**JSON Databases**

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

In two words, it’s a text file that stores a python dictionary.

For example:

{

"firstName": "Jane",

"lastName": "Doe",

"hobbies": ["running", "sky diving", "singing"],

"age": 35,

"children": [

{

"firstName": "Alice",

"age": 6

},

{

"firstName": "Bob",

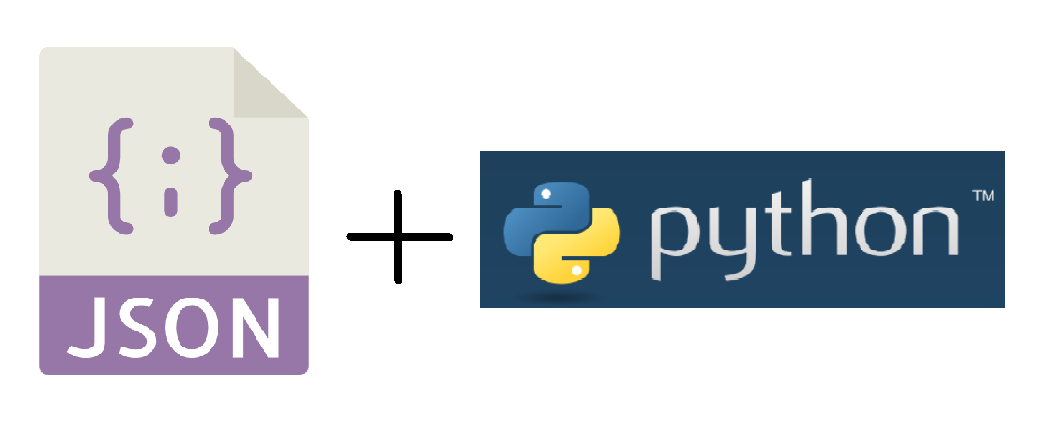
"age": 8

}

]

