http://pythonworld.ru/moduli/modul-json.html#more-966

JSON (JavaScript Object Notation) - простой формат обмена данными, основанный на подмножестве синтаксиса JavaScript. Модуль json позволяет кодировать и декодировать данные в удобном формате.

Кодирование основных объектов Python:

>>> import json

>>> json.dumps(['foo', {'bar': ('baz', None, 1.0, 2)}])

'["foo", {"bar": ["baz", null, 1.0, 2]}]'

>>> print(json.dumps("\"foo\bar"))

"\"foo\bar"

>>> print(json.dumps('\u1234'))

"\u1234"

>>> print(json.dumps('\\'))

"\\"

>>> print(json.dumps({"c": 0, "b": 0, "a": 0}, sort\_keys=True))

{"a": 0, "b": 0, "c": 0}

Компактное кодирование:

>>> import json

>>> json.dumps([1,2,3,{'4': 5, '6': 7}], separators=(',', ':'))

'[1,2,3,{"4":5,"6":7}]'

Красивый вывод:

>>> import json

>>> print(json.dumps({'4': 5, '6': 7}, sort\_keys=True, indent=4))

{

"4": 5,

"6": 7

}

Декодирование JSON:

>>> import json

>>> json.loads('["foo", {"bar":["baz", null, 1.0, 2]}]')

['foo', {'bar': ['baz', None, 1.0, 2]}]

>>> json.loads('"\\"foo\\bar"')

'"foo\x08ar'

**Основы**

**json.dump**(obj, fp, skipkeys=False, ensure\_ascii=True, check\_circular=True, allow\_nan=True, cls=None, indent=None, separators=None, default=None, sort\_keys=False, \*\*kw) - сериализует obj как форматированный JSON поток в fp.

Если skipkeys = True, то ключи словаря не базового типа (str, unicode, int, long, float, bool, None) будут проигнорированы, вместо того, чтобы вызывать исключение TypeError.

Если ensure\_ascii = True, все не-ASCII символы в выводе будут экранированы последовательностями \uXXXX, и результатом будет строка, содержащая только ASCII символы. Если ensure\_ascii = False, строки запишутся как есть.

Если check\_circular = False, то проверка циклических ссылок будет пропущена, а такие ссылки будут вызывать OverflowError.

Если allow\_nan = False, при попытке сериализовать значение с запятой, выходящее за допустимые пределы, будет вызываться ValueError (nan, inf, -inf) в строгом соответствии со спецификацией JSON, вместо того, чтобы использовать эквиваленты из JavaScript (NaN, Infinity, -Infinity).

Если indent является неотрицательным числом, то массивы и объекты в JSON будут выводиться с этим уровнем отступа. Если уровень отступа 0, отрицательный или "", то вместо этого будут просто использоваться новые строки. Значение по умолчанию None отражает наиболее компактное представление. Если indent - строка, то она и будет использоваться в качестве отступа.

Если sort\_keys = True, то ключи выводимого словаря будут отсортированы.

**json.dumps**(obj, skipkeys=False, ensure\_ascii=True, check\_circular=True, allow\_nan=True, cls=None, indent=None, separators=None, default=None, sort\_keys=False, \*\*kw) - сериализует obj в строку JSON-формата.

Аргументы имеют то же значение, что и для dump().

Ключи в парах ключ/значение в JSON всегда являются строками. Когда словарь конвертируется в JSON, все ключи словаря преобразовываются в строки. В результате этого, если словарь сначала преобразовать в JSON, а потом обратно в словарь, то можно не получить словарь, идентичный исходному. Другими словами, loads(dumps(x)) != x, если x имеет нестроковые ключи.

**json.load**(fp, cls=None, object\_hook=None, parse\_float=None, parse\_int=None, parse\_constant=None, object\_pairs\_hook=None, \*\*kw) - десериализует JSON из fp.

object\_hook - опциональная функция, которая применяется к результату декодирования объекта (dict). Использоваться будет значение, возвращаемое этой функцией, а не полученный словарь.

object\_pairs\_hook - опциональная функция, которая применяется к результату декодирования объекта с определённой последовательностью пар ключ/значение. Будет использован результат, возвращаемый функцией, вместо исходного словаря. Если задан так же object\_hook, то приоритет отдаётся object\_pairs\_hook.

parse\_float, если определён, будет вызван для каждого значения JSON с плавающей точкой. По умолчанию, это эквивалентно float(num\_str).

parse\_int, если определён, будет вызван для строки JSON с числовым значением. По умолчанию эквивалентно int(num\_str).

parse\_constant, если определён, будет вызван для следующих строк: "-Infinity", "Infinity", "NaN". Может быть использовано для возбуждения исключений при обнаружении ошибочных чисел JSON.

Если не удастся десериализовать JSON, будет возбуждено исключение ValueError.

**json.loads**(s, encoding=None, cls=None, object\_hook=None, parse\_float=None, parse\_int=None, parse\_constant=None, object\_pairs\_hook=None, \*\*kw) - десериализует s (экземпляр str, содержащий документ JSON) в объект Python.

Остальные аргументы аналогичны аргументам в load().

**Кодировщики и декодировщики**

Класс **json.JSONDecoder**(object\_hook=None, parse\_float=None, parse\_int=None, parse\_constant=None, strict=True, object\_pairs\_hook=None) - простой декодер JSON.

Выполняет следующие преобразования при декодировании:

|  |  |
| --- | --- |
| **JSON** | **Python** |
| object | dict |
| array | list |
| string | str |
| number (int) | int |
| number (real) | float |
| true | True |
| false | False |
| null | None |

Он также понимает NaN, Infinity, и -Infinity как соответствующие значения float, которые находятся за пределами спецификации JSON.

Класс **json.JSONEncoder**(skipkeys=False, ensure\_ascii=True, check\_circular=True, allow\_nan=True, sort\_keys=False, indent=None, separators=None, default=None)

Расширяемый кодировщик JSON для структур данных Python. Поддерживает следующие объекты и типы данных по умолчанию:

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