Функціональні можливості пакету JSON

Пакет JSON має чимало різних функцій, які повертають об’єкти типу даних str, однією з яких є: 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)

Ця функція серіалізує obj як форматований json потік в fp, використовуючи шифрувальну таблицю JSON для структур даних Python.

skipkeys – ключові слова нестандартних типів будуть ігноруватися(якщо = False(за дефолтом)), або буде збуджуватися TypeError(якщо = True)

ensure\_ascii – повертає всі вхідні дані без не-ASCII символів, якщо дорівнює True(за дефолтом). В протилежному випадку просто виводить отримані дані.

check\_circular – якщо дорівнює False(за дефолтом True), то ціклічна перевірка з типом контейнер буде пропущена, а такі циклічні згвдування збуджуватимуть OverflowError або гріше.

allow\_nan - якщо дорівнює False(за дефолтом True), то буде збуджуватися ValueError, щоб серіалізувати змінні з плавачою крапочкою (nan, inf, -inf), які вийшли за межі, згідно суворої специфікації JSON. Якщо дорівнює True,то будуть використовуватися їхні JavaScript еквіваленти (NaN, Infinity, -Infinity) .

Якщо indent є невід'ємним цілим числом або рядком, то елементи масиву JSON і члени об'єкта будуть гарно надруковані з цим рівнем відступу. Якщо indent - 0, від’ємне число, або «», то будуть тільки вставлені нові рядки. None (за замовчуванням) вибирає найбільш компактну презентацію. Використовуючи позитивне ціле число indent відступає стільки пробілів у рівні. Якщо indent є рядок (наприклад, «\ t»), то цей рядок використовується для відступу в кожному рівні.

Якщо вказано, сепаратори повинні бути (item\_separator, key\_separator) кортежем. За замовчуванням ('', ':'), якщо indent є None, і (',', ':') в іншому випадку. Щоб отримати найбільш компактне представлення JSON, слід вказати («», «:») для усунення прогалин.

Якщо вказано, за замовчуванням повинна бути функція, яка викликається для об'єктів, які не можуть бути серіалізовані. Вона повина повертати закодовану версію JSON об'єкта або збуджувати TypeError. Якщо не вказано, збуджується TypeError.

Якщо sort\_keys True (за замовчуванням: False), то вихідні дані словників будуть відсортовані за ключем.

Щоб використовувати підклас JSONEncoder, потрібно вказати його з cls kwarg; або JSONEncoder використовується.

Ще одна така функція: 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)

Серіалізує obj в форматований JSON str використовуючи шифрувальну таблицю JSON для структур даних Python. Аргументи мають таке саме знаяення як і в функції .dump().

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

Десеріалізує fp в Python об’єкт використовуючи шифрувальну таблицю JSON для структур даних Python.

object\_hook - додаткова функція, яка застосовується до результату декодування об'єкта (dict). Використовуватися буде значення, яке повертає ця функція, а не отриманий словник.

object\_pairs\_hook - додаткова функцією, яка викликається в результаті якого-небудь декодування об'єкта з впорядкованим списком пар. Значення, що повертається object\_pairs\_hook буде використовуватися замість dict. Ця функція може бути використана для реалізації користувальницьких декодерів, які покладаються на порядок, в якому ключ і значення пари декодуються. Якщо object\_hook є також визначений, object\_pairs\_hook має вищий пріоритет.

parse\_float, якщо вказаний, буде викликатися з рядок кожного JSON числа з плаваючою крапкою для розшифрування. За замовчуванням, це еквівалентно float (num\_str). Це може бути використано, щоб використовувати інший тип даних або аналізатор для JSON чисел з плаваючою крапкою.

parse\_int, якщо вказаний, буде викликатися з рядком кожного JSON цілого числа для декодування. За замовчуванням, це еквівалентно int(num\_str). Це може бути використано, щоб використовувати інший тип даних або аналізатор для JSON цілих чисел.

parse\_constant, якщо вказаний, буде викликатися з одним із наступних рядків '-Infinity', 'Infinity', 'NaN'. Це може бути використано для збудження винятку, якщо неправильні числа JSON зустрічаються.

Щоб використовувати підклас JSONDecoder, вкажіть його з cls kwarg; в іншому випадку JSONDecoder is used. Додаткові аргументи ключового слова будуть передані конструктору класу.

Якщо десеріалізовані дані не є дійсним документ у форматі JSON, буде збуджено JSONDecodeError.

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

Десеріалізовані s d об'єкт Python, використовуючи шифрувальну таблицю JSON для структур даних Python.

Інші аргументи мають таке ж значення, що і в load(), за винятком кодування, яке ігнорується.

Якщо десеріалізовані дані не є дійсним документ у форматі JSON, буде збуджено JSONDecodeError.

Також JSON має реалізовано декілька класів, один із них:

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

Це простий JSON декодер, в основі якого лежить шифрувальна таблиця JSON для структур даних Python.

Він також розуміє, NaN, Infinity, and -Infinity, як їхні відповідні значення із плаваючою крапочкою, які є за межами специфікації JSON.

object\_hook, якщо вказаний, буде викликатися з результатом кожного JSON декодованого об’єкту та його повернене значення буде використане замість [dict](https://docs.python.org/3.5/library/stdtypes.html#dict). Це може бути використано, щоб забезпечити для користувача десеріалізацію.

object\_pairs\_hook якщо вказаний, буде викликатися з результатом кожного JSON декодованого об’єкту в порядку списку пар. Повернене значення object\_pairs\_hook використовується замість dict. Може бути використаний для декодування, яке залежить від ключа і значення пар, які декодуються. Якщо object\_hook є також вказаний, object\_pairs\_hook має більший пріоритет.

parse\_float викликається з кожним рядком числа з плавачою крапкою JSON, щоб бути розшифрованим. За замовчуванням, це еквівалентно плавати (num\_str). Може бути використаний для використання іншого типу даних або аналізатором для JSON чисел з плаваючою крапкою.

parse\_int викликається з кожним рядком цілого числа JSON, щоб бути розшифрованим. Може бути використаний для використання іншого типу даних або аналізатором для JSON цілих чисел.

parse\_constant буде викликатися з одним із цих рядків:  '-Infinity', 'Infinity', 'NaN'. Також може бути використаним для збудження винятку, якщо неправильні JSON числа є закодовані.

Функція decode(s) – повертає Python репрезентацію параметру s. Збуджує JSONDecodeError, якщо JSON документ є недійсний.

Функція raw\_decode(s) – розшифровує JSON документ з s, і повертає кортеж Python репрезентації та індекс з s, де документ закінчується. Використовується для декодування документа JSON з даними типу str, які можуть мати сторонні дані в кінці.

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

Екстенсивний JSON шифратор для структур даних Python.

default(o) - Реалізовуйте цей метод в підкласі таким чином, що він повертає об'єкт серіалізації для о, або збуджує TypeError.

encode(o) - Повертає JSON str представлення структури даних Python

iterencode(o) - Кодує об’єкт o ,і надає str репрезентацію

Також в пакеті JSON реалізований клас винятків json.JSONDecodeError(msg, doc, pos, end=None)

 Це підклас [ValueError](https://docs.python.org/3.5/library/exceptions.html#ValueError) з додатковими атрибутами:

msg – повідомлення невідформатованої помилки

doc – JSON документ, який аналізувався

pos – початковий індекс документа, де аналіз не вдався

lineno – лінія, що відповідає за pos

colno – стовпчик, що відповідає позиції