Формат JSON

JSON, самый популярный формат файлов в Интернете. Некоторые из вас могут удивиться этому утверждению, так как услышат про этот формат впервые. На самом деле все из вас пользовались данным форматом, пусть и неявно. Именно JSON обмениваются большинство приложений в Интернете.

JSON (англ. JavaScript Object Notation) — один из самых популярных типов структурированных файлов, поддерживающих произвольную вложенность. Это формат объекта в языке JavaScript, содержащий в себе сочетание словарей и списков (в терминах Python). Оказывается, что вместо того, чтобы передавать или хранить файлы в каких-то форматах, потом при помощи библиотек обрабатывая их, удобней передавать сами объекты языка. Ведь в этом случае не надо ничего дополнительно обрабатывать. Перед нами — уже готовый объект. Так как в последнее время очень распространились веб-приложения на JavaScript, JSON стал одним из самых популярных форматов, в том числе и в других языках.

Чтобы лучше понять JSON, давайте сначала посмотрим на модуль pickle. Он служит для того, чтобы превращать любой объект Python в байтовую структуру и обратно:

```
from pickle import loads, dumps
s = {'Иван': 24, 'Сергей': 11}
d = dumps(s)
print(d)
b'\x80\x03}q\x00(X\x08\x00\x00\x00\xd0\x98\xd0\xb2\xd0\xb0\xd0\xbdq\x01K\x18X
\x0c\x00\x00\x00\xd0\xa1\xd0\xb5\xd1\x80\xd0\xb3\xd0\xb5\xd0\xb9q\x02K\x0bu.'
loads(d)
{'Иван': 24, 'Сергей': 11}
```

Объект, представленный в виде массива байт, легко хранить на жестком диске, в базах данных, передавать по сети между двумя приложениями или от одного приложения другому в рамках одного компьютера и т. д.

Идея в том, что любой объект можно сбросить на диск (или по сети) и восстановить в другом месте другим интерпретатором. Это очень здорово. И очень сложно для компилируемых языков. Постоянно надо проводить аналогии между pickle и JSON. Что это одна и та же технология, только по-разному в разных языках реализованная.

Теперь посмотрим, что же такое JSON.

Запросим по ссылке у Яндекс.Карт информацию о московском аэропорте Внуково.

Пройдите по этой ссылке и убедитесь, что ответ действительно приходит.

