Можливості пакету json.

Json(JavaScript Object Notation) – простий формат обміну даними, оснований на підмножині синтаксиса JAVASCRIPTAa. Модуль json позволяє кодувати і декодувати данні в зручному/потрібному 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) - серіалізуются obj як форматований JSON потік в fp.

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

Якщо ensure\_asci = True, всі не-ASCII символи в виводі будуть показані послідовностями \uXXXX, і результатом буде стрічка, яка складається тільки з ASCII символів. Якщо ensure asci = False, стрічки запишуться такими як є.

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

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

Якщо indent є невід’ємним числом, то масиви і об’єкти в JSON будуть виводитись з тим рівнем відступу. Якщо рівень відступу 0(indent = 0), від’ємний або indent = “”, то замість цього будуть просто використовуватись нові стрічки. Значення за замовчуванням None відображає найбільш компактне представлення. Якщо indent – стрічка, то вона і буде використовуватись як відступ.

Якщо sort keys = True, то клюві аутпутного словника будуть відсортовані.

**json.dumps**(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) - серіалізуются obj в стрічку JSON-формату.

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

Ключі в парах ключ/значення в JSON завжди є стрічками. Коли словник конвертується в JSON, всі ключі словника перетворюються в стрічки. В результаті цього, якщо словник перетворити в JSON, а потім назад в словник, не можливо получити словник, ідентичний вихідному. Іншими словами loads(dumps(x)) != х, якщо х має ключі, які не є стрічками.

**json.load**(fp, cls=None, object\_hook=None, parse\_float=None, parse\_int=None, parse\_constant=None, object\_pairs\_hook=None, \*\*kw) - десеріалізує JSON из 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 певною послідовністю пар ключ/значення. Буде використаний результат який повертає функція, замість вихідного словника. Якщо заданий також object\_hook то пріоритет віддається object\_pairs\_hook.

Parse\_float, якщо визначений, буде викликаний для кожного значення JSON з плаваючою крапкою. За замовчування це еквівалентно float(num\_str).

Parse\_int, якщо визначений, буде викликаний для стрічки JSON з числовим значенням. За замовчуванням еквівалентно до int(num\_str).

Parse\_constant, якщо визначений, буде викликаний для наступних стічок : = "-Infinity", "Infinity", "NaN". Може бути використаний для збудження винятків при найдених помилкових чисел JSON.

Якщо не вийде десеріалізувати JSON, буде збуджено виняток ValuseError.

**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 утримує документ JSON) в об’єкт Python.

Решта аргументів аналогічні аргументам в load().