Опис json формату

На сьогодні багато api повертають дані в форматі json (JavaScript Object Notation), за допомогою якого можна кодувати та декодувати певні об'єкти.

Основними функціями python бібліотеки json є dump, dumps, load, loads.

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)

Конвертує об'єкт(obj) в стрічку json формату використовуючи таблицю перетворення.

Параметри:

1) **skipkeys**

Визначає поведінку при конвертуванні не базових типів(базові типи: str, int, float, bool, None):

значення False: ключі будуть пропущені
значення True: буде збуджено TypeError

2) **ensure_ascii**

Визначає дії з символами, які не є в таблиці ascii:

значення True: будуть виведені escape послідовності

значення False: будуть виведені як є

3) **check_circular**

Визначає дії з циклічними посиланнями:

значення False: циклічні посилання будуть пропущені, що може призвести до OverflowError

4) **allow_nan**

Визначає поведінку з кодуванням великих чисел:

значення True: будуть використані (NaN, Infinity, -Infinity)

значення False: буде збуджено ValueError

5) **indent(int)**

Якщо це значення - невід'ємне ціле число, то кожний елемент масиву буде відображатися з нового рядка з відступом indent.

Якщо значення - None - не будуть ставитись символів нового рядка

6) **separator**

Значення - tuple(str, str) - визначає item_separator та key_separator

7) **sort_keys**

Значення True: буде повернуто об'єкт з посортованими ключами

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)

fp - file-like object, який підтримує метод .write(). Записує в fp конвертований в json формат об'єкт. Аргументи див json.dumps

json.loads(s,

encoding=None,

cls=None,

object_hook=None,

parse_float=None,

parse_int=None,

parse_constant=None,

object_pairs_hook=None, **kw)

Перетворює стрічку s, записану в json форматі, в python об'єкт, використовуючи порівняльну таблицю. Параметри:

1) **encoding(str)** - кодування стрічки s
2) **cls** - вказується при використанні власного класу декодера. Якщо значення даного параметра None, то буде використано клас JSONDecoder.
3) **object_hook**
   Необов'язкова функція, яка буде виконана над значенням, яке повертає json.loads (dict). Значення, яке повертається, буде замінено на результат виконання цієї функції.
4) **parse_float**
   Функція, яка буде викликана при обробці кожного числа з плаваючою крапкою.
   Якщо значення параметру None, то буде викликатися функція float(num_str).
5) **parse_int**
   Аналогічно до попереднього параметру можна задати функцію, яка буде викликатися при обробці цілих чисел.
6) **parse_constant**
   Функція, яка буде викликатися при обробці '-Infinity', 'Infinity' або 'NaN'

json.load(fp,

cls=None,

object_hook=None,

parse_float=None,

parse_int=None,

parse_constant=None,

object_pairs_hook=None,

**kw)

fp - file-like object, який підтримує метод .read(). Переворює вміст файлу в python об'єкт (аргументи див json.loads).

Винятки

exception json.JSONDecodeError - підклас ValueError, який має наступні атрибути:

1) **msg** - повідомлення про помилку
2) **doc** - json документ, який мав бути перетворений в python об'єкт
3) **pos** - індекс позиції в doc, де відбулася помилка
4) **lineno** - номер лінії, який відповідає pos
5) **colno** - номер колонки, який відповідає pos