}

**Python & JSON**

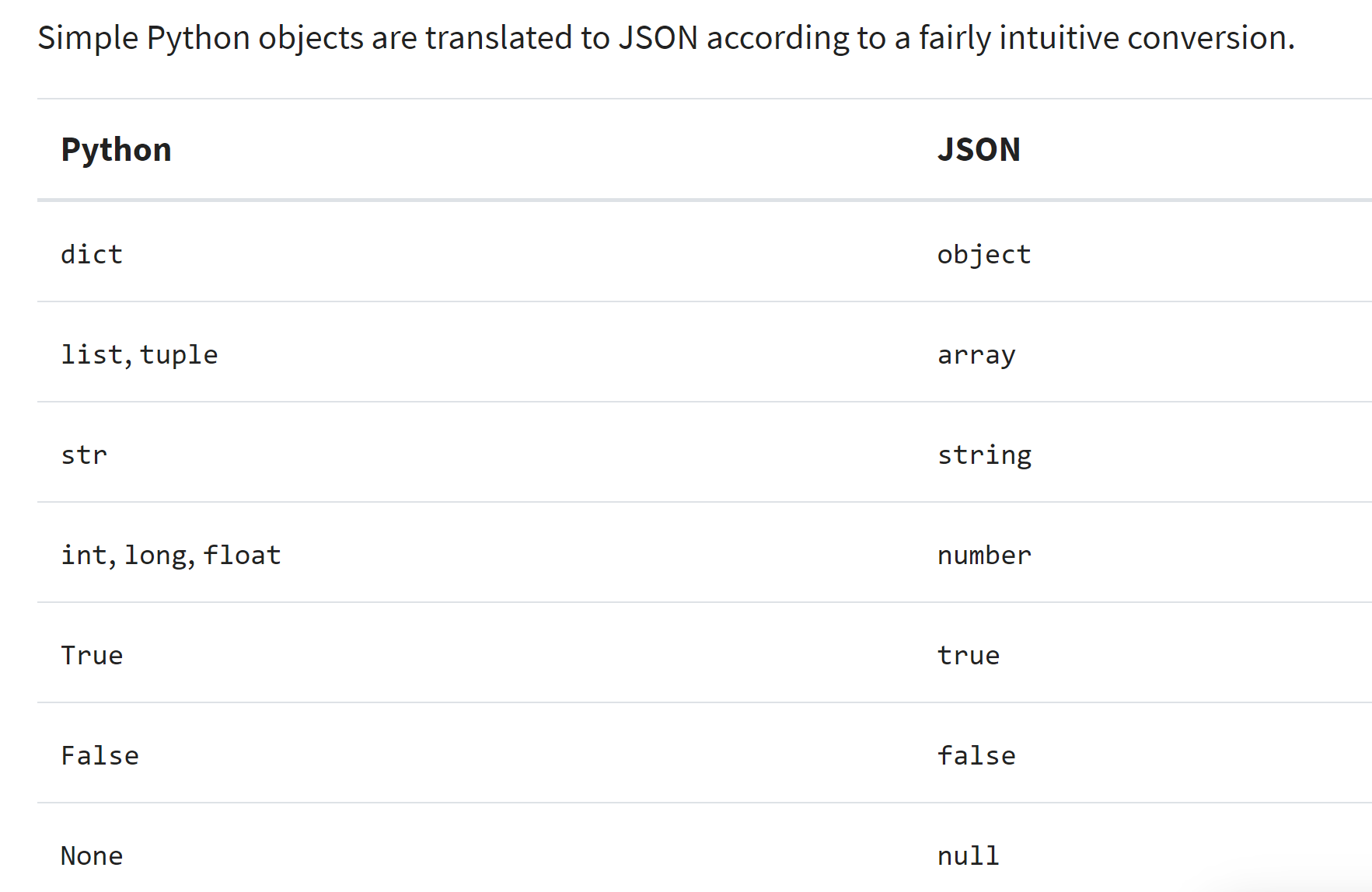


Python supports json natively, you don’t need to install anything to import the json module, just add this line in your code:

import json

Now you can easily save and read JSON data.

**Conversion Between Python To JSON**



**Write A Python Dictionary Into A JSON File**

1. First, let’s create a python dictionary :

my\_family = {

"parents":['Beth', 'Jerry'],

"children":['Summer', 'Morty']

}

Only Strings,Numbers, Booleans, Dictionaries, Lists, Tuples or None can be converted to JSON.

Here, my\_family is a dictionary with two entries, children and parents, which are both assigned to lists.

1. We want to save this python dictionary into a json file.

To save it, you first need a **destination file** (remember that JSON is stored in a text file, most of the time extended by .json).

Let’s take my\_file.json as example.

The python function **to save an object into a file** is :

* json.dump(object\_to\_save, destination\_file)

This function accept two arguments, the object to save and the destination file object.

**Careful!** The destination file should be a file object and **not a filename**.  
To get a file object, use the [open](https://www.programiz.com/python-programming/methods/built-in/open) function.

import json

my\_family = {

"parents":['Beth', 'Jerry'],

"children":['Summer', 'Morty']

}

json\_file = "my\_file.json"

with open(json\_file, 'w') as file\_obj:

json.dump(my\_family, file\_obj)

#json.dump(obj2save , destination\_file)

json.dump() transforms my\_family to a JSON string which will be saved in the my\_file.json file.

Now my\_file.json looks like:

{"parents": ["Beth", "Jerry"], "children": ["Summer", "Morty"]}

**Convert A Python Dictionary To JSON String**

You can also write the dictionary to a native Python str object, with the json.dumps function.

import json

my\_family = {

"parents" :['Beth', 'Jerry'],

"children" :['Summer', 'Morty']

}

json\_my\_family = json.dumps(my\_family)

print(json\_my\_family)

# >> {"parents": ["Beth", "Jerry"], "children": ["Summer", "Morty"]}

**Convert A Python Dictionary To JSON String : Pretty Print**

You saw in the two notions above (json.dump and json.dumps) that it convert a python dictionary into a JSON string. But the result is not really pretty and hard to understand because it writes the dictionary in one single line. What should we do to make it more readable ?

We can use the parameters indent and sort\_keys:

import json

my\_family = {

"parents" :['Beth', 'Jerry'],

"children" :['Summer', 'Morty']

}

json\_file = "index.json"

with open(json\_file, 'w') as file\_obj:

json.dump(my\_family, file\_obj, indent = 2, sort\_keys=True)

Now my\_file.json looks like this. Each key is indented by 2 and the keys are sorted alphabetically.

{

"children": [

"Summer",

"Morty"

],

"parents": [

"Beth",

"Jerry"

]

}

**Retrieve JSON Data**

Retrieving JSON data is done by the opposite function: json.load(source\_file). It only accepts a JSON file in argument:

import json

json\_file = 'my\_file.json'

with open(json\_file, 'r') as file\_obj:

my\_family = json.load(file\_obj)

print(my\_family)

#>> {'children': ['Summer', 'Morty'], 'parents': ['Beth', 'Jerry']}

**Exercise**

* Create a folder with two files : index.py and file.json. Save this code into the json file

{

"firstName": "Jane",

"lastName": "Doe",

"hobbies": ["running", "sky diving", "singing"],

"age": 35,

"children": [

{

"firstName": "Alice",

"age": 6

},

{

"firstName": "Bob",

"age": 8

}

]

}

* Retrieve the data into the python file, inside a variable called family
* Print nicely the details about Jane's children
* Inside the family variable, add to each children, a key 'favorite\_color' with a value
* Then, save back all the new data into the json file
* Use the indent argument inside the dump function. Check out the documentation and the video in the **Useful Resources**

Run the python file