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 ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины «Анализ данных»
Вариант №5

Выполнил:
Михеева Елена Александровна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика
и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных 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.

Порядок выполнения работы:

1. Был проработан пример из лабораторной работы.

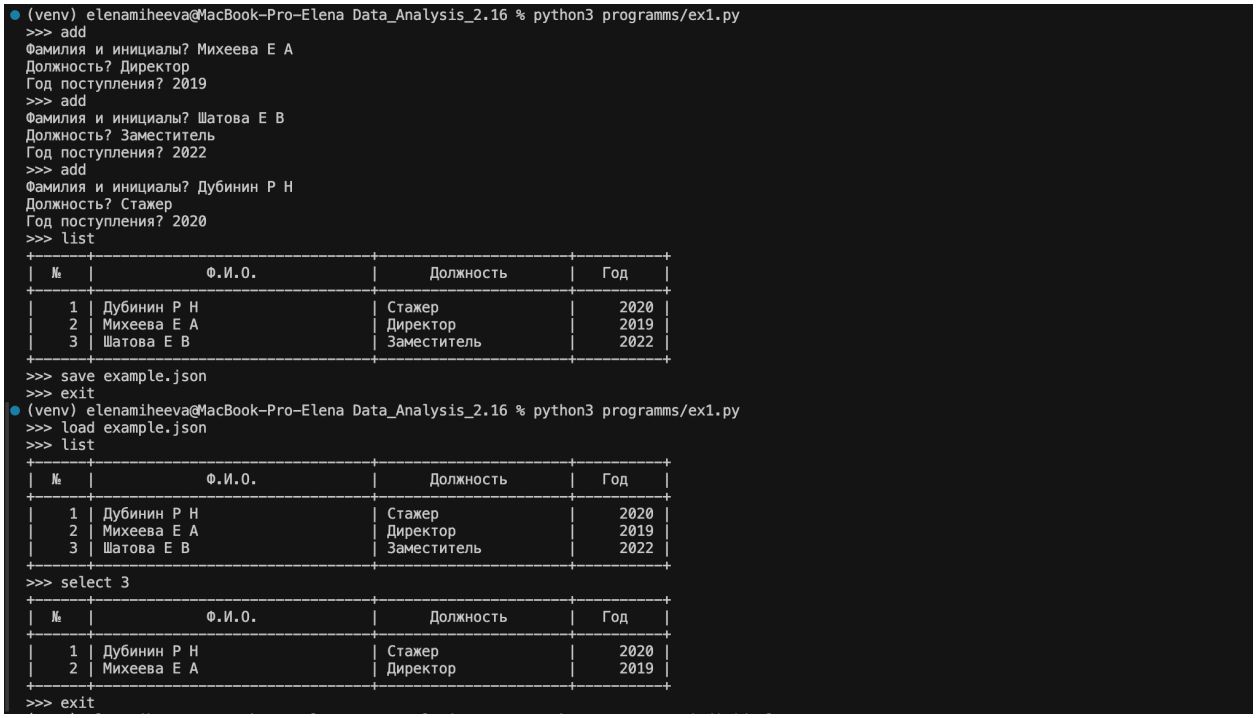

Рисунок 1. Выполнение примера 1

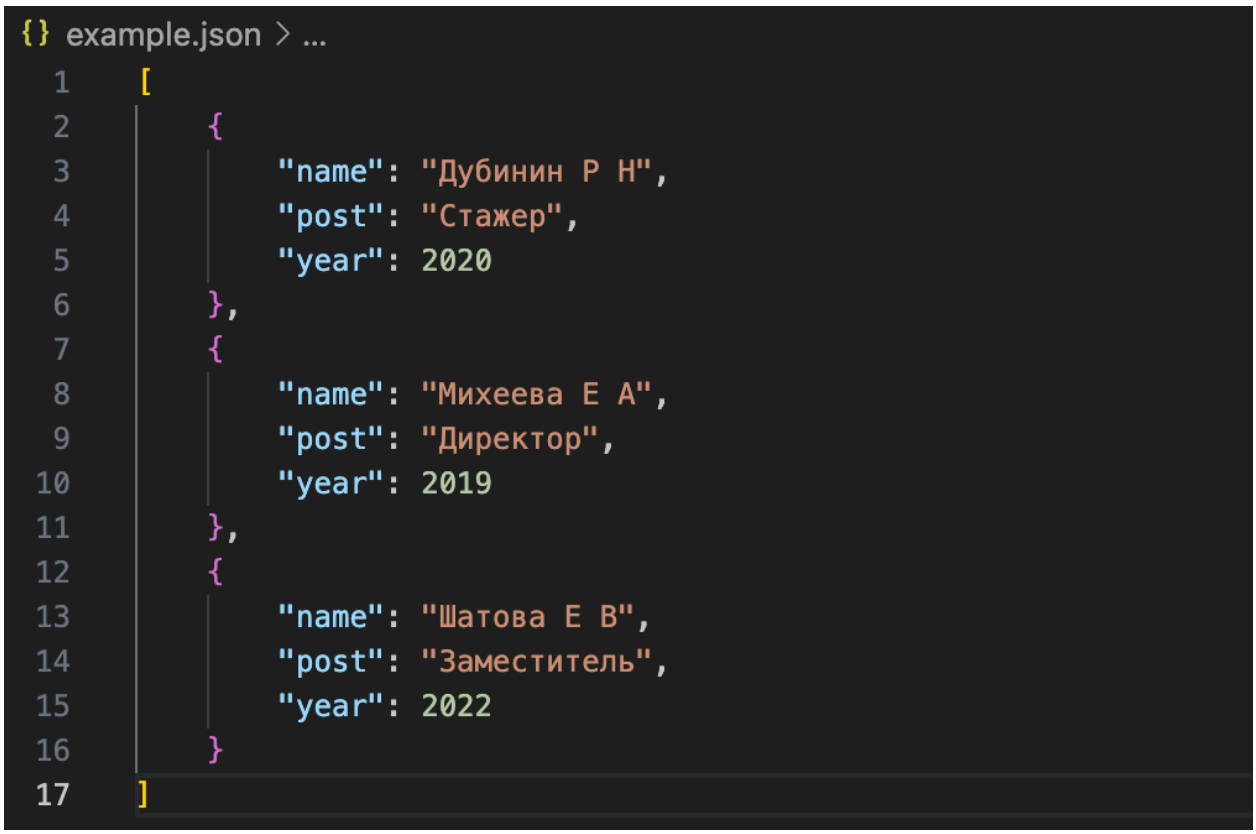

Рисунок 2. Файл json

2. Провели валидацию загруженных данных с помощью JSON Schema.

```python
def load_workers(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "name": {"type": "string"},
                "post": {"type": "string"},
                "year": {"type": "integer"}
            },
            "required": ["name", "post", "year"]
        }
    }
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        loaded = json.load(fin)
    try:
        jsonschema.validate(loaded, schema)
        print(">>> Data is obtained!")
    except jsonschema.exceptions.ValidationError as e:
        print(">>> Error:")
        print(e.message)  # Ошибка валидацци будет выведена на экран
    return loaded
```

Рисунок 3. Функция, отвечающая за загрузку и валидацию данных

```
(venv) elenamiheeva@MacBook-Pro-Elena Data_Analysis_2.16 % python3 programms/ex1.py
>>> load example.json
>>> Data is obtained!
>>> load flyex.json
>>> Error:
'name' is a required property
>>> list
+------+------------------------------+----------------------+----------+
|  №   |            Ф.И.О.            |      Должность       |   Год    |
+------+------------------------------+----------------------+----------+
|    1 |                              |                      |        0 |
|    2 |                              |                      |        0 |
|    3 |                              |                      |        0 |
+------+------------------------------+----------------------+----------+
>>> load example.json
>>> Data is obtained!
>>> list
+------+------------------------------+----------------------+----------+
|  №   |            Ф.И.О.            |      Должность       |   Год    |
+------+------------------------------+----------------------+----------+
|    1 | Дубинин Р Н                  | Стажер               |     2020 |
|    2 | Михеева Е А                  | Директор             |     2019 |
|    3 | Шатова Е В                   | Заместитель          |     2022 |
+------+------------------------------+----------------------+----------+
>>> exit
```