Нам вернется вот такой ответ:

```
{
        "response": {
                "GeoObjectCollection": {
                        "metaDataProperty": {
                                "GeocoderResponseMetaData": {
                                        "request": "аэропорт Внуково",
                                        "found": "2",
                                        "results": "10"
                                }
```

```
                        },
                        "featureMember": [{
                                "GeoObject": {
                                        "metaDataProperty": {
                                                "GeocoderMetaData": {
                                                        "kind": "airport",
                                                        "text": "Россия, Москва,
аэропорт Внуково",
                                                        "precision": "other",
                                                        "Address": {
                                                                "country_code":
"RU",
                                                                "formatted":
"Москва, аэропорт Внуково",
                                                                "Components": [{
                                                                        "kind":
"country",
                                                                        "name":
"Россия"
                                                                },
                                                                {
                                                                        "kind":
"province",
                                                                        "name":
"Центральный федеральный округ"
                                                                },
                                                                {
                                                                        "kind":
"province",
                                                                        "name":
"Москва"
                                                                },
                                                                {
                                                                        "kind":
"airport",
                                                                        "name":
"аэропорт Внуково"
                                                                }]
                                                        },
                                                        "AddressDetails": {
                                                                "Country": {
        "AddressLine": "Москва, аэропорт Внуково",
        "CountryNameCode": "RU",
        "CountryName": "Россия",
        "AdministrativeArea": {
        "AdministrativeAreaName": "Москва",
        "Locality": {
                                                                                                                "
                                                                                                                }
                                                                                                        }
                                                                                                }
                                                                                        }
                                                                                }
                                                                        }
                                                                }
                                                        },
                                                        "description": "Москва, Россия",
                                                        "name": "аэропорт Внуково",
```

```
                                        "boundedBy": {
                                                "Envelope": {
                                                        "lowerCorner":
"37.229833 55.583169",
                                                        "upperCorner":
"37.303809 55.61598"
                                                }
                                        },
                                        "Point": {
                                                "pos": "37.286292 55.605058"
                                        }
                                }
                        },
                        {
                                "GeoObject": {
                                        "metaDataProperty": {
                                                "GeocoderMetaData": {
                                                        "kind": "railway",
                                                        "text": "Россия,
Киевское направление Московской железной дороги, платформа Аэропорт Внуково",
                                                        "precision": "other",
                                                        "Address": {
                                                                "country_code":
"RU",
                                                                "formatted":
"Киевское направление Московской железной дороги, платформа Аэропорт
Внуково",
                                                                "Components": [{
                                                                        "kind":
"country",
                                                                        "name":
"Россия"
                                                                },
                                                                {
                                                                        "kind":
"province",
                                                                        "name":
"Центральный федеральный округ"
                                                                },
                                                                {
                                                                        "kind":
"route",
                                                                        "name":
"Киевское направление Московской железной дороги"
                                                                },
                                                                {
                                                                        "kind":
"railway",
                                                                        "name":
"платформа Аэропорт Внуково"
                                                                }]
                                                        },
                                                        "AddressDetails": {
                                                                "Country": {

        "AddressLine": "Киевское направление Московской железной дороги,
платформа Аэропорт Внуково",

        "CountryNameCode": "RU",

        "CountryName": "Россия",

        "Country": {
```

```
        "Locality": ""
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    },
                                                    "description": "Киевское направление
Московской железной дороги, Россия",
                                                    "name": "платформа Аэропорт Внуково",
                                                    "boundedBy": {
                                                            "Envelope": {
                                                                    "lowerCorner":
"37.279914 55.601106",
                                                                    "upperCorner":
"37.296371 55.610423"
                                                            }
                                                    },
                                                    "Point": {
                                                            "pos": "37.288142 55.605765"
                                                    }
                                            }
                                    }]
                            }
                    }
}
```

Как видно из этого примера, JSON очень похож по синтаксису на формат словаря в Python. Поэтому любая работа с JSON в Python происходит по алгоритму:

1. Получение словаря с данными
2. Преобразование словаря в JSON-объект
3. Передача данных

Или в обратную сторону:

1. Получение файла или строки с JSON-содержимым
2. Преобразование данных в словарь Python
3. Работа с данными

1. Библиотека для работы с JSON

Для работы с JSON есть стандартный модуль json.

Этот модуль содержит аналогичный *pickle* интерфейс, то есть нам доступны функции `loads()` (загрузка из строки JSON-объекта и преобразование его в Python-объект) и `dumps()` (обратное преобразование).

Чтение json.

Откроем в IDE файл `cats_json.json`. Мы видим, что там в виде словаря записана некоторая информация о питомце:

```
{
  "name": "Barsik",
  "age": 7,
  "meals": [
```

```
    "Wiskas",
    "Royal Canin",
    "Purina",
    "Hills",
    "Brit Care"
  ]
}
```

Обратите внимание: в формате JSON используются только двойные кавычки.

Для чтения данных в модуле *json* есть два метода:

1. `json.load()` — считывает целиком файл в формате JSON и возвращает объекты Python
2. `json.loads()` — считывает строку в формате JSON и возвращает объекты Python

Напишем код, читающий файл и выводящий содержимое полученного словаря:

```
import json

with open('cats_json.json') as cat_file:
    data = json.load(cat_file)
for key, value in data.items():
    if type(value) == list:
        print(f'{key}: {", ".join(value)}')
    else:
        print(f'{key}: {value}')
name: Barsik
age: 7
meals: Wiskas, Royal Canin, Purina, Hills, Brit Care
```

В отличие от метода `json.load(filename)`, метод `json.loads(string)` считывает строку и возвращает объект json, если его возможно получить из переданной строки.

