**МОЖЛИВОСТІ ПАКЕТУ JSON**

**JSON (JavaScript Object Notation)** - простий формат обміну даними, що базується на підмножині синтаксису JavaScript. Модуль json дозволяє кодувати і декодувати дані в зручному форматі.

Приклад:

**>>> import** **json**

**>>>** json.dumps(['foo', {'bar': ('baz', **None**, 1.0, 2)}])

'["foo", {"bar": ["baz", null, 1.0, 2]}]'

**>>>** print(json.dumps("**\"**foo**\b**ar"))

"\"foo\bar"

**>>> from** **io** **import** StringIO

**>>>** io = StringIO()

**>>>** json.dump(['streaming API'], io)

**>>>** io.getvalue()

'["streaming API"]'

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'

Основи:

**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*)

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

Якщо **ensure\_ascii = True**, все не-ASCII символи у висновку будуть виведені на екрані послідовностями **\uXXXX**, і результатом буде рядок, що містить тільки ASCII символи. Якщо **ensure\_ascii = False**, рядки запишуться як є.

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

Якщо **indent** є невід'ємним числом, то масиви і об'єкти в JSON будуть виводитися з цим рівнем відступу. Якщо рівень відступу 0, негативний або "", то замість цього просто використовуватимуться нові рядки. Значення за замовчуванням None відображає найбільш компактне уявлення. Якщо indent - рядок, то вона і буде використовуватися в якості відступу.

Якщо sort\_keys = True, то ключі виведеного словника будуть відсортовані.

**json.loads** (*s, encoding = None, cls = None, object\_hook = None, parse\_float = None, parse\_int = None, parse\_constant = None, object\_pairs\_hook = None, \*\* kw*)

**object\_hook** - функція, яка застосовується до результату декодування об'єкта (dict). Використовуватися буде значення, повернене цією функцією, а не отриманий словник.

**object\_pairs\_hook** - функція, яка застосовується до результату декодування об'єкта з певною послідовністю пар *ключ/значення*. Буде використаний результат, що повертається функцією, замість отриманого словника. Якщо заданий так само **object\_hook**, то пріоритет віддається **object\_pairs\_hook**.

**parse\_float**, якщо визначений, буде викликаний для кожного значення JSON з плаваючою точкою. За замовчуванням, це еквівалентно float (num\_str).

**parse\_int**, якщо визначений, буде викликаний для рядка JSON з числовим значенням. За замовчуванням еквівалентно int (num\_str).

**parse\_constant**, якщо визначений, буде викликаний для наступних рядків: "-Infinity", "Infinity", "NaN". Може бути використано для порушення винятків при виявленні помилкових чисел JSON.

Клас **json.JSONDecoder** (*object\_hook = None, parse\_float = None, parse\_int = None, parse\_constant = None, strict = True, object\_pairs\_hook = None*) - простий декодер JSON.

Класс **json.JSONEncoder**(*skipkeys=False, ensure\_ascii=True, check\_circular=True, allow\_nan=True, sort\_keys=False, indent=None, separators=None, default=None*) – декодує в зворотню сторону

| **JSON** | **Python** |
| --- | --- |
| object | dict |
| array | list |
| string | str |
| number (int) | int |
| number (real) | float |
| true | True |
| false | False |
| null | None |