JSON (JavaScript Object Notation), зазначений в RFC 7159 (як Obsoletes RFC 4627) і ECMA-404, являє собою легкий формат обміну даними, який походить від мови JavaScript і використовує його синтаксис. JSON виставляє API, знайомий користувачам стандартної бібліотеки marshal і pickle.

**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)**

Серіалізація OBJ як JSON-форматованого потоку до FP (а .WRITE () - файло-подібний об'єкт). Якщо skipkeys True (за замовчуванням: False), то Dict ключі, що не є базовим типом (вулиця, ціле число, дробове, булеве(1, 0), None) будуть пропущені, а не викликати TypeError. Модуль завжди справляє JSON StR об'єкти, а не байти об'єктів. Тому, fp.write () повинен підтримувати .str об’єкт. Якщо ensure\_ascii істинна (за замовчуванням), вивід гарантовано матиме всі вхідні символи НЕ-ASCII. Якщо ensure\_ascii фальш, ці символи будуть виводитися як є. Якщо check\_circular фальш (за замовчуванням: True), то циклічна перевірка типів контейнерів буде пропущена, то циклічне посилання приведе до OverflowError (або гірше). Якщо allow\_nan фальш (за замовчуванням: True), то це викличе ValueError з значень діапазону з плаваючою точкою (нан, інф, -inf) в суворій відповідності специфікації JSON. Якщо allow\_nan істина, то буде використовуватися їх еквіваленти JavaScript (NaN, нескінченність, нескінченність). Якщо відступ є невід'ємним цілим числом або рядком, то елементи масиву JSON і члени об'єкта будуть виведені з цим рівнем відступу. Рівень відступу 0 або «» буде вставляти нові рядки. None (за замовчуванням) вибирає найбільш компактне уявлення. Якщо відступ є рядок (наприклад, «\ т»), цей рядок використовується для відступу на кожному рівні. Якщо вказано що, сепаратори повинні бути (item\_separator, key\_separator) використовуватиметься кортеж. За замовчуванням ( '', ':'), якщо відступи не є None і ( ',', ':') в іншому випадку. Щоб отримати найбільш компактне представлення JSON, слід вказати ( «», «:») для усунення прогалин.

**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)**

Серіалізація OBJ в JSON відформатований str. Аргументи мають той же сенс, що і в dump ()

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

Deserialize FP (а .read () - підтримка файлу, як об'єкт, що містить документ JSON) на об'єкт Python. object\_hook є додатковою функцією, яка буде викликана в результаті якого-небудь об'єкта буквальним декодуванням словника. Значення, що повертається object\_hook буде використовуватися замість словника. Ця функція може бути використана для реалізації користувальницьких декодерів (наприклад, JSON-RPC класу). object\_pairs\_hook є додатковою функцією, яка буде викликана в результаті якого-небудь об'єкта буквального декодованого з впорядкованим списком пар. Значення, що повертає object\_pairs\_hook буде використовуватися замість словника. Ця функція може бути використана для реалізації користувальницьких декодерів, які покладаються на порядок, так як і ключ, значення пари декодованого (наприклад, collections.OrderedDict () буде пам'ятати порядок вставки). Якщо object\_hook також визначено, object\_pairs\_hook має більшу пріріотетність. parse\_float, якщо вказано, буде викликатися з рядока кожного JSON integer і буде розшифрованим. За замовчуванням, це еквівалентно int(num\_str). Це може бути використано, щоб використовувати інший тип даних або аналізатор для JSON integer (наприклад, decimal.Decimal). parse\_int, якщо вказано, буде викликатися з струна кожного JSON int для декодування. За замовчуванням, це еквівалентно int (num\_str), що може бути використано, щоб використовувати інший тип даних, або аналізатор для JSON цілих чисел з плаваючою точкою.  
parse\_constant, якщо вказано, буде називатися однією з наступних рядків: «-Infinity», «Infinity», «None». Це може бути використано для генерування винятків, якщо недійсні числа JSON зустрічаються. Щоб використовувати користувальницький підклас JSONDecoder, вкажіть його з аргументом kwarg; в іншому випадку використайте JSONDecoder. Додаткові аргументи ключового слова будуть передані в конструктор класу. Якщо дані десеріалізовані і не є дійсним документом у форматі JSON, буде викликакно JSONDecodeError .

**json.loads(s, \*, encoding=None, cls=None, object\_hook=None, parse\_float=None, parse\_int=None, parse\_constant=None, object\_pairs\_hook=None, \*\*kw)**

Deserialize це (а, int byte або ByteArray екземпляр, що містить документ JSON) і трансформує в об'єкт Python. Інші аргументи мають таке ж значення, як і в навантаженні (), за винятком кодування, який ігнорується. Якщо дані десеріалізовані і не є дійсним документом у форматі JSON, буде викликано JSONDecodeError. Змінено в версії 3.6: s тепер може бути ByteType або ByteArray. Кодування вхідного сигналу повинне бути UTF-8, UTF-16 або UTF-32.

json.JSONDecoder**(*\**, *object\_hook=None*, *parse\_float=None*, *parse\_int=None*, *parse\_constant=None*, *strict=True*, *object\_pairs\_hook=None*)**

object\_hook, якщо вказано, буде викликатись з результатом кожного об'єкта JSON декодується і її значення, що повертається буде використовуватися замість даного словника. Це може бути використано, щоб забезпечити для користувача deserializations (наприклад, для підтримки JSON-RPC класу hinting). object\_pairs\_hook, якщо вказано буде викликатись з результатом кожного об'єкта JSON декодованого з впорядкованим списком пар. Значення, що повертається object\_pairs\_hook буде використовуватися замість словника. Ця функція може бути використана для реалізації користувальницьких декодерів, які покладаються на порядок, що ключ і значення пари декодованого (наприклад, collections.OrderedDict () буде пам'ятати порядок вставки). Якщо object\_hook також визначено, object\_pairs\_hook має вищий пріріотет. parse\_float, якщо вказано, буде викликатись з рядоком кожного JSON integer. За замовчуванням, це еквівалентно int (num\_str). Це може бути використано, для іншого типу даних або аналізатора для JSON integer (наприклад, decimal.Decimal). parse\_int, якщо вказано, буде викликатись з стрічкою кожного JSON int для декодування. За замовчуванням, це еквівалентно int (num\_str). Це може бути використано, щоб використовувати інший тип даних або аналізатор для JSON цілих чисел з плаваючою точкою. parse\_constant, якщо вказано, буде називатися одним з наступних рядків: «-Infinity», «Infinity», «None». Це може бути використано для викликання винятока, якщо недійсні числа JSON зустрічаються. Якщо False(True за замовчуванням), то керуючі символи будуть дозволені всередині рядка. Керуючі символи в цьому контексті є ті, з символьними кодами в діапазоні 0-31, включаючи «\» т (табл), «\ п», «\ г» і «\ 0». Якщо дані десеріалізовані і не є дійсним документом у форматі JSON, буде викликано JSONDecodeError.