В случае, если строка или файловый объект содержат некорректный JSON, будет выброшено исключение типа json.decoder.JSONDecodeError.

```
import json

s = 'Не Json'
json.loads(s)
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

Создайте в IDE новый файл cats_2.json, в который кроме Барсика добавьте еще одного кота по вашему усмотрению. Поскольку теперь в файле у нас уже два словаря, их надо объединить в одну структуру, пусть это будет список. Получим что-то вроде такого содержимого:

```
[
  {
    "name": "Barsik",
    "age": 7,
    "meals": [
      "Wiskas",
      "Royal Canin",
      "Purina",
      "Hills",
      "Brit Care"
```

```
    ]
  },
  {
    "name": "Mursik",
    "age": 3,
    "meals": [
      "Purina",
      "Hills",
      "Brit Care"
    ]
  }
]
```

Чтобы воспользоваться методом `loads()`, нужно сначала считать весь файл в строку, а затем передать ее методу для преобразования в json-объект.

```
import json

with open('cats_2.json') as cat_file:
    f = cat_file.read()
    data = json.loads(f)
    for item in data:
        for key, value in item.items():
            if type(value) == list:
                print(f'{key}: {", ".join(value)}')
            else:
                print(f'{key}: {value}')
        print()
```

И в том, и в другом случае мы получили объект языка Python, словарь или список, с которым можно сразу работать средствами языка.

В таблице представлено соответствие между данными в Python и в JSON:

Python	JSON
dict	object
list, tuple	array
str	string
int, float	number
True	true
False	false
None	null

При обратном преобразовании массив *array* преобразуется в список.

Запись в файл.

Для записи информации в файл в *json* также есть два метода:

1. `json.dump()` — метод записывает объект Python в файл в формате JSON
2. `json.dumps()` — метод преобразует объект Python в строку в формате JSON

Давайте создадим словарь, в котором коту добавим хозяев, а затем полученную информацию сохраним в файле:

```
import json

cats_dict = {
    'name': 'Pushin',
    'age': 1,
    'meals': [
        'Purina', 'Cat Chow', 'Hills'
    ],
    'owners': [
        {
            'first_name': 'Bill',
            'last_name': 'Gates'
        },
        {
            'first_name': 'Melinda',
            'last_name': 'Gates'
        }
    ]
}

with open('cats_3.json', 'w') as cat_file:
    json.dump(cats_dict, cat_file)
```

Если сейчас открыть в IDE PyCharm файл cats_3.json, который был создан нашей программой, мы увидим, что json выведен в одну строку без форматирования. Для представления json в удобном для чтения виде в PyCharm можно использовать горячую клавишу Ctrl-Alt-L.

Метод `dumps()` используется, когда надо просто преобразовать объект в json-формат, необязательно для записи в файл. Это нужно, например, при передаче информации в веб-приложении.

Посмотрим на примере:

```
import json

weekdays = {i: day
            for i, day in enumerate(['Sunday',
                                     'Monday',
                                     'Tuesday',
                                     'Wednesday',
                                     'Thursday',
                                     'Friday',
                                     'Saturday'
                                     ])}
data = json.dumps(weekdays)
print(data)
print(type(data))
```

Получим строку в формате json:

```
{"0": "Sunday", "1": "Monday", "2": "Tuesday", "3": "Wednesday", "4":
"Thursday", "5": "Friday", "6": "Saturday"}
<class 'str'>
```

Дополнительные параметры методов записи.

Методы записи имеют несколько необязательных параметров, которые можно менять для более удобного чтения человеком. Рассмотрим некоторые из них.

ensure_ascii. В случае, если `ensure_ascii=True` (по умолчанию), все не-ASCII символы в выводе будут экранированы последовательностями \uXXXX. Если `ensure_ascii=False`, эти символы будут записаны как есть. Это важно, если в содержимом есть русские буквы.

Запишем в файл cats.json содержимое списка без флага:

```
with open('cats.json', 'w') as file:
    json.dump(data, file)
```

В файле:

