2.2 Можливості пакету json

Перед тим як говорити про можливості пакету json, я думаю, що потрібно розібратися, що ж таке цей пакет json. Отож JSON або JavaScript Object Notation – це простий формат обміну даних, основаним на підмножині синтаксису JavaScript.

json.**dump**(obj, fp, skipkeys=False, ensure\_ascii=True, check\_circular=True, allow\_nan=True, cls=None, indent=None, separators=None, encoding="utf-8", default=None, sort\_keys=False, \*\*kw)

Сереалізує obj як форматований JSON потік в fp – файло-подібний об’єкт

Якщо skipkeys = True тоді ключі словника не базового типу будуть проігноровані, замість того, щоб викликати виключення TypeError.

Якщо ensure\_ascii = True всі не-ASCII символи в виводі будуть екранізовані послідовностями \uXXXX, і результатом буде стрічка, вміщаюча тільки ASCII символи. Якщо ensure\_ascii = False , то деякі частини, що записуються в fp можуть бути екземплярами **unicode.** Зазвичай це відбувається через те, що початковий потік вміщає юникод або використовує параметр ***encoding.*** Якщо ви не впевнені, що  fp.write()  розуміє юнікод, то це практично точно приведе до помилки.

Якщо check\_circular = False тоді перевірка циклічних ссилок для типів контейнера буде пропущена, а такі ссилки будуть викликати ***OverflowError.***

Якщо allow\_nan = False при спробі сереалізувати значення з комою, що виходить за допустимі межі, буде викликати ***ValueError*** в строгому співвідношенні з специфікацією JSON, крім того, щоб використовувати еквіваленти із JavaScript.

Якщо indent є додатнім числом, тоді масиви и об’єкти в JSON будуть виводитись з цим рівнем відступа. Якщо рівень відступа = 0 або від’ємний, то замість цього будуть просто використовуватись нові стрічки. Значення по замовчуванню **None** відображає найбільш компактні представлення.

Якщо separators є кортежем, то він буде використовуватись замість значення по замовчуванню (', ', ': '). (',', ':') дає найбільш компактне представлення JSON.

Encoding – кодування для екземплярів str , по замовчуванню – UTF-8 .

default(obj) - функція, яка має повернути сереалізовану версію obj або викликати **TypeError.**

Якщо sort\_keys = True тоді ключі виводимого словника будуть відсортовані.

Для того, щоб використовувати власний підклас ***JSONEncoder***, задайте його за допомогою аргумента cls, в противному випадку буде використовуватись ***JSONEncoder.***

json.**load**(fp[, encoding[, cls[, object\_hook[, parse\_float[, parse\_int[, parse\_constant[, object\_pairs\_hook[, \*\*kw]]]]]]]])

Десереалізує fp в об’єкт Pethon.

Якщо вміщающе fp закодовано кодуванням на основі ASCII, а не UTF-8, тоді повинно бути вказане відповідне кодування . Кодування, не основані на ASCII використовувати неможна і в такому випадку потрібно використовувати обгортку ||codecs.getreader(encoding)(fp)|| або просто декодувати в об’єкт ***Unicode*** та передати в **loads().**

object\_hook – опціональна функція, яка використовується до результату декодування об’єкта(**dict**). Використовуватись буде значення, повернене цією функцією, а не получений словник. Ця можливість корисна для реалізації власних декоддувальників.

parse\_float , якщо визначений, буде викликаний для кожного значення JSON з плаваючою крапкою для його декодування. По замовчуванню, це еквівалентно float(num\_str). Можна використовувати інший тип даних або парсер для цього значення.