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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2
дисциплины «Анализ данных»

Выполнил:
Говоров Егор Юрьевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизирование систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук, доцент, доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2024 г.

Тема: Работа с данными формата JSON в языке Python

Цель: приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

Ход работы:

1. Создал файл (example.py), в котором проработал пример лабораторной работы на рис. 1 расположен ввод-вывод, а на рис. 2 содержимое созданного json файла:

```
D:\Users\Admin\anaconda3\envs\Lab_2\python.exe "D:\Users\Admin\Рабочий стол\Data_Analize\Data_Analize_Govorov-2\example\example.py"
>>> add
Фамилия и инициалы? Говоров.Е.Ю
Должность? Управляющий
Год поступления? 2004
>>> add
Фамилия и инициалы? Данилецкий.Д.В
Должность? Управлящий, но не такой крутой
Год поступления? 2008
>>> list
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Говоров.Е.Ю                    | Управляющий          |     2004 |
|    2 | Данилецкий.Д.В                 | Управлящий, но не такой крутой |     2008 |
+------+--------------------------------+----------------------+----------+
>>> save egor.json
>>>
```

Рисунок 1 – Результат работы example.py

```
[
    {
        "name": "Говоров.Е.Ю",
        "post": "Управляющий",
        "year": 2004
    },
    {
        "name": "Данилецкий.Д.В",
        "post": "Управлящий, но не такой крутой",
        "year": 2008
    }
]
```

Рисунок 2 – Json файл

2. Индивидуальное задание: Для своего варианта лабораторной работы 2.8 необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON.

Создал файл (individual.py), в котором выполнил индивидуальное задание в соответствии с лабораторной 2.8, на рис. 3 указан результат выполнениякоманд, а на рис. 4 содержимое созданного json файла.

```
Введите номер действия: 1
Введите название пункта назначения: Питер-Москва
Введите номер рейса: 12
Введите тип самолета: Пассажирский
Выберите действие:
1. Добавить рейс
2. Вывести список рейсов
3. Поиск рейсов по типу самолета
4. Сохранить и выйти
Введите номер действия: 1
Введите название пункта назначения: Ростов-Ставрополь
Введите номер рейса: 23
Введите тип самолета: Грузовой
Выберите действие:
1. Добавить рейс
2. Вывести список рейсов
3. Поиск рейсов по типу самолета
4. Сохранить и выйти
Введите номер действия: 2
+-------------------------------+----------------------+-----------------+
|   Название пункта назначения  |     Номер рейса      |  Тип самолета   |
+-------------------------------+----------------------+-----------------+
| Питер-Москва                  | 12                   | Пассажирский    |
| Ростов-Ставрополь             | 23                   | Грузовой        |
+-------------------------------+----------------------+-----------------+
Выберите действие:
1. Добавить рейс
2. Вывести список рейсов
3. Поиск рейсов по типу самолета
4. Сохранить и выйти
Введите номер действия: 4
```

Рисунок 3 – Результат работы individual.py


```
[
    {
        "название пункта назначения": "Питер-Москва",
        "номер рейса": "12",
        "тип самолета": "Пассажирский"
    },
    {
        "название пункта назначения": "Ростов-Ставрополь",
        "номер рейса": "23",
        "тип самолета": "Грузовой"
    }
]
```


Рисунок 4 – Json файл

Задание повышенной сложности: Очевидно, что программа в примере 1 и в индивидуальном задании никак не проверяет правильность загружаемых данных формата JSON. В следствие чего, необходимо после загрузки из файла JSON выполнять валидацию загруженных данных. Валидацию данных необходимо производить с использованием спецификации JSON Schema, описанной на сайте https://json-schema.org/. Одним из возможных вариантов работы с JSON Schema является использование пакета jsonschema, который не является частью стандартной библиотеки Python. Таким образом, необходимо реализовать валидацию загруженных данных с помощью спецификации JSON Schema.

Входные данные (flight.json):

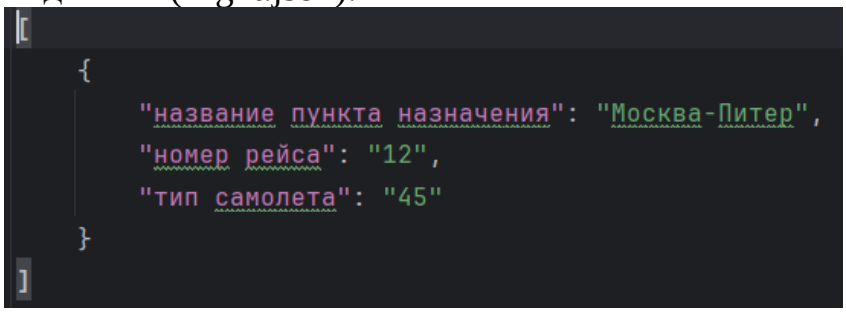

Рисунок 5 – Входные данные

Результат валидации (рис.6):

