

JSON, CSV

JSON (JavaScript Object Notation) — это иерархический формат файлов, который легко анализировать и достаточно легко читать и писать вручную. Он берет своё начало в Интернете и означает нотацию объектов JavaScript. Одна из причин, по которой стоит обратить внимание на JSON, заключается в том, что многие Web-API используют JSON в качестве формата передачи данных.

JSON — это формат, определённый в Юникоде, а не в байтах. При сериализации он принимает структуру данных и преобразует ее в строку Юникода, а при десериализации он принимает строку Юникода и возвращает структуру данных. Однако недавно в стандарт были внесены поправки, указывающие предпочтительную кодировку: utf-8. Благодаря этому дополнению формат также определяется как поток байтов. В некоторых случаях кодировка по-прежнему отделена от формата. В частности, при отправке или получении JSON через HTTP кодировка является истиной в последней инстанции. Однако даже в этом случае, когда кодировка явно не указана, следует использовать UTF-8.

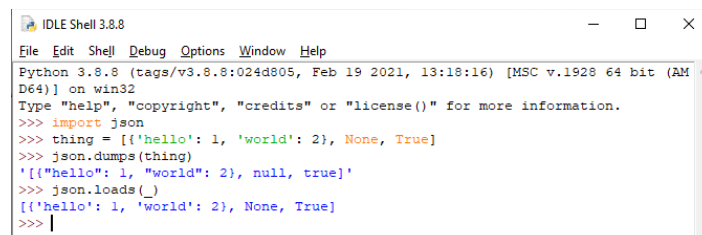
JSON — это простой формат сериализации, поддерживающий лишь несколько типов:

- строки;
- числа;
- логические значения;
- тип null;
- массивы значений JSON;
- объекты (словари, сопоставляющие строки со значениями JSON).

JSON не полностью определяет числовые диапазоны или точность. Если требуются точные целые числа, обычно можно предположить, что диапазон от -2^{53} до 2^{53} представляется достаточно точно.

Хотя библиотека Python JSON может читать/записывать файлы напрямую, следует всегда разделять задачи: считывать столько данных, сколько нужно, и передавать строку непосредственно в JSON.

Наиболее важными функциями модуля json являются loads и dumps. Символ s в конце означает строку, которую эти функции принимают и возвращают (рисунок 1).



```
IDLE Shell 3.8.8
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:18:16) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import json
>>> thing = {'hello': 1, 'world': 2}, None, True]
>>> json.dumps(thing)
'{"hello": 1, "world": 2}, null, true]'
>>> json.loads('{"hello": 1, "world": 2}, null, true]')
[{'hello': 1, 'world': 2}, None, True]
>>>
```

Рисунок 1 - Использование dumps() и loads()

Объект None в Python сопоставляется с null объектом JSON, логические значения в Python сопоставляются с логическими значениями в JSON, а числа и строки сопоставляются с числами и строками. Обратите внимание, что библиотеки синтаксического анализа JSON Python принимают специальные решения о том, должно ли число сопоставляться с целым числом или с плавающей запятой, на основе его обозначения (рисунок 2).

```
>>> json.loads('1')
1
>>> json.loads('1.0')
1.0
>>>
```

Рисунок 2 - Сопоставление чисел

Но не всегда `loads()` библиотеки JSON принимают одно и то же решение, и в некоторых случаях это может привести к проблемам совместимости. В целях отладки часто полезно иметь возможность печатать JSON. Функция `dumps` может сделать это с помощью дополнительных аргументов. Ниже приводится обычный набор аргументов в пользу красивой печати (рисунок 3).

```
>>> encoded_string='{"b":1,"a":2}'
>>> print(json.dumps(json.loads(encoded_string), indent=4))
{
    "b": 1,
    "a": 2
}
>>> |
```

Рисунок 3 - Вывод JSON в читабельном формате

Поскольку JSON обычно используется для передачи данных при работе с Web-API рассмотрим следующий пример - запросим свой IP адрес. Эту услугу предоставляет сервис `httpbin.org`, возвращающий данные в формате JSON.

```
import json
import urllib.request
with urllib.request.urlopen('https://httpbin.org/ip') as f:
    print(json.loads(f.read()))
```

В этом примере сначала подключаем два модуля, `json` - для работы с форматом JSON и `urllib.request` - для работы с URL-адресами (базовая и дайджест-аутентификация, перенаправление, файлы cookie и многое другое) и необходимым методом `urlopen()`. Далее выводим полученные данные в формате JSON (рисунок 4).

```
{'origin': '78.37.98.161'}
>>> |
```

Рисуно 4 - Результат работы

Как видно из рисунка, формат вывода результата ключ:значение. При работе с JSON можно указать значение какого ключа собираемся получить. Модифицируем программу следующим образом

```
import json
import urllib.request
with urllib.request.urlopen('https://httpbin.org/ip') as f:
    print(f"my ip address = {json.loads(f.read())['origin']}")
```

Результат работы программы представлен на рисунке 5

```
my ip address = 78.37.98.161
>>> |
```

Рисунок 5 - Результат работы программы

Формат CSV (Comma-Separated Values) имеет несколько преимуществ

- ограниченность;
- представляет скалярные типы в двумерном массиве.

