

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 №2.16
дисциплины «Анализ данных»

Выполнил:
Магдаев Даламбек Магомедович
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., 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. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.
2. Проработал пример лабораторной работы:

```
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> add
Фамилия и инициалы? Magdaev D.M.
Должность? programmer
Год поступления? 2020
>>> add
Фамилия и инициалы? Ivanov I.I.
Должность? programmer
Год поступления? 2022
>>> select 3
+-----+-----+-----+-----+
| No |           Ф.И.О.           | Должность | Год |
+-----+-----+-----+-----+
|  1 | Magdaev D.M.               | programmer | 2020 |
+-----+-----+-----+-----+
>>> save file.json
>>> exit
```

Рисунок 1. Сохранение данных перед завершением работы программы

```

C:\Users\dalam\.conda\envs\pythonProject\python.exe C:\Users\dalam\Desktop\projects\study\data_analysis\lab_216\prim.py
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> load file.json
>>> list
+-----+-----+-----+-----+
| No |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Ivanov I.I.              | programmer           |      2022     |
|  2 | Magdaev D.M.             | programmer           |      2020     |
+-----+-----+-----+-----+
>>> |

```

Рисунок 2. Загрузка данных после повторного запуска программы

3. Выполнил индивидуальное задание №1: Для своего варианта лабораторной работы 2.8 необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON. Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.

```

>>> add
Название начального пункта маршрута? Stavropol
Название конечного пункта маршрута? Grozny
Номер маршрута? 25
>>> add
Название начального пункта маршрута? Moscow
Название конечного пункта маршрута? Stavropol
Номер маршрута? 46
>>> add
Название начального пункта маршрута? Grozny
Название конечного пункта маршрута? Stavropol
Номер маршрута? 25
>>> list
+-----+-----+-----+-----+
| № |      Начальный пункт      |      Конечный пункт      |      Номер маршрута      |
+-----+-----+-----+-----+
|  1 | Stavropol                 | Grozny                   |      25                   |
|  2 | Grozny                    | Stavropol                 |      25                   |
|  3 | Moscow                    | Stavropol                 |      46                   |
+-----+-----+-----+-----+
>>> select 25
+-----+-----+-----+-----+
| № |      Начальный пункт      |      Конечный пункт      |      Номер маршрута      |
+-----+-----+-----+-----+
|  1 | Stavropol                 | Grozny                   |      25                   |
|  2 | Grozny                    | Stavropol                 |      25                   |
+-----+-----+-----+-----+
>>> save route_num.json
>>> exit

```

Рисунок 3. Результат работы программы и сохранение данных в файл

```

C:\Users\dalam\.conda\envs\pythonProject\python.exe C:\Users\dalam\Desktop\projects\study\data_analysis\lab_216\ind.py
>>> help
Список команд:

add - добавить маршрут;
list - вывести список маршрутов;
select <номер маршрута> - запросить данные о маршруте;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> list
Список маршрутов пуст.
>>> load route_num.json
>>> list
+-----+-----+-----+-----+
| № | Начальный пункт | Конечный пункт | Номер маршрута |
+-----+-----+-----+-----+
| 1 | Stavropol | Grozny | 25 |
| 2 | Grozny | Stavropol | 25 |
| 3 | Moscow | Stavropol | 46 |
+-----+-----+-----+-----+
>>> |

```

Рисунок 4. Загрузка данных после повторного запуска программы

4. Выполнил индивидуальное задание №2: 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. 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.

```

C:\Users\dalam\.conda\envs\pythonProject\python.exe C:\Users\dalam\Desktop\projects\study\data_analysis\
>>> list
Список маршрутов пуст.
>>> load file.json
Ошибка валидации: 'start' is a required property
>>> load route_num.json
JSON валиден по схеме.
>>> list
+-----+-----+-----+-----+
| № | Начальный пункт | Конечный пункт | Номер маршрута |
+-----+-----+-----+-----+
| 1 | Stavropol | Grozny | 25 |
| 2 | Grozny | Stavropol | 25 |
| 3 | Moscow | Stavropol | 46 |
+-----+-----+-----+-----+
>>> |

```

Рисунок 5. Загрузка данных и валидация

Код индивидуального задания:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import json
import sys
from jsonschema import validate, ValidationError

def get_route():
    """
    Запросить данные о маршруте
    """
    start = input("Название начального пункта маршрута? ")
    finish = input("Название конечного пункта маршрута? ")
    number = int(input("Номер маршрута? "))

    return {
        'start': start,
        'finish': finish,
        'number': number,
    }

def display_route(routes):
    """
    Отобразить список маршрутов
    """
    if routes:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Начальный пункт",
                "Конечный пункт",
                "Номер маршрута"
            )
        )
        print(line)

    for idx, worker in enumerate(routes, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                idx,
                worker.get('start', ''),
                worker.get('finish', ''),
                worker.get('number', 0)
            )
        )
```