Рисунок 4. Проверка валидации загруженных данных

3. Приступили к выполнению индивидуального задания: использовать словарь, содержащий следующие ключи: название пункта

назначения рейса; номер рейса; тип самолета. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения; вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение. Необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON, а также валидацию загруженных данных

4. Были добавлены несколько функций, которые реализуют сохранение созданного списка и загрузку данных в формате json. Бесконечный цикл запроса команд организован в функции main().

```python
def save_flights(file_name, flights):
    """
    Сохранить всех работников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кирилицы установим ensure_ascii=False
        json.dump(flights, fout, ensure_ascii=False, indent=4)


def load_flights(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "destination": {"type": "string"},
                "flight number": {"type": "integer"},
                "type of plane": {"type": "string"}
            },
            "required": ["destination", "flight number", "type of plane"]
        }
    }
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        loaded = json.load(fin)
    try:
        jsonschema.validate(loaded, schema)
        print(">>> Data is obtained!")
    except jsonschema.exceptions.ValidationError as e:
        print(">>> Error:")
        print(e.message)  # Ошибка валидацци будет выведена на экран
    return loaded
```

Рисунок 5. Функции сохранения данных в файл и загрузки из файла

```python
def main():
    """
    Главная функция программы.
    """
    flights = []
    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()
        # Выполнить действие в соответствие с командой.
        if command == "exit":
            break

        elif command == "add":
            # Запросить данные о рейсе.
            flight = add_flight()
            # Добавить словарь в список.
            flights.append(flight)
            # Отсортировать список в случае необходимости.
            if len(flights) > 1:
                flights.sort(key=lambda item: item.get('destination', ''))

        elif command == "list":
            list_flights(flights)

        elif command == "find":
            find_flights(flights)

        elif command.startswith("save "):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]

            # Сохранить данные в файл с заданным именем.
            save_flights(file_name, flights)

        elif command.startswith("load "):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]

            # Сохранить данные в файл с заданным именем.
            flights = load_flights(file_name)

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить рейс;")
            print("list - вывести список рейсов;")
            print("find <тип> - запросить все рейсы обслуживаемые самолетом;")
            print("help - отобразить справку;")
            print("load - загрузить данные из файла;")
            print("save - сохранить данные в файл;")
            print("exit - завершить работу с программой.")
        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)


if __name__ == '__main__':
    main()
```

Рисунок 6. Функция main()

```
(venv) elenamiheeva@MacBook-Pro-Elena Data_Analysis_2.16 % python3 programms/individual.py
>>> load flyex.json
>>> Data is obtained!
>>> list
+------+--------------------------------+---------------------+----------------------+
|  No  |        Пункт назначения        |     Номер рейса     |     Тип самолета     |
+------+--------------------------------+---------------------+----------------------+
|    1 | Moscow                         | 245                 |                 td34 |
|    2 | tallin                         | 89                  |                 td34 |
|    3 | toronto                        | 654                 |                  tu4 |
>>> load example.json
>>> Error:
'destination' is a required property
>>>
```

Рисунок 7. Результат работы программы

Ответы на контрольные вопросы:

1. Для чего используется JSON?

Используется для передачи и хранения структурированных данных между компьютерами. Он часто используется в веб-разработке для обмена данными между сервером и клиентом, а также для сохранения конфигурационных файлов, передачи данных в API и в других сценариях.

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.
- 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?