```
[
    {
        "name": "\u0411\u0430\u0440\u0441\u0438\u043a",
        "age": 7,
        "toys": [
            "\u041c\u044b\u0448\u043a\u0430",
            "\u041f\u0440\u0443\u0442\u0438\u043a",
            "\u0411\u0430\u043d\u0442\u0438\u043a",
            "\u0421\u0432\u043e\u0439 \u0445\u0432\u043e\u0441\u0442"
        ]
    },
    {
        "name": "\u041c\u0443\u0440\u0437\u0438\u043a",
        "age": 3,
        "toys": [
            "\u0420\u0443\u043a\u0430
\u0445\u043e\u0437\u044f\u0439\u043a\u0438",
            "\u0428\u043d\u0443\u0440 \u043e\u0442
\u0442\u0435\u043b\u0435\u0432\u0438\u0437\u043e\u0440\u0430",
            "\u041e\u0431\u043e\u0438 \u043d\u0430
\u0441\u0442\u0435\u043d\u0435"
        ]
    }
]
```

А теперь — с флагом:

```
with open('cats.json', 'w') as file:
    json.dump(data, file, ensure_ascii=False)
```

В файле:

```
[
    {
        "name": "Барсик",
        "age": 7,
        "toys": [
            "Мышка",
            "Прутик",
            "Бантик",
            "Свой хвост"
        ]
    },
    {
        "name": "Мурзик",
        "age": 3,
        "toys": [
            "Рука хозяйки",
            "Шнур от телевизора",
```

```
            "Обои на стене"
        ]
    }
]
```

indent. Отступ *indent* нужен для удобного для чтения человеком представления информации в объекте JSON. По умолчанию имеет значение None для более компактного представления, то есть без отступов. Также отступов не будет, если значение *indent* равно 0, отрицательному числу или пустой строке. Если *indent* — строка (например, `\t`), эта строка используется в качестве отступа.

Пример со значением `indent=2`:

```
with open('cats.json', 'w') as file:
    json.dump(data, file, ensure_ascii=False, indent=2)
```

Результат:

```
[
  {
    "name": "Барсик",
    "age": 7,
    "toys": [
      "Мышка",
      "Прутик",
      "Бантик",
      "Свой хвост"
    ]
  },
  {
    "name": "Мурзик",
    "age": 3,
    "toys": [
      "Рука хозяйки",
      "Шнур от телевизора",
      "Обои на стене"
    ]
  }
]
```

sort_keys. Если `sort_keys=True` (по умолчанию False), ключи выводимого словаря будут отсортированы. Это удобно, если ключей много.

Пример со значением `sort_keys=True`:

```
with open('cats.json', 'w') as file:
    json.dump(data, file, ensure_ascii=False,
              indent=2, sort_keys=True)
```

Результат:

```
[
  {
    "age": 7,
    "name": "Барсик",
    "toys": [
      "Мышка",
      "Прутик",
      "Бантик",
```

```
      "Свой хвост"
    ]
  },
  {
    "age": 3,
    "name": "Мурзик",
    "toys": [
      "Рука хозяйки",
      "Шнур от телевизора",
      "Обои на стене"
    ]
  }
]
```

В заключении

Сериализация и десериализация

Мы производили преобразования между объектами языка Python и json-объектами. Такие преобразования называются **сериализацией** (кодирование в json-формат, то есть в поток байт) и **десериализацией** (декодирование в объект языка).

Есть еще несколько моментов при работе с JSON, о которых стоит помнить:

1. Чтобы не возникали проблемы с кодировкой, если в файл передаются данные с русскими буквами, как и во всех других случаях работы с файлом, при открытии нужно принудительно устанавливать кодировку (особенно актуально для OS семейства Windows):
```
with open('cats_3.json', 'w', encoding='utf8') as cat_file:
    cat_file.write(json.dumps(cats_dict))
```
4. При создании json-файла «вручную» нужно помнить, что в нем нельзя использовать одинарные кавычки. При создании программными средствами нужные кавычки ставятся автоматически.
5. Ключами словаря в json не могут быть кортежи и числа. Но ключ-число не вызовет ошибку при сериализации, он будет просто преобразован в строку.
6. Помните, что при преобразовании данные будут не всегда того же типа, что были в Python, то есть:

```
print(json.loads(json.dumps(weekdays)) == weekdays)  # False
```

Более подробную информацию по работе с модулем *json* можно найти по ссылке.