sJSON

**Availability**

This repository has code to implement sJSON in Python, PHP, and Swift. You can use this code to run sJSON as a command-line converter or to create a module to integrate into your code base.

**Introduction**

Take this valid JSON string:

[{“books”:[“Cat in the Hat”,”Are You My Mother?”,”Fox in Socks”]},{“colors”:[“red”,”yellow”,  
”blue”]},{“flavors”:[“chocolate”,”vanilla”,”strawberry”]}]

A computer has no problem reading that, but a human might struggle. When humans are involved, we spread things out and line things up to make it easier to see what’s going on.

[

{

“books”: [

“Cat in the Hat”,

“Are You My Mother?”,

“Fox in Socks”

]

},

{

“colors”: [

“red”,

”yellow”,

”blue”

]

},

{

“flavors”: [

“chocolate”,

”vanilla”,

”strawberry”

]

}

]

This extra spacing is helpful for the humans, but the computer ignores it. It still needs braces and brackets, commas and colons, and quotation marks to interpret it all correctly.

Let’s make it easier on us humans. Since we naturally rely on spacing, let’s make that meaningful, and the computer can figure out the special punctuation. That’s exactly what sJSON let’s you do. Here is our data written out using sJSON.

books:

Cat in the Hat

Are You My Mother?

Fox in Socks

colors:

red

yellow

blue

flavors:

chocolate

vanilla

strawberry

sJSON is easier for humans to both read and write, and we’ll let the computer do the work of figuring out where punctuation needs to go to render this as valid JSON. There are only a few rules to follow.

**Basic values**

To create a JSON array, write each element on its own line using the same indentation.

Tom

Dick

Harry[

“Tom”,

“Dick”,

“Harry”

]

To create a JSON object, separate the key and value with a colon. Write each key-value pair on its own line using the same indentation.

City: Tulsa

Population: 45 million

Location: 45’ W 37’N{

“City”: “Tulsa”,

“Population” : “45 million”,

“Location” : “45’ W 37’N”

}

String values, as you can see above, are written without quotations marks. Usually. More on that later.

**Nested values**

To nest arrays, write a hyphen all by itself as an array marker, then indent the next lines.

-

1

2

3

-

red

yellow

blue[

[

1,

2,

3

],

[

“red”,

“yellow”,

“blue”

]

]

To nest inside an object, write the key and colon on one line, then indent the next lines.

flavors:

chocolate

vanilla

strawberry{

“flavors”: [

“chocolate”,

“vanilla”,

“strawberry”

]

}

**Cheating with commas**

You can use commas to separate simple array elements on the same line.

numbers :

1,2,3

colors :

red, yellow, blue{

“numbers”: [

1,

2,

3

],

“colors”: [

“red”,

“yellow”,

“blue”

]

}

You can do the same comma trick with objects that contain simple values only.

Name: Tom, Age: 42, City: Miami

Name: Fred, Age: 37, City: Houston

Name: Lucy, Age: 43, City: Cincinnati

[

{

“Name”: “Tom”,

“Age”: 42,

“City”: “Miami”

},

{

“Name”: “Fred”,

“Age”: 37,

“City”: “Houston”

},

{

“Name”: “Lucy”,

“Age”: 43,

“City”: “Cincinnati”

}

]

**String values**

Notice we have not used quotes at all so far. sJSON is smart about putting quotes around your strings. Here are some things to keep in mind:

* Colons and commas are markers for arrays and objects. If you don’t want those interpreted as markers, they need to be inside a quoted string.
* A hyphen on a line by itself is an array marker. If you want it to be a string instead, then put quotes around it.
* sJSON will recognize and not put quotes around JSON numbers and JSON literals (true, false, null).
* Except for object keys. Only strings are allowed as object keys, so whatever you write for the key will always get quoted.
* We’ll add to this list after we’ve covered comments and raw JSON below.

[[NEED SOME EXAMPLES]]

**Empty values**

JSON allows empty arrays and empty objects. To write those in simpJSON, use a lone comma or a lone colon.

list of nobel prizes I’ve won so far:

,

{ “list of nobel prizes won I’ve won so far”: [] }

## a whole lot of nothing

:

:

:[

{},

{},

{},

]

**Comments**

As you might have noticed in the last example, two pound signs will mark the rest of the line as a comment.

members:

Larry

Dana

##Fred – he dropped out last month

Tom{

“members”: [

“Larry”,

“Dana”,

“Tom”

]

}

So in addition to commas and colons, if you need two pound signs to be output as part of a string, the string needs to be quoted.

**Records**

Look at this example again. These table-like structures, with the same keys repeated for each object, come up all the time.

Name: Tom, Age: 42, City: Miami

Name: Fred, Age: 37, City: Houston

Name: Lucy, Age: 43, City: Cincinnati

[

{

“Name”: “Tom”,

“Age”: 42,

“City”: “Miami”

},

{

“Name”: “Fred”,

“Age”: 37,

“City”: “Houston”

},

{

“Name”: “Lucy”,

“Age”: 43,

“City”: “Cincinnati”

}

]

sJSON has a shortcut for you, called a record, which you trigger with a double colon.

:: Name, Age, City

Tom, 42, Miami

Fred, 37, Houston

Lucy, 43, Cincinnati

[

{

“Name”: “Tom”,

“Age”: 42,

“City”: “Miami”

},

{

“Name”: “Fred”,

“Age”: 37,

“City”: “Houston”

},

{

“Name”: “Lucy”,

“Age”: 43,

“City”: “Cincinnati”

}

]

Isn’t that awesome! You specify the keys one time, after the double colon, and then write just the values on the lines that follow. Indented, of course.

If you have a record nested inside an object, you can write it like this.

## You can write

people:

:: Name, Age, City

Tom, 42, Miami

Fred, 37, Houston

Lucy, 43, Cincinnati

## Or you can collapse

people :: Name, Age, City

Tom, 42, Miami

Fred, 37, Houston

Lucy, 43, Cincinnati

{

“people”: [

{

“Name”: “Tom”,

“Age”: 42,

“City”: “Miami”

},

{

“Name”: “Fred”,

“Age”: 37,

“City”: “Houston”

},

{

“Name”: “Lucy”,

“Age”: 43,

“City”: “Cincinnati”

}

]

}

Keys and values should be in the same order. You can leave a value blank and it will be skipped in the output JSON.

[[put an example here of leaving a value blank]]

**Warnings and errors**

In sJSON, indentation is allowed (and expected) after an array marker, for nesting objects, and for record sets. If you write one of those but don’t indent the next lines, then instead of being interpreted as array or object markers, they will revert to strings and sJSON will emit a warning.

-

oops

not

nested[

“-“,

“oops”,

“not”,

“nested”

]

Warning: Expecting indentation after array marker “-“ on line 1.

If you indent in an unexpected place, sJSON will emit an error and stop.

Can, I, do, this:

the

answer

is

noWarning: Object is missing a value on line 1. Converted to string.

Can, I, do, this:

^^^^^

Error: Illegal indentation on line 2.

Can

I

do

this:

the

answer

is

yes[

“Can”,

“I”,

“do”,

{

“this”: [

“the”,

“answer”,

“is”,

“yes”

]

}

]