**Json**

Json – це простий формат обміну даними(створений за ідеєю JavaScript). Json допомагає користувачу у роботі із АРІ.

**Використання**

**Функціонал:**

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):

Серіалізовує OBJ як Json-потік до fp.

Якщо skipkeys правдиві, то dict ключі з небазовим типом будуть пропущенні, а помилка типу TypeError не буде викликана. Json модуль надає користувачу об’єкти типу str, а не типу bytes. Тому fp.write () має підтримувати ввід str.

Якщо ensure\_ascii=true, то всі вхідні символи НЕ-ASCII будуть пропущені. Якщо ensure\_ascii=false,то ці символи будуть виводитися як є.

Якщо check\_circular=false (за замовчуванням: True), то циклічна перевірка типів контейнерів буде пропущена і посилання приведе до OverflowError.

Якщо allow\_nan=false (за замовчуванням: True), то це викличе ValueError при серіалізуванні значень з плаваючою точкою у відповідності до специфікації JSON.

Якщо sort\_keys=true (за замовчуванням: False), то вихід словників буде відсортований по ключу.

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

object\_hook є додатковою функцією, яка буде викликана в результаті декодування об'єкта. Значення, що повертається object\_hook буде використовуватися замість Dict. Ця функція може бути використана для реалізації декодерів.

object\_pairs\_hook щось схоже до object\_hook. Функція буде викликана як результат декодування об‘єкту з впорядкованим списком пар. Значення, що повертаються object\_pairs\_hook буде використовуватися замість Dict. Ця функція може бути використана для реалізації користувальницьких декодерів, які покладаються на порядок.

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 str екземпляр, що містить документ JSON на об'єкт Python, використовуючи таблицю перетворення.

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

**Кодери і декодери**

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

Виконує наступні перетворення в декодуванні за замовчуванням відповідно:

**Json:**

object array string number (int) number (real) true false null

**Python:**

dict list str int float True False None

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

Кодер JSON для структур даних Python.

Підтримуються наступні об'єкти і типи за замовчуванням відповідно:

**Python:**

Dict list, tuple str int, float True False None

**Json:**

Object array string number true false null

encode(o)

Повертає стрічкове представлення JSON структури даних Python

iterencode(o)

Кодує даний об'єкт (о) і дає представлення кожній стрічці в мірі поступання.

**Винятки**

exception json.JSONDecodeError(msg, doc, pos, end=None)

Підклас ValueError з наступними додатковими атрибутами

**msg**

Неформатована помилка повідомлення

**doc**

JSON документ розбирається.

**pos**

Помилка індексу розбору документу.

**lineno**

Лінія відповідна до pos.

**colno**

Стовпець відповідний до pos.

**Кодування Знаків**

RFC вимагає, щоб JSON був представлений з використанням або UTF-8, UTF-16 або UTF-32. UTF-8 рекомендований за замовчуванням для максимальної сумісності.

В RFC, серіалізатор модуля встановлює ensure\_ascii = True за замовчуванням. Так отримані рядки містять тільки ASCII-символи.

Крім ensure\_ascii параметра, модуль визначається строго в термінах перетворення між об'єктами Python і рядками Unicode. Таким чином, безпосередньо вирішується проблема кодування.