JSON

Aarju Dixit
Research Scholar
Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur, Rajasthan, India 342030

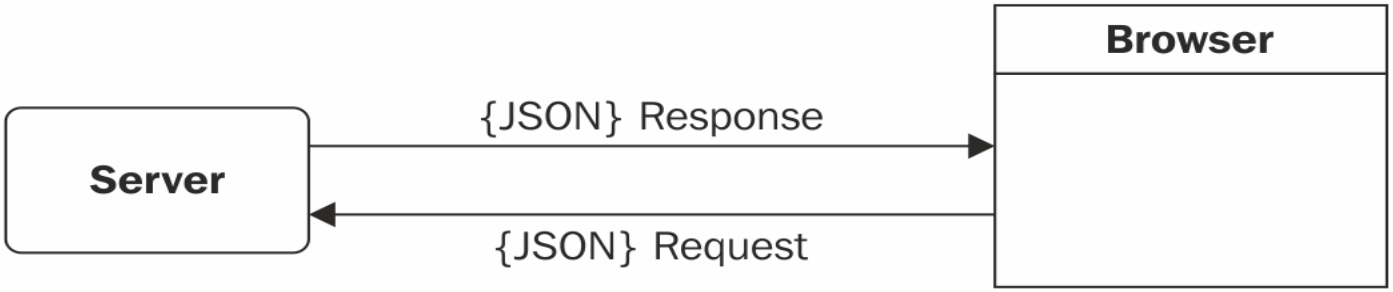
CSL4030 Data Engineering Lab 7
October 4th 2023

Document-based databases

- **MongoDB**: A top NoSQL database engine that uses something similar to JSON. MongoDB has its own query language.
- **Cosmos DB**: Saves documents in JSON.
- **CouchDB:** An open source database management system that uses JSON natively.
- **Oracle:** Provides SQL querying and reporting over JSON documents. Oracle Autonomous JSON Database is MongoDB compatible.
- **Elasticsearch:** A search engine based on the document-store data model.

Data Interchange Formats in JSON [4]

- JSON can be used in web applications for data transfer. Consider the following block diagram of the simple client-server architecture. Assume that the client is a browser that sends an HTTP request to the server, and the server serves the request and responses as expected.
- In the preceding two-way communication, the data format used is a serialized string, with the combination of key-value pairs enveloped in parentheses; that is JSON!
- Prior to JSON, XML was considered to be the chosen data interchange format.
- XML parsing required an XML DOM implementation on the client side that would ingest the XML response, and then XPath was used to query the response in order to access and retrieve the data. This made life tedious, as querying for data had to be performed at two levels: first on the server side where the data was being queried from a database, and a second time on the client side using XPath.
- JSON does not need any specific implementations; the JavaScript engine in the browser handles JSON parsing.



Introducing JSON [1]

- JSON (JavaScript Object Notation) is a **lightweight** data-interchange format.
- It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999.
- JSON is a **text format** that is completely language **independent** but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.
- These properties make JSON an ideal data-interchange language.

- JSON is built on two structures:
  - A **collection of name/value pairs**. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
  - An **ordered list** of values. In most languages, this is realized as an array, vector, list, or sequence.

Example of XML and JSON [4]

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This is an example of student feed in XML -->
<students>
  <student>
    <studentid>101</studentid>
    <firstname>John</firstname>
    <lastname>Doe</lastname>
    <classes>
      <class>Business Research</class>
      <class>Economics</class>
      <class>Finance</class>
    </classes>
  </student>
  <student>
    <studentid>102</studentid>
    <firstname>Jane</firstname>
    <lastname>Dane</lastname>
    <classes>
      <class>Marketing</class>
      <class>Economics</class>
      <class>Finance</class>
    </classes>
  </student>
</students>
```

