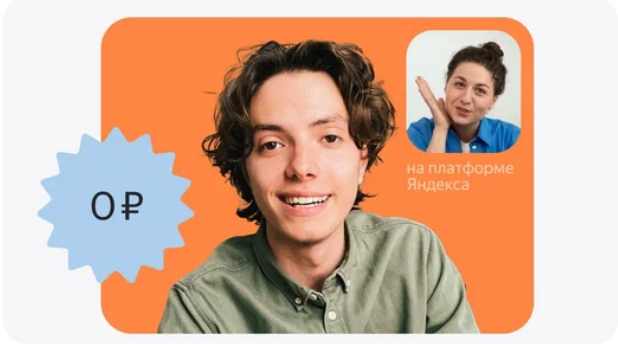


# Модуль json в Python, работа с JSON форматом



practicum.yandex.ru

РЕКЛАМА · 18+ Я

**Бесплатное занятие английским в Яндекс Практикуме**  
Полноценное занятие с преподавателем, а не презентация курсов  
[Узнать больше](#)

[Стандартная библиотека Python3.](#) / Модуль json в Python, работа с JSON форматом

## Чтение и парсинг json формата, запись данных в файл

JavaScript Object Notation (JSON) - это независимый от языка формат обмена данными, представленный в текстовом виде и понятный человеку. Он был получен из стандарта языка программирования ECMAScript. Формат JSON определяет маленький набор правил форматирования для переносимого представления структурированных данных.

Информация в формате JSON может быть представлена в двух видах:

- Последовательность пар с ключами и соответствующими им значениями, подобно [словарям](#);
- Просто упорядоченный набор значений, подобно [спискам](#).

[Модуль json](#) предоставляет API, подобный стандартным библиотечным модулям [piclike](#).

**Примечания:**

- JSON - это подмножество YAML 1.2. Объект JSON, создаваемый с значением разделителей по умолчанию, также является подмножеством YAML 1.0 и 1.1. Таким образом, этот модуль также можно использовать в качестве сериализатора YAML.
- Сериализатор и десериализатор этого модуля по умолчанию сохраняют порядок ввода и вывода элементов. Порядок теряется только в том случае, если вложенные контейнеры неупорядочены.

## Стандартное соответствие и совместимость:

Этот модуль НЕ строго соответствует RFC, реализуя некоторые расширения, которые являются допустимым JavaScript, но не допустимым JSON.

- Бесконечные и NaN числовые значения принимаются и выводятся;
- Повторные имена внутри объекта принимаются и используется только значение последней пары name:value.

Поскольку RFC позволяет RFC-совместимым синтаксическим анализаторам принимать входные тексты, не соответствующие RFC, десериализатор этого модуля технически RFC-совместим при настройках по умолчанию.

## Бесконечные числовые значения и NaN:

RFC не допускает представление бесконечных или числовых значений NaN. Несмотря на это, по умолчанию этот модуль принимает и выводит Infinity, -Infinity и NaN, как если бы они были действительными значениями литералов числа JSON:

```
# Ни один из этих вызовов не вызывает исключение,  
# но результаты не являются допустимыми JSON  
>>> import json  
>>> json.dumps(float('-inf'))  
# '-Infinity'  
>>> json.dumps(float('nan'))  
# 'NaN'  
>>> # Same when deserializing  
>>> json.loads('-Infinity')  
# -inf  
>>> json.loads('NaN')  
#
```

Вверх

- В сериализаторе параметр `allow_nan` может использоваться для изменения этого поведения.
- В десериализаторе параметр `parse_constant` может использоваться для изменения этого поведения.

## Повторяющиеся имена внутри объекта:

RFC указывает, что имена в объекте JSON должны быть уникальными и не повторяются, но не предписывает, как они должны обрабатываться в объектах JSON. По умолчанию этот [модуль json](#) не вызывает исключение, вместо этого он игнорирует все, кроме последней пары имя-значение для данного имени:

```
>>> import json
>>> weird_json = '{"x": 1, "x": 2, "x": 3}'
>>> json.loads(weird_json)
{'x': 3}
```

Параметр `object_pairs_hook` можно использовать для изменения этого поведения.

## Кодировки символов:

RFC требует, чтобы JSON был представлен с использованием UTF-8, UTF-16 или UTF-32, при этом UTF-8 является рекомендуемым значением по умолчанию для максимальной совместимости.

В соответствии с разрешением, хотя и не обязательным для RFC, сериализатор этого модуля по умолчанию задает значение `sure_ascii=True`, таким образом экранируя выходные данные, поэтому результирующие строки содержат только символы ASCII.

Помимо параметра `sure_ascii`, этот модуль определен строго с точки зрения преобразования между объектами Python и [строками Unicode](#), в противном случае напрямую не решает проблему кодировки символов.

RFC запрещает добавлять метку порядка байтов (BOM) в начало текста JSON, сериализатор этого модуля не добавляет BOM в свой вывод. RFC разрешает, но не требует от десериализаторов JSON игнорировать начальную спецификацию в своих входных данных. Десериализатор модуля `json` вызывает [ошибку ValueError](#), когда присутствует начальная спецификация.

RFC явно не запрещает строки JSON, которые содержат последовательности байтов, которые не соответствуют действительным символам Unicode, например непарные суррогаты UTF-16, но отмечает, что они могут вызывать проблемы взаимодействия. По умолчанию этот модуль принимает и выводит, если присутствует в исходной строке, кодовые точки для таких последовательностей.

### Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Приемы работы с модулем json](#)
- [Функция dump\(\) модуля json](#)
- [Функция dumps\(\) модуля json](#)
- [Функция load\(\) модуля json](#)
- [Функция loads\(\) модуля json](#)
- [Класс JSONDecoder\(\) модуля json](#)
- [Функция JSONEncoder\(\) модуля json](#)
- [Исключение JSONDecodeError\(\) модуля json](#)
- [Интерфейс командной строки модуля json](#)

ХОЧУ ПОМОЧЬ  
ПРОЕКТУ

Вверх



DOCS-Python.ru™, 2023 г.

(Внимание! При копировании материала ссылка на источник обязательна)

@docs\_python\_ru

Вверх