### **Краткое руководство по использованию JSON в Python**

[JSON](http://json.org/) (JavaScript Object Notation) это легковесный формат обмена данными. Людям его легко читать и вести в нем записи, а компьютеры запросто справляются с его синтаксическим анализом и генерацией.

JSON основан на языке программирования JavaScript. Но этот текстовый формат не зависит от языка и среди прочих может использоваться в Python и Perl. В основном его применяют для передачи данных между сервером и веб-приложением.

JSON (JavaScript Object Notation-формат передачи данных), определен стандартом [**RFC 7159**](https://tools.ietf.org/html/rfc7159.html) (который вытекает из устаревшей версии [**RFC 4627**](https://tools.ietf.org/html/rfc4627.html)) и [**ECMA-404**](https://www.ecma-international.org/publications/standards/Ecma-404.htm), легким текстовым форматом обмена данных, основанным на синтаксисе литералов объекта JavaScript (хотя он и не является подгруппой JavaScript).

JSON построен на двух структурах:

* Набор пар «имя-значение». Они могут быть реализованы как объект, запись, словарь, хеш-таблица, список «ключей-значений» или ассоциативный массив.
* Упорядоченный список значений. Его реализуют в виде массива, вектора, списка или последовательности.

### **JSON в Python**

В Python есть ряд пакетов, поддерживающих JSON, в частности [metamagic.json](https://pypi.org/project/metamagic.json/), [jyson](http://opensource.xhaus.com/projects/jyson/wiki), [simplejson](https://simplejson.readthedocs.io/en/latest/), [Yajl-Py](http://pykler.github.io/yajl-py/), [ultrajson](https://github.com/esnme/ultrajson), и [json](https://docs.python.org/3.6/library/json.html). В этом руководстве мы будем использовать json, имеющий «родную» поддержку в Python. Для проверки данных JSON мы можем воспользоваться [этим сайтом,](https://jsonlint.com/)предоставляющим JSON-линтер.

Ниже приведен пример записи JSON. Как видим, представление данных очень похоже на словари Python.

{

"article": [

{

"id": "01",

"language": "JSON",

"edition": "first",

"author": "Derrick Mwiti"

},

{

"id": "02",

"language": "Python",

"edition": "second",

"author": "Derrick Mwiti"

}

],

"blog": [

{

"name": "Datacamp",

"URL": "datacamp.com"

}

]

}

### **Конвертируем JSON в объекты Python**

Вышеуказанную JSON-строку мы можем спарсить при помощи метода json.loads() из модуля json. В итоге получим словарь Python.

import json

to\_python = json.loads(my\_json\_string)

### **Конвертируем объекты Python в JSON**

Используя json.dumps(), мы можем сконвертировать объекты Python в формат JSON.

blog = {'URL': 'datacamp.com', 'name': 'Datacamp'}

to\_json= json.dumps(blog)

to\_json

'{"URL": "datacamp.com", "name": "Datacamp"}'

Теперь давайте сравним типы данных в Python и JSON.

|  |  |
| --- | --- |
| **Python** | **JSON** |
| dict | Object |
| list | Array |
| tuple | Array |
| str | String |
| int | Number |
| float | Number |
| True | true |
| False | false |
| None | null |

Ниже мы покажем, как сконвертировать некоторые объекты Python в типы данных JSON.

#### **Кортеж Python — в массив JSON**

tuple\_example = 'Mango', 'Banana', 'Apple'

print(json.dumps(tuple\_example))

["Mango", "Banana", "Apple"]

#### **Список Python — в массив JSON**

list\_example = ["Mango", 1, 3, 6, "Oranges"]

print(json.dumps(list\_example))

["Mango", 1, 3, 6, "Oranges"]

#### **Строка Python — в строку JSON**

string\_example = "This is a cool example."

print(json.dumps(string\_example))

"This is a cool example."

#### **Булевы значения Python — в булевы значения JSON**

boolean\_value = **False**

print(json.dumps(boolean\_value))

false

### **Запись в файл JSON**

Модуль json позволяет также записывать данные JSON в файл. Такие файлы сохраняют с расширением .json.

**with** open('test\_file.json', 'w') **as** file:

json.dump(my\_json\_string, file)

### **Чтение файлов JSON**

Теперь продемонстрируем, как прочитать только что созданный нами файл JSON. Для его загрузки вызовем json.load().

**with** open('test\_file.json', 'r') **as** j:

json\_data = json.load(j)

print(json\_data)

### **Ограничения имплементации**

Процесс кодирования в JSON называется **сериализацией**, а декодирования – **десериализацией**. Некоторые реализации десериализаторов имеют ограничения на:

* размер принимаемых текстов JSON
* максимальный уровень вложенности объектов и массивов JSON
* диапазон точности чисел JSON
* содержание и максимальную длину строк JSON.

Впрочем, подобные ограничения связаны только с типами данных Python и работой самого интерпретатора Python.