```json
{
 "students" :{
  "0" :{
    "studentid" : 101,
    "firstname" : "John",
    "lastname" : "Doe",
    "classes" : [
      "Business Research",
      "Economics",
      "Finance"
    ]
  },
  "1" :{
    "studentid" : 102,
    "firstname" : "Jane",
    "lastname" : "Dane",
    "classes" : [
      "Marketing",
      "Economics",
      "Finance"
    ]
  }
 }
}
```

JSON Data Types

- In JSON, values must be one of the following data types: [2]

  - a string
  - a number
  - an object (JSON object)
  - an array
  - a Boolean
  - Null

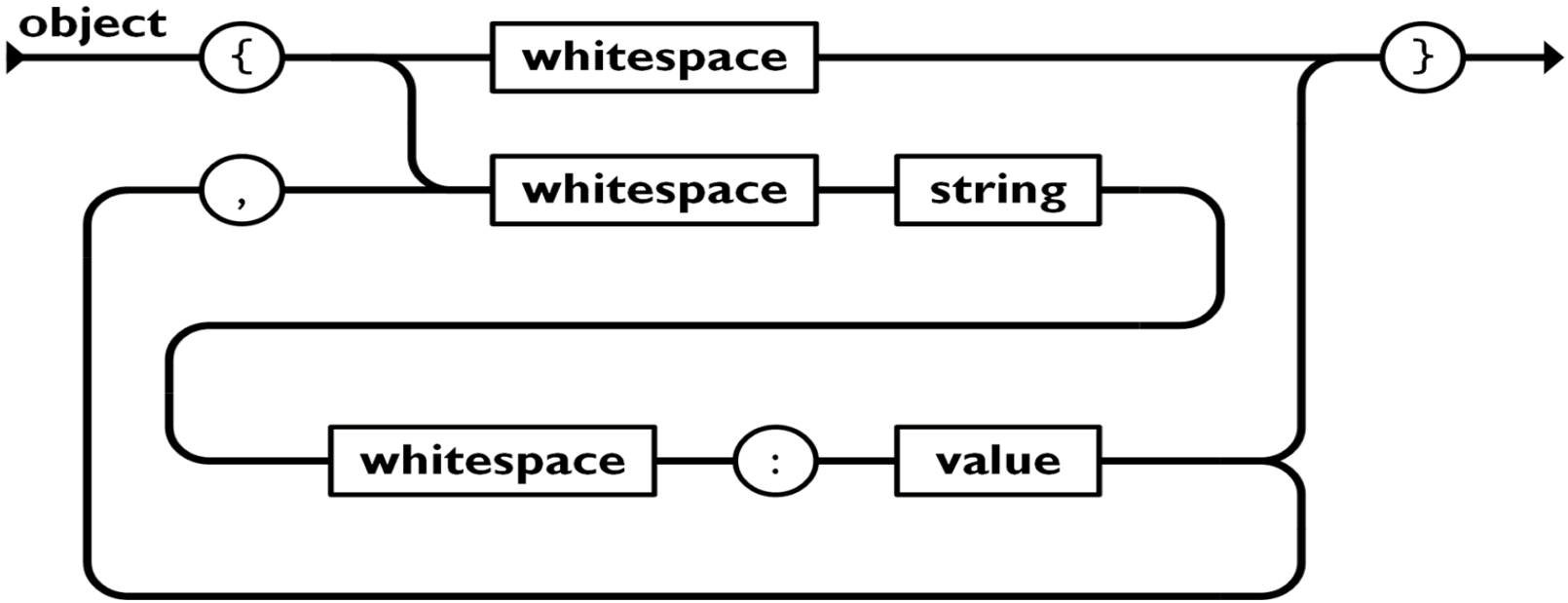