```

        print(line)
    else:
        print("Список маршрутов пуст.")

def select_route(routes, period):
    """
    Выбрать маршрут
    """
    result = []
    for employee in routes:
        if employee.get('number') == period:
            result.append(employee)

    return result

def save_routes(file_name, staff):
    """
    Сохранить данные в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_routes(file_name):
    """
    Загрузить данные из файла JSON.
    """
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "start": {"type": "string"},
                "finish": {"type": "string"},
                "number": {"type": "integer"},
            },
            "required": [
                "start",
                "finish",
                "number",
            ],
        },
    }
    # Открыть файл с заданным именем и прочитать его содержимое.
    with open(file_name, "r") as file_in:
        data = json.load(file_in) # Прочитать данные из файла

    try:
        # Валидация
        validate(instance=data, schema=schema)
        print("JSON валиден по схеме.")
    except ValidationError as e:
        print(f"Ошибка валидации: {e.message}")

```

```
return data
```

```
def main():
```

```
    """
```

```
    Главная функция программы
```

```
    """
```

```
    routes = []
```

```
    while True:
```

```
        command = input(">>> ").lower()
```

```
        if command == 'exit':
```

```
            break
```

```
        elif command == 'add':
```

```
            route = get_route()
```

```
            routes.append(route)
```

```
            if len(routes) > 1:
```

```
                routes.sort(key=lambda item: item.get('number', ""))
```

```
        elif command == 'list':
```

```
            display_route(routes)
```

```
        elif command.startswith('select'):
```

```
            parts = command.split(' ', maxsplit=1)
```

```
            period = int(parts[1])
```

```
            selected = select_route(routes, period)
```

```
            display_route(selected)
```

```
        elif command.startswith("save "):
```

```
            parts = command.split(maxsplit=1)
```

```
            file_name = parts[1]
```

```
            save_routes(file_name, routes)
```

```
        elif command.startswith("load "):
```

```
            parts = command.split(maxsplit=1)
```

```
            file_name = parts[1]
```

```
            routes = load_routes(file_name)
```

```
        elif command == 'help':
```

```
            print("Список команд:\n")
```

```
            print("add - добавить маршрут;")
```

```
            print("list - вывести список маршрутов;")
```

```
            print("select <номер маршрута> - запросить данные о маршруте;")
```

```
            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()
```

Ответы на контрольные вопросы:

1) Для чего используется JSON?

Ответ: JSON - текстовый формат обмена данными, основанный на JavaScript. JSON легко читаемый, его формат был разработан Дугласом Крокфордом. Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON. 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). Легко читаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента.

2) Какие типы значений используются в JSON?

Ответ: В качестве значений в JSON могут быть использованы: запись (неупорядоченное множество пар ключ-значение, заключённое в фигурные скобки «{ }»). Описывается строкой, между ним и значением стоит символ «:». Пары ключ-значение отделяются друг от друга запятыми), массив (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 (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?

Ответ: 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.

4) В чём отличие формата данных JSON5 от JSON?

Ответ: JSON5 – это расширение стандарта JSON, которое повышает читаемость и удобство написания JSON-данных. Главные отличия JSON5: допустимы комментарии, необязательно использовать кавычки для ключей (только если ключ состоит из букв, цифр или знаков подчёркивания и не является зарезервированным словом), есть специальный формат для дат и времени, поддерживает многострочный текст (это позволяет записывать строки без неудобного экранирования), допускает запись чисел с подчёркиваниями для улучшения читаемости, поддерживает шестнадцатеричную и восьмеричную системы счисления.

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

Ответ: `json5` – библиотека Python для работы с данными в формате JSON5. Она предоставляет функции для чтения и записи данных в/из формата JSON5. Команды: `json5.load(text)`, `json5.dump(text)`.

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

Ответ: `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()`?

Ответ: `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?

Ответ: `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 кириллицу?

Ответ: 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.

10) Что такое схема данных? Приведите схему данных для примера 1.

Ответ: JSON Schema – это распространенный стандарт описания структуры данных. Спецификация стандарта и популярные сценарии его использования доступны на ресурсе <http://json-schema.org/>. Схема создана для описания JSON-данных, но и сама она при этом является JSON-объектом. С помощью ключевых слов в схеме создаются правила валидации структуры объекта и типов его полей. Для примера №1 схема выглядит:

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

Вывод: в ходе выполнения лабораторной работы, приобретены навыки работы с данными формата JSON с помощью языка программирования Python версии 3.x.