Работа со сложными данными в JSON включает создание, чтение, обновление и удаление информации. Для создания JSON-данных, структуры данных в языке программирования преобразуются в JSON формат. При чтении JSON-данных, JSON-строка преобразуется в структуры данных языка программирования. Обновление данных включает изменение или добавление

значений в структурах данных, а удаление - исключение ненужных элементов. В случае вложенных структур, таких как массивы или объекты, операции выполняются на соответствующих уровнях.

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

JSON5 — это расширение формата данных JSON, предназначенное для повышения удобства написания и читаемости данных. В отличие от JSON, JSON5 позволяет использовать комментарии, одиночные кавычки для строковых значений, разделители в конце списков и объектов, а также не обязывает к использованию двойных кавычек для ключей в объектах. Это делает JSON5 более гибким и удобным для разработчиков, улучшая читаемость кода и упрощая процесс написания JSON-данных. Однако, JSON5 не является стандартом и не все библиотеки и инструменты поддерживают его, поэтому выбор между JSON и JSON5 зависит от конкретных потребностей проекта и его экосистемы.

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

Может быть использована библиотека json5.

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

json.dump() - конвертировать python объект в json и записать в файл.

json.dumps() - тоже самое, но в строку.

Обе эти функции принимают следующие необязательные аргументы: если skipkeys = True, то ключи словаря не базового типа (str, int, float, bool, None) будут проигнорированы, вместо того, чтобы вызывать исключение TypeError.

Если ensure_ascii = True, все не-ASCII символы в выводе будут экранированы последовательностями \uXXXX, и результатом будет строка, содержащая только ASCII символы. Если ensure_ascii = False, строки запишутся как есть.

Если check_circular = False, то проверка циклических ссылок будет пропущена, а такие ссылки будут вызывать OverflowError.

Если allow _nan = False, при попытке сериализовать значение с запятой, выходящее за допустимые пределы, будет вызываться ValueError (nan, inf, -inf) в строгом соответствии со спецификацией JSON, вместо того чтобы использовать эквиваленты из JavaScript (NaN, Infinity, -Infinity).

Если indent является неотрицательным числом, то массивы и объекты в JSON будут выводиться с этим уровнем отступа. Если уровень отступа 0, отрицательный или "', то вместо этого будут просто использоваться новые строки. Значение по умолчанию None отражает наиболее компактное представление. Если indent - строка, то она и будет использоваться в качестве отступа.

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?

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)

Обе эти функции принимают следующие аргументы:

- object_hook - опциональная функция, которая применяется к результату декодирования
- объекта (dict). Использоваться будет значение, возвращаемое этой функцией, а не полученный словарь.
- object_pairs_hook - опциональная функция, которая применяется к результату декодирования объекта с определённой последовательностью пар ключ/значение. Будет использован результат, возвращаемый функцией, вместо исходного словаря. Если задан так же object_hook , то приоритет отдаётся object_pairs_hook .

- parse_float , если определён, будет вызван для каждого значения JSON с плавающей точкой. По умолчанию, это эквивалентно float(num_str) .
- parse_int , если определён, будет вызван для строки JSON с числовым значением. По умолчанию эквивалентно int(num_str) .
- parse_constant , если определён, будет вызван для следующих строк: "-Infinity", "Infinity", "NaN". Может быть использовано для возбуждения исключений при обнаружении ошибочных чисел JSON.

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

ensure_ascii=False

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

SON Schema — это описание формата данных в форме схемы, которая определяет ожидаемую структуру, типы данных и другие ограничения. Она используется для валидации JSON-данных, гарантируя соответствие определенным правилам. JSON Schema предоставляет механизмы для определения обязательных и необязательных полей, типов данных, вложенных структур, и даже регулярных выражений для строковых значений. Разработчики могут использовать схемы данных для обеспечения консистентности и корректности обмена данными между системами.

Схема данных для примера 1:

```
{
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "name": {"type": "string"},
            "post": {"type": "string"},
            "year": {"type": "integer"}
        },
        "required": ["name", "post", "year"]
    }
}
```