- JSON values cannot be one of the following data types: [2]

  - a function
  - a date
  - undefined

Object in JSON

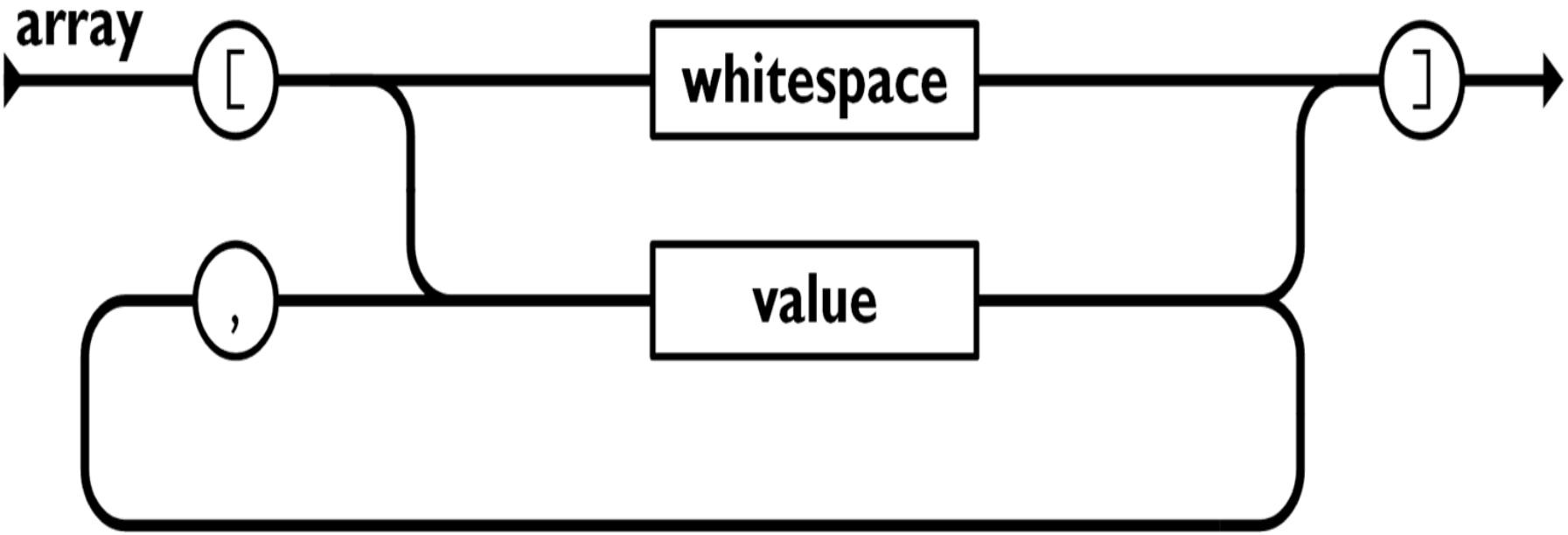
An **object** is an unordered set of name/value pairs. An object begins with {left brace and ends with }right brace. Each name is followed by :colon and the name/value pairs are separated by ,comma.

- Ex: {
  "employee":{"name":"John", "age":30, "city":"New York"}
  } [2]



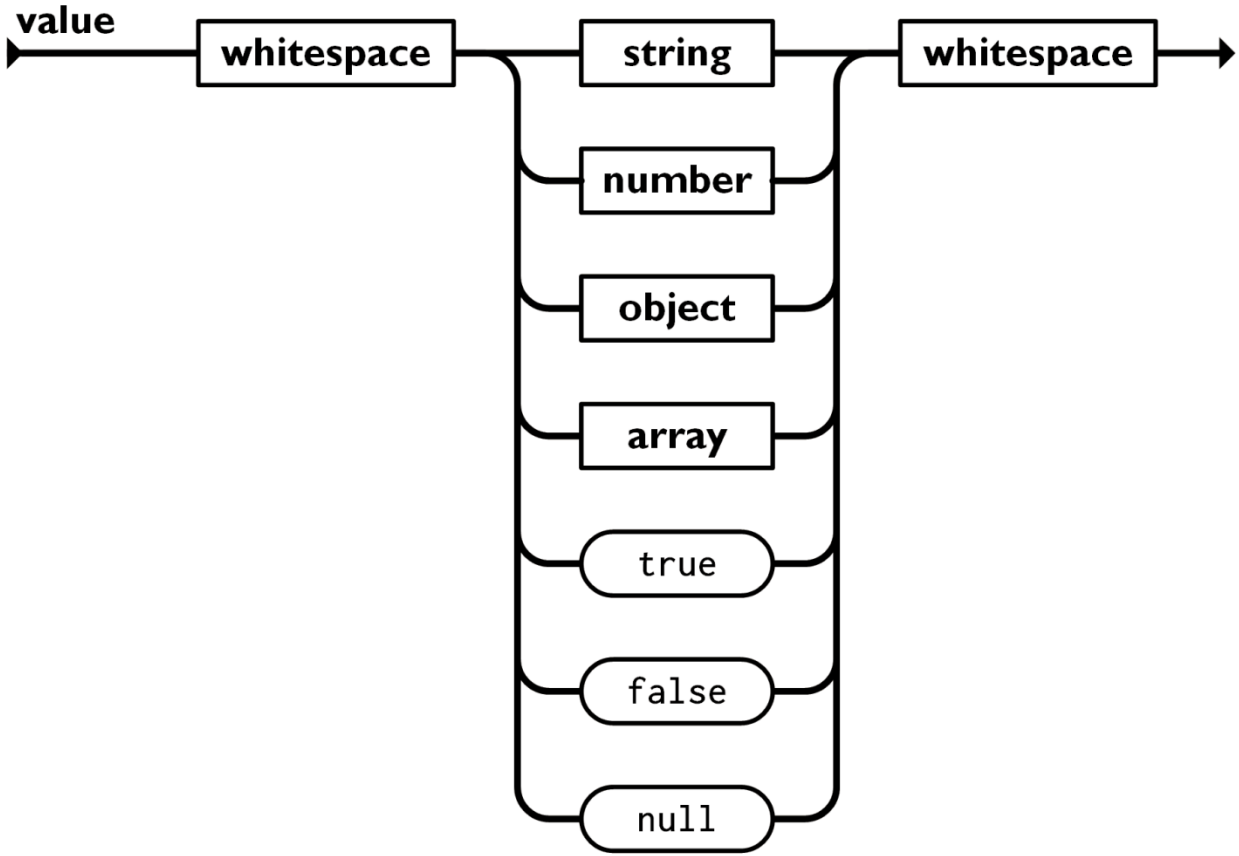
Array in JSON

- An **array** is an ordered collection of values. An array begins with [left bracket and ends with]right bracket. Values are separated by ,comma.
