JSON – це текстовий формат обміну даних, простий та зручний для використання. Його просто писати і читати адже він написаний читабельним для людей текстом. Він базується на мові програмування JavaScript і був розроблений Дугласом Крокфордом, проте цей текстовий формат повністю незалежний від JavaScript і може використовуватись практично в будь-якій мові програмування.

Модуль json дозволяє кодувати і декодувати дані в зручному форматі

**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) – серіалізує (серіалізація – процес перетворення структури даних в послідовність бітів) obj як форматований JSON потік в fp.

Якщо skipkeys = True, то ключі словника не базового типу (str, unicode, int, long, float, bool, None) будуть проігноровані, замість того, щоб викликати виняток TypeError.

Якщо ensure\_ascii = True, всі не ASCII символи при вводі будуть зображені послідовністю \uXXXX, і результатом буде стрічка, яка містить тільки ASCII символи. Інакше, якщо ensure\_ascii = False, то стрічки не зміняться.

Якщо check\_circular = False, то перевірка циклічних посилань буде пропущена, а такі посилання будуть викликати OverflowError.

Якщо allow\_nan = False, то при спробі серіалізувати значення з комою, які виходять за допустимі межі, буде викликатись ValueError (nan, inf, -inf) в строгій відповідності зі специфікацією JSON, замість того щоб використовувати еквіваленти із JavaScript (NaN, Infinity, -Infinity).

Якщо indent є невід’ємним числом, то масиви і об’єкти в JSON будуть виводитися з цим рівнем відступу. Якщо рівень відступу 0, від’ємний або ‘’, то замість цього просто будуть використовуватись нові стрічки. Типове значення None відображає компактне представлення. Якщо indent – стрічка, то вона і буде використовуватись в якості відступу.

Якщо sort\_keys = True, то ключі вихідного словника будуть відсортвані.

**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 стрічку JSON-формату.

Аргументи мають таке ж значення як і для dump().

\*

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

object\_hook – опціональна функція, яка застосовується до результату декодування об’єкта (dict). Використовуватись буде значення, яке повертається цією функцією, а не отриманий словник.

object\_pairs\_hook – опціональна функція, яка застосовується до результату декодування об’єкта с конкретної послідовності пар ключ/значення. Буде використано результат, який повертається функцією, замість вхідного словника. Ця функція може бути використана, щоб забезпечити інструментами декодери, які покладаються на порядок, яким ключ і пари значень є розкодовані (на приклад collections.OrderedDict() пам’ятатиме порядок вводу). Якщо objet\_hook також заданий, то пріорітет віддається object\_hook\_pairs.

parse\_float, якщо визначений, буде викликаний для кожного значення JSON із плаваючою крапкою. Стандартно, він еквівалентний float(num\_str). Це може бути використано щоб використовувати інакший тип даних або аналізатор для JSON floats(e.g. decimal.Decimal).

parse\_int, якщо визначений, то буде викликаний для стрічки JSON із числовим значенням. Стандартно, він еквівалентний int(num\_int). Це може бути використано щоб використовувати інакший тип даних або аналізатор для JSON integers(e.g. float).

parse\_constant, якщо визначений, буде викликаний для наступних стрічок: "-Infinity", "Infinity", "NaN". Може бути використано для пробудження виключень при виявленні помилкових чисел JSON.

Якщо не вдасться десеріалізувати JSON, буде пробуджено виключення 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) – десеріалізує s(екземпляр str, який містить документ JSON) в об’єкт Python.

Решта аргументів аналогічні load().

**Кодування основних об’єктів в 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}

**Компактне кодування:**

>>> import json

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

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

**Декодування(парсинг) 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'