A (Very) Brief History of JSON

Not so surprisingly, **J**ava**S**cript **O**bject **N**otation was inspired by a subset of the JavaScript programming language dealing with object literal syntax. They've got a nifty website that explains the whole thing. Don't worry though: JSON has long since become language agnostic and exists as its own standard, so we can thankfully avoid JavaScript for the sake of this discussion.

Ultimately, the community at large adopted JSON because it's easy for both humans and machines to create and understand.

Look, it's JSON!

Get ready. I'm about to show you some real life JSON—just like you'd see out there in the wild. It's okay: JSON is supposed to be readable by anyone who's used a C-style language, and Python is a C-style language…so that's you!

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

As you can see, JSON supports primitive types, like strings and numbers, as well as nested lists and objects.

Python Supports JSON Natively!

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

Just throw this little guy up at the top of your file:

```
import 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. You may also hear the term **marshaling**, but that’s a whole other discussion. Naturally, **deserialization** is the reciprocal process of decoding data that has been stored or delivered in the JSON standard.

Serializing JSON

What happens after a computer processes lots of information? It needs to take a data dump. Accordingly, the `json` library exposes the `dump()` method for writing data to files. There is also a `dumps()` method (pronounced as “dump-s”) for writing to a Python string.

Simple Python objects are translated to JSON according to a fairly intuitive conversion.

Python	**JSON**
`dict`	`object`
`list`, `tuple`	`array`
`str`	`string`
`int`, `long`, `float`	`number`
`True`	`true`
`False`	`false`
`None`	`null`