  - Ex: {
    "employees":["John", "Anna", "Peter"]
    } [2]



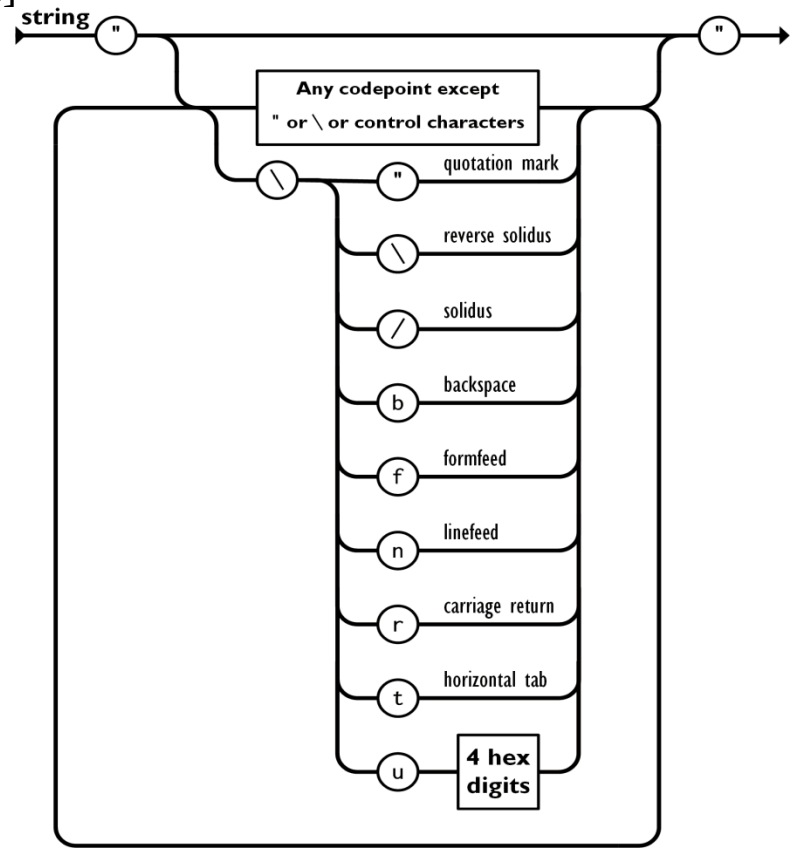
Value in JSON

- A **value** can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.



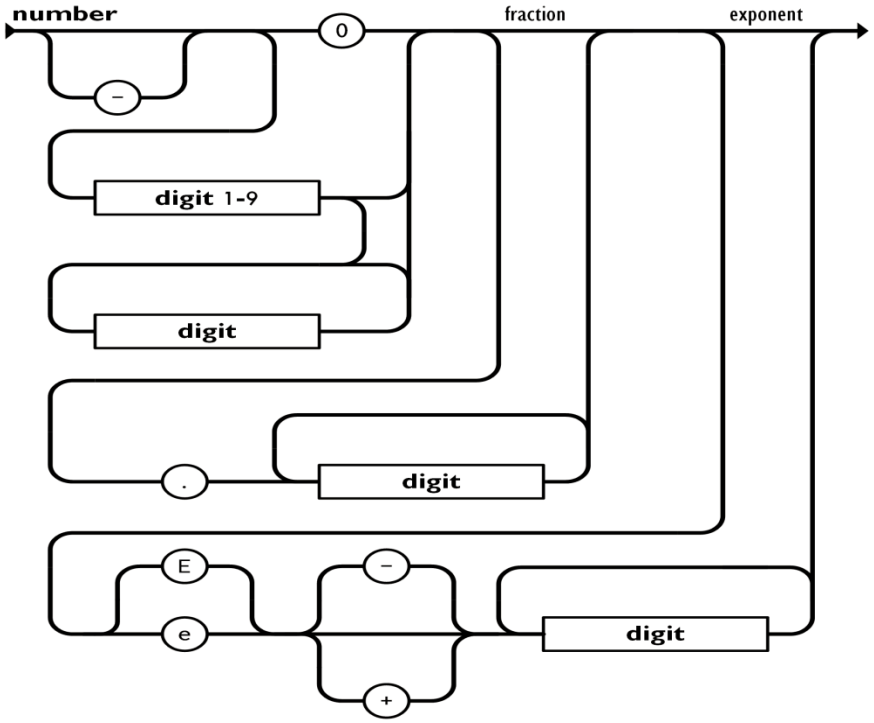
String in JSON

- A **string** is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.
  - Ex: {"name":"John"} [2]



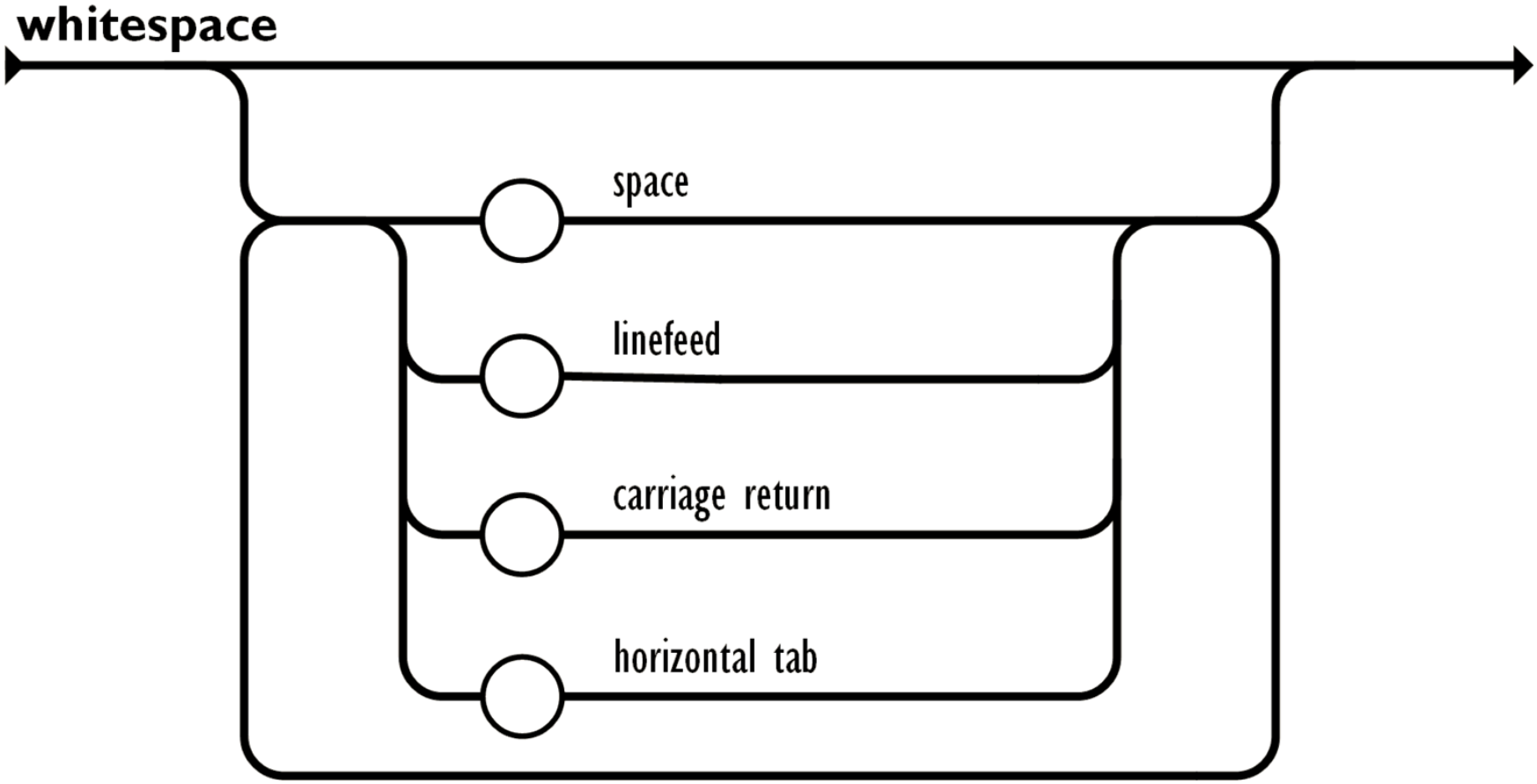
Number in JSON

- A **number** is very much like a C or Java number, except that the octal and hexadecimal formats are not used.
  - Ex: {"age":30} [2]



Whitespace in JSON

- Whitespace can be inserted between any pair of tokens. Excepting a few encoding details, that completely describes the language.



Simple Encoder in JSON

- class json.JSONEncoder(*, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None) [3]
- Extensible JSON encoder for Python data structures.
- Supports the following objects and types by default:

Python	JSON
dict	object
list, tuple	array
str	string
int, float, int- & float-derived Enums	number
True	true
False	false
None	null

Encoding in JSON

- JSON exposes an API familiar to users of the standard library **marshal** and **pickle** modules.

- **Encoding basic Python object hierarchies:** [3]

