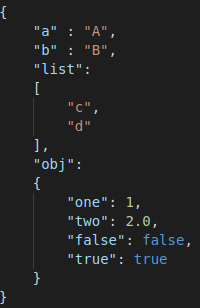
# **JSON Parser**

# **What is JSON?**

JSON stands for JavaScript Object Notation and is a file format used to hold data from JavaScript Objects in text form. JavaScript Objects form a tree structure where the containers (trees) are the objects or arrays and the end nodes contain values like strings, numbers or boolean values. A JSON Object in text form looks like this:

 The objects are surrounded by curly brackets { and }, while arrays are surrounded by [ and ]. Each variable (“a” : “A”) inside an object contains a name (“a”) and a value (“A”) while in arrays it contains on a value. Like inside list there are only values “c” and “d”. So the values inside an object are referenced by name which is a string while for arrays they are referenced by index. Also as we can see objects and arrays can contain other object and arrays which forms a tree structure.

The string values are surrounded by double quotes just like names are. Numbers are just written as numbers and can be integers or floating point numbers. Boolean values are either true or false without double quotes like in the example. Variables are separated with a comma. The name and value in objects are separated by two points ( : ).

**The Problem**

The Problem is to turn a JSON text file to a JSON tree structure that we can use to get and modify the values. The tree structure for the example looks like this:

root

a b list obj

A B c d one two true false

1 2.0 true false

**The Solution**

First we need to find the type of the root object. We can find that by checking the first non space character:

1. { → object
2. [ → array
3. “ → string
4. digit → number
5. t or f → boolean

For each of those we then need to parse the value:

* For strings we just read all the characters up to the next double quote.
* For numbers we read all characters as long as they are digits. That works for integers however a number can also be floating point so we need to check whether the next character is a dot. If it is we skip it and and read all digits after it. Otherwise we just return the integer value we found earlier.
* For boolean read characters until we hit a non-alpha one. If the string we found is either ‘true’ or ‘false’ than we return the corresponding boolean value.
* For arrays we go through a loop which read all the comma separated until instead of a comma there is the closing character ‘]’.
* For objects the method is similar to arrays since they are both containers. The difference is that objects have a name and the character ‘:’ which needs to be read before the actual value followed by a comma.

This is also applied for all the other values further down the tree.

Once we have read the first value if it is not a container then we are done. Otherwise we follow the rules laid out before to read either an object or an array.

**Runtime**

The runtime of this algorithm depends on the size of the file or more simply the number of characters. Since we go through each character once the time complexity is O(n) where n is the number of characters. There are some more things at play like the types of variables. For example parsing a float will take longer than copying a string or an object/array will require memory allocation. But in essence these are O(1) complexity since they are the same. So they cancel out and leave O(n).

To verify that we can create JSON objects with different sizes and calculate how much time it takes to parse. Here are the results:

|  |  |  |
| --- | --- | --- |
| Nr of characters | Nr of variables | Time (ns) |
| 4 | 0 | 4.77 |
| 223 | 20 | 139.47 |
| 463 | 40 | 236.27 |
| 703 | 60 | 346.18 |
| 943 | 80 | 459.19 |
| 1183 | 100 | 576.50 |
| 1463 | 120 | 704.77 |
| 1743 | 140 | 880.24 |
| 2023 | 160 | 951.05 |
| 2303 | 180 | 1112.22 |

As we can see the results are consistent with what we expected.

**Notes**

* If you find this project on github it’s because I put it there and not because I plagiarized :) . My username is peTheProgrammer. Repository: https://github.com/peTheProgrammer/python\_json
* The data.json file was generated using this website: https://www.json-generator.com/
* I only added comments where I thought necessary. Most of the code, I found, just speaks for itself and it is a waste of time to comment code like this example from line 82:

def nextChar(): function to go to next character

global index declare global variables that will be used

global string

global line

if string[index] == '\n': if character is new line

line += 1 increment line

index += 1 increment index to go to next character