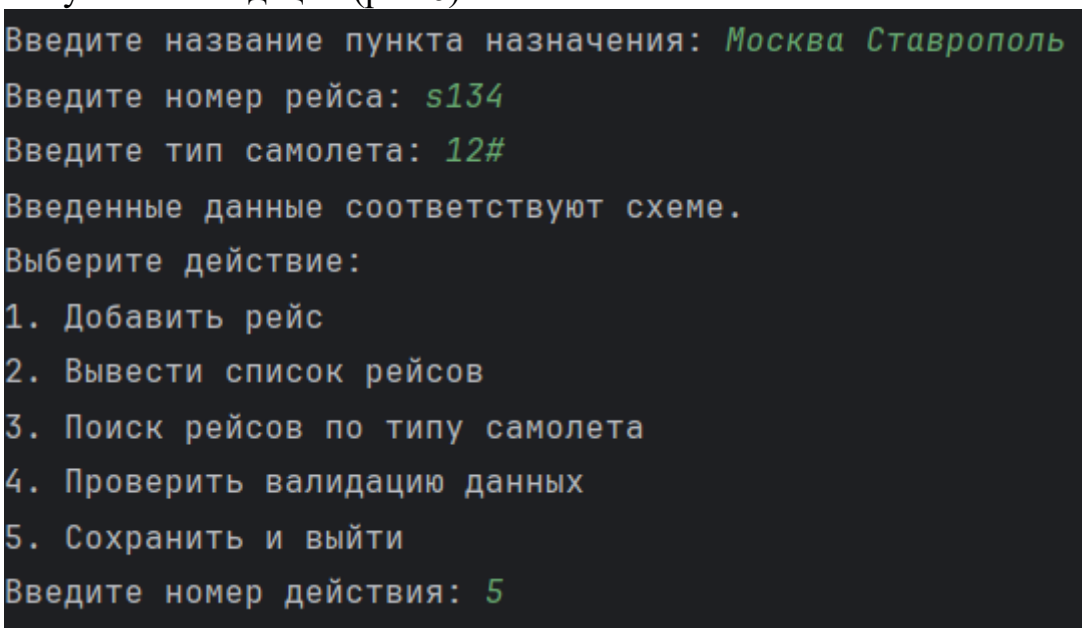

Рисунок 6 – Результат программы dop_individual.py

Ответы на контрольные вопросы:

1. Для чего используется JSON? JSON — это стандарт обмена данными. Он позволяет легко сериализовать и десериализовать объекты. Стандарт часто применяют, когда разрабатывают API и веб-приложения.

2. Какие типы значений используются в JSON? 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. Список (одномерный) — это упорядоченное множество значений. Список заключается в квадратные скобки «[ ]». Значения разделяются запятыми. 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.

3. Как организована работа со сложными данными в JSON? JSON позволяет организовать сложные структуры данных, такие как списки и вложенные словари (объекты). В JSON можно хранить разные типы данных, включая числа, строки, логические значения, массивы и объекты. Для организации сложных данных в JSON используются вложенные объекты и списки, позволяя создавать структуры данных любой сложности.

4. Самостоятельно ознакомьтесь с форматом данных JSON5? В чем отличие этого формата от формата данных JSON? JSON5 – это расширение формата данных JSON, разработанное для улучшения читаемости и удобства записи JSON-данных. Отличие JSON5 от обычного JSON включает в себя дополнительные возможности, такие как использование комментариев, разделителей ключей и значений, а также возможность использования одиночных кавычек вместо двойных. JSON5 является более гибким и читаемым форматом для записи данных, но не является стандартом и не поддерживается всеми JSON-парсерами.

5. Какие средства языка программирования Python могут быть использованы для работы с данными в формате JSON5? Для работы с данными в формате JSON5 на Python, можно использовать парсеры, поддерживающие

JSON5, такие как demjson. Однако, JSON5 не является стандартом, поэтому поддержка может быть ограничена.

6. Какие средства предоставляет язык Python для сериализации данных в формате JSON? Для сериализации данных в формат JSON в Python можно использовать модуль json. Он предоставляет функции json.dump() и json.dumps(), а также класс json.JSONEncoder, который может быть настроен для сериализации данных в формат JSON.

7. В чем отличие функций json.dump() и json.dumps()? «json.dump()» записывает данные в файл. Вы используете его, когда хотите сохранить данные в файле. json.dumps() превращает данные в строку. Вы используете его, когда хотите получить данные в виде строки для дальнейшей обработки, но не сохранять их в файле.

8. Какие средства предоставляет язык Python для десериализации данных из формата JSON? Для десериализации данных из формата JSON в Python используется модуль json, предоставляющий функции json.load() и json.loads().
9. Какие средства необходимо использовать для работы с данными формата JSON, содержащими кирилицу? Для работы с данными JSON, содержащими кириллицу, важно убедиться, что данные правильно кодируются и декодируются. Обычно это не вызывает проблем, поскольку JSON поддерживает Unicode, включая кириллические символы. Однако, при чтении и записи JSON-файлов, убедитесь, что правильно установлена кодировка (например, utf-8) для текстовых данных.

10. Самостоятельно ознакомьтесь со спецификацией JSON Schema? Что такое схема данных? JSON Schema - это спецификация, которая описывает формат данных JSON и правила их валидации. С помощью JSON Schema можно определить структуру, типы данных и ограничения для JSON-данных. JSON Schema используется для проверки соответствия данных определенным правилам. Это полезно, например, при валидации данных, получаемых из внешних источников. JSON Schema не является частью стандартной библиотеки Python, но существуют библиотеки и инструменты, поддерживающие JSON Schema, которые могут использоваться в Python.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки по работе с JSON-файлами, а также была изучена спецификация «JSON Schema», позволяющая проводить валидацию подгружаемых данных