Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа с JSON форматом»

ОТЧЕТ
по лабораторной работе №19(1)
дисциплины
«Основы программной инженерии»

Выполнил:
Мизин Глеб Егорович
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

```python
from jsonschema import validate
import json
import sys

schema = {
    "type": "object",
    "title": "MySchema",
    "required": [
        "s_b_a",
        "b_a",
        "t_a"
    ],
    "properties": {
        "s_b_a": {"type": "number"},
        "b_a": {"type": "number"},
        "t_a": {"type": "number"}
    },
}


def validating(check_data):
    try:
        validate(instance=check_data, schema=schema)
    except jsonschema.exceptions.ValidationError:
        err = "JSON data is not correct"
        return False, err

    message = "JSON data is correct"
    return True, message
```

Рисунок 1 – Валидация по средствам библиотеки jsonschema

```python
import marshmallow
from marshmallow import Schema, fields
import json
import sys


class Bank(Schema):
    s_b_a = fields.Int(required=True)
    b_a = fields.Int(required=True)
    t_a = fields.Int(required=True)


BankSchema = Bank()


def validating(check_data):
    try:
        BankSchema.load(check_data)
    except marshmallow.exceptions.ValidationError as err:
        return False, err

    message = "JSON data is correct"
    return True, message
```

Рисунок 2 – Валидация по средствам библиотеки marshmallow

```python
import json
from pydantic import BaseModel, Field, root_validator
import sys


class Bank(BaseModel):

    s_b_a: str = Field()
    b_a: str = Field()
    t_a: str

    @root_validator()
    def validation(cls, values):
        s_b_a = values.get("s_b_a")
        b_a = values.get("s_b_a")
        t_a = values.get("s_b_a")

        if s_b_a.isdigit() and b_a.isdigit() and t_a.isdigit():
            return values
        else:
            raise ValueError("Счета должны состоять только из цифр!")
```

Рисунок 3 – Валидация по средствам библиотеки pydantic

Контрольные вопросы

1. Для чего используется JSON?

JSON (англ. JavaScript Object Notation, обычно произносится как /ˈdʒeɪsən/ JAY-sən) - текстовый формат обмена данными, основанный на JavaScript.

За счёт своей лаконичности по сравнению с XML формат JSON может быть более подходящим для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения). Легкочитаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента. Это информативное руководство поможет вам быстрее разобраться с данными, которые вы можете использовать с JSON и основной структурой с синтаксисом этого же формата.

2. Какие типы значений используются в JSON?

Объект JSON это формат данных — ключ-значение, который обычно рендерится в фигурных скобках. Когда вы работаете с JSON, то вы скорее всего видите JSON объекты в .json файле, но они также могут быть и как JSON объект или строка уже в контексте самой программы.

Вот так выглядит JSON объект:

```
{
    "first_name" : "Sammy",
    "last_name" : "Shark",
    "location" : "Ocean",
    "online" : true,
    "followers" : 987
}
```

Ключи в JSON находятся с левой стороны от двоеточия. Их нужно оборачивать в скобки, как с "key" и это может быть любая строка. В каждом объекте, ключи должны быть уникальными. Такие ключевые строки могут

содержать пробелы, как в "first_name" , но такой подход может усложнить получение доступа к ним во время процесса разработки, так что лучшим вариантом в таких случаях будет использование нижнего подчеркивания, как сделано тут "first_name" . JSON значения находятся с правой стороны от двоеточия. Если быть точным, то им нужно быть одним из шести типов данных: строкой, числом, объектом, массивом, булевым значением или null .

В качестве значений в JSON могут быть использованы:

запись — это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{ }». Ключ описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми.

массив (одномерный) — это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми. Массив может быть пустым, т.е. не содержать ни одного значения. Значения в пределах одного массива могут иметь разный тип.

число (целое или вещественное).

литералы true (логическое значение «истина»), false (логическое значение «ложь») и null.

строка — это упорядоченное множество из нуля или более символов юникода, заключённое в двойные кавычки. Символы могут быть указаны с использованием escape последовательностей, начинающихся с обратной косой черты «\» (поддерживаются варианты ', ", \, \/, \t, \n, \r, \f и \b), или записаны шестнадцатеричным кодом в кодировке Unicode в виде \uFFFF.

3. Как организована работа со сложными данными в JSON?

JSON может содержать другие вложенные объекты в JSON, в дополнение к вложенным массивам. Такие объекты и массивы будут передаваться, как значения назначенные ключам и будут представлять собой связку ключ-значение.

Данные также могут быть вложены в формате JSON, используя JavaScript массивы, которые передаются как значения. JavaScript использует квадратные скобки [ ] для формирования массива. Массивы по своей сути — это упорядоченные коллекции и могут включать в себя значения совершенно разных типов данных.

4. Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON?

Формат обмена данными JSON5 — это расширенная JSON-версия, которая призвана смягчить некоторые ограничения JSON, расширив его синтаксис и включив в него некоторые функции из ECMAScript 5.1.

Объекты

Ключи объектов могут быть именами идентификаторов ECMAScript 5.1.

Объекты могут иметь одну запятую.

Массивы

Массивы могут иметь одну запятую.

Строки

Строки могут заключаться в одинарные кавычки.

Строки могут охватывать несколько строк, экранируя символы новой строки.

Строки могут включать в себя экранирование символов.

Числа

Числа могут быть шестнадцатеричными.

Числа могут иметь ведущую или последующую десятичную точку.

Числа могут быть Infinity, -Infinity2 и NaN.

Числа могут начинаться с явно определенного знака +.

Комментарии

Допускаются однострочные и многострочные комментарии.

Пробельные символы

Разрешены дополнительные пробельные символы.

5. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5?

PyJSON5 — документация PyJSON5 1.6.2

6. Какие средства предоставляет язык Python для сериализации данных в формате JSON?

1. Сериализация данных в формат JSON:

```
json.dump()  # конвертировать python объект в json и записать в файл
json.dumps() # тоже самое, но в строку
```

7. В чем отличие функций json.dump() и json.dumps()?

json.dump() # конвертировать python объект в json и записать в файл json.dumps() # тоже самое, но в строку

8. Какие средства предоставляет язык Python для десериализации данных из формата JSON?

2. Десериализация данных из формата JSON:

```
json.load()  # прочитать json из файла и конвертировать в python объект
json.loads() # тоже самое, но из строки с json  (s на конце от string/строка)
```

9. Какие средства необходимо использовать для работы с данными формата JSON, содержащими кирилицу?

Выполнить сериализацию данных в формат JSON.

Для поддержки кирилицы установим ensure_ascii=False

json.dump(staff, fout, ensure_ascii=False, indent=4)

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных? Приведите схему данных для примера 1.

Схема JSON — это декларативный язык, позволяющий аннотировать и проверять документы JSON.

jsonschema · PyPI

Понимание схемы JSON (JSON Schema), часть 1 (infostart.ru)