

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 №19
дисциплины «Основы программной инженерии»

Выполнила:
Ламская Ксения Вячеславовна
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и
сопровождение программного
обеспечения», очная форма обучения

(подпись)

Доцент кафедры инфокоммуникаций
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Лабораторная работа 2.16. Работа с данными формата JSON в языке Python.

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


Порядок выполнения работы

1. Создание репозитория GitHub.

Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *	Repository name *
 ksenia-lamskaya ▾	/ 18laba
	✓ 18laba is available.

Great repository names are short and memorable. Need inspiration? How about [silver-dollop](#) ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 – Создание репозитория

2. Запуск Anaconda Powershell Prompt.

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

# вариант - 9

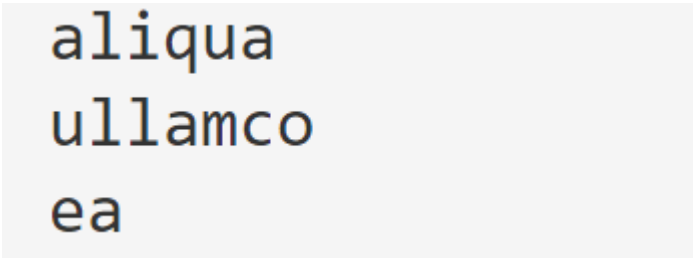
if __name__ == "__main__":

    vowels = {'a', 'e', 'i', 'o', 'u'}

    try:
        with open('./1.txt', 'r', encoding='utf-8') as file:
            text = file.readlines()
            for line in text:
                words = line.split()
                for word in words:
                    word = word.strip('.', ',')
                    if (word[0].lower() in vowels) and (word[-1].lower() in vowels):
                        print(word)

    except FileNotFoundError:
        print(f"There