```
import json json.dumps(['foo', {'bar': ('baz', None, 1.0, 2)}])
print(json.dumps("\"foo\bar"))
print(json.dumps('\u1234'))
print(json.dumps('\\'))
print(json.dumps({"c": 0, "b": 0, "a": 0}, sort_keys=True))
from io import StringIO io = StringIO() json.dump(['streaming API'], io) io.getvalue()
```

- **Compact encoding:** [3]

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

- **Pretty printing:** [3]

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

Simple JSON decoder

- class json.JSONDecoder(*, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, strict=True, object_pairs_hook=None) [3]
- Translations in decoding by default:

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

Decoding in JSON

- **Decoding JSON:** [3]

```
import json
json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')
json.loads('"\\"foo\\bar"')
from io import
StringIO io = StringIO('["streaming API"]')
json.load(io)
```

Specializing JSON object decoding: [3]

```
import json
def as_complex(dct):
    if '__complex__' in dct:
        return complex(dct['real'], dct['imag'])
    return dct

json.loads('{"__complex__": true, "real": 1, "imag": 2}',
    object_hook=as_complex)

import decimal
json.loads('1.1', parse_float=decimal.Decimal)
```

Python - convert image to JSON [5]

As the base64.encodebytes() has been deprecated in base64 and base64.b64decode(data['img']) to convert back

```
import json
import base64

data = {}
with open('some.gif', mode='rb') as file:
        img = file.read()
data['img'] =base64.encodebytes(img).decode('utf-8')

print(json.dumps(data))
```

Working With JSON Data in Python

- JSON is supposed to be readable by anyone who's used a C-style language, and Python is a C-style language.
- JSON supports primitive types, like strings and numbers, as well as nested lists and objects.

```
{
   "firstName": "Jane",
   "lastName": "Doe",
   "hobbies": ["running", "sky diving", "singing"],
   "age": 35,
   "children": [
      {
         "firstName": "Alice",
         "age": 6
      },
      {
         "firstName": "Bob",
         "age": 8
      }
   ]
}
```

Python Supports JSON Natively!

- Python comes with a built-in package called json for encoding and decoding JSON data.
  Ex: import json

References

[1]. https://www.json.org/json-en.html

[2]. https://www.w3schools.com/js/js_json_datatypes.asp

[3]. https://docs.python.org/3/library/json.html

[4].https://subscription.packtpub.com/book/webdevelopment/9781788624701/1/ch01lvl1sec10/json-a-dataexchangeformat#:~:text=To%20define%20JSON%2C%20we%20can,is%20not%20dependent%20on%20JavaScript

[5] https://stackoverflow.com/questions/54660800/python-convert-image-to-json/54661844#54661844