested lists, tuples and objects.



**Serializing JSON:**

* The process of encoding JSON is usually called **serialization**. This term refers to the transformation of data into a series of bytes (hence serial) to be stored or transmitted across a network.
* To handle the data flow in a file, the JSON library in Python uses **dump()** function to convert the Python objects into their respective JSON object, so it makes easy to write data to files.

**Deserializing JSON:**

* The Deserialization is the opposite of Serialization, i.e. conversion of JSON object into their respective Python objects.
* The load() method is used for it. If you have used Json data from another program or obtained as a string format of Json, then it can easily be deserialized with load(), which is usually used to load from string, otherwise, the root object is in list or dict.

**Encoding and Decoding:**

* Encoding is defined as converting the text or values into an encrypted form that can only be used by the desired user through decoding it. Here encoding and decoding is done for JSON (object)format.
* Encoding is also known as Serialization and Decoding is known as Deserialization. Python has a popular package for this operation and known as **Demjson**.
* The encode() function is used to convert the python object into a JSON string
* The decode() function is used to convert the JSON object into python-format type.

**CRUD operations and http method:**

C – Create - POST

R – Read - GET

U – Update - PUT

D – Delete - DELETE

**Resources:**

<https://www.w3schools.com/python/python_json.asp>

<https://www.geeksforgeeks.org/python-json/>