Кроме того, этот формат изначально импортируется в приложения для работы с электронными таблицами, такие как Microsoft Excel или Google Sheets, что пригодится при подготовке отчётов. Примерами таких отчётов являются детализация расходов на оплату сторонних услуг для финансового отдела или отчёт об управляемых инцидентах и времени на восстановление для руководства. Во всех этих случаях наличие формата, который легко

создавать и импортировать в приложения для работы с электронными таблицами, позволяет легко автоматизировать задачу.

По соглашению первая строка должна быть строкой заголовка. Хотя API Python не требует этого, настоятельно рекомендуется следовать этому соглашению. CSV может представлять только строки и числа, поэтому вместо того, чтобы полагаться на плохо документированные стандарты записи логических значений, необходимо делать это явно.

Существует два основных подхода для чтения файлов CSV. Использование `csv.reader` возвращает итератор, который выдаёт строку в виде списка. Однако, предполагая, что соблюдается соглашение о том, что первая строка является именами полей. И `csv.DictReader`, который ничего не требует для первой строки и словарь для каждой последующей строки, используя имена полей в качестве ключей. Это обеспечивает более надёжный анализ, когда конечные пользователи добавляют поля или изменяют их порядок.

Используем в качестве примера публичный доступ к CSV файлам (<https://www.stats.govt.nz/large-datasets/csv-files-for-download/>). Сначала используем `csv.reader` для работы с файлом, предварительно скачав его.

```
import csv
with open('d:\\temp\\research-and-survey-2022-csv-notes.csv', 'r') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row[1])
```

В этой программе сначала подключаем модуль `csv`, далее открываем файл и передаём его в `csv.reader()`. Затем проходим по списку и выводим второго поля. Результат работы программы показан на рисунке 6

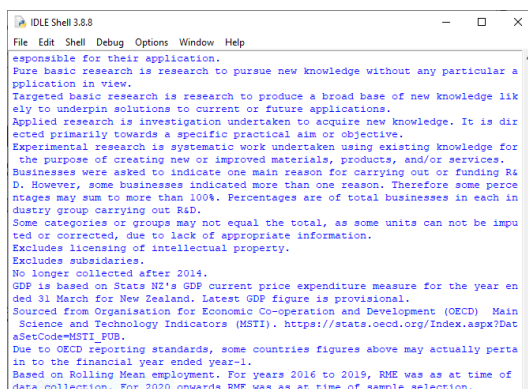


Рисунок 6 - Вывод второго поля

При использовании `csv.DictReader` можно обратиться к полю по его названию

```
import csv
with open('d:\\temp\\research-and-survey-2022-csv-notes.csv', 'r') as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(row['Footnote'])
```

Аналогичным образом с помощью модуля `csv` можно и записать файл в формате CSV

```
import csv
data = [['hostname', 'vendor', 'model', 'location'],
        ['sw1', 'Cisco', '3750', 'London, Best str'],
        ['sw2', 'Cisco', '3850', 'Liverpool, Better str'],
        ['sw3', 'Cisco', '3650', 'Liverpool, Better str'],
```

```

['sw4', 'Cisco', '3650', 'London, Best str']]
with open('d:\\temp\\sw_data_new.csv', 'w') as f:
    writer = csv.writer(f, quoting=csv.QUOTE_NONNUMERIC)
    for row in data:
        writer.writerow(row)
with open('d:\\temp\\sw_data_new.csv') as f:
    print(f.read())

```

Результат работы программы показан на рисунке 7

```

"hostname","vendor","model","location"
"sw1","Cisco","3750","London, Best str"
"sw2","Cisco","3850","Liverpool, Better str"
"sw3","Cisco","3650","Liverpool, Better str"
"sw4","Cisco","3650","London, Best str"

```

Рисунок 7 - Вывод сохранённого файла csv

С помощью DictWriter можно записать словари в формат CSV.

В целом DictWriter работает так же, как writer, но так как словари не упорядочены, надо указывать явно в каком порядке будут идти столбцы в файле. Для этого используется параметр fieldnames.

```

import csv
data = [{
    'hostname': 'sw1',
    'location': 'London',
    'model': '3750',
    'vendor': 'Cisco'
}, {
    'hostname': 'sw2',
    'location': 'Liverpool',
    'model': '3850',
    'vendor': 'Cisco'
}, {
    'hostname': 'sw3',
    'location': 'Liverpool',
    'model': '3650',
    'vendor': 'Cisco'
}, {
    'hostname': 'sw4',
    'location': 'London',
    'model': '3650',
    'vendor': 'Cisco'
}]
with open('d:\\temp\\csv_write_dictwriter.csv', 'w') as f:
    writer = csv.DictWriter(f, fieldnames=list(data[0].keys()),
        quoting=csv.QUOTE_NONNUMERIC)
    writer.writeheader()
    for d in data:
        writer.writerow(d)

```

Для сохранения этих данных в JSON будет использоваться метод dumps()

```

with open('d:\\temp\\sw_templates.json', 'w') as f:
    f.write(json.dumps(data))

```

Или dump()

```
with open('d:\\temp\\sw_templates.json', 'w') as f:  
    json.dump(data, f)
```

Задания:

- выведите данные имя, город, номер телефона в читабельном формате из сервиса <https://jsonplaceholder.typicode.com/users>;
- выведите в читабельном формате данные, указанные преподавателем, с сайта <https://www.stats.govt.nz/large-datasets/csv-files-for-download/> используя оба способа чтения CSV;
- просуммируйте в каждой строке числовые значения в CSV файле и выведите общий итог;
- запишите полученные данные из CSV в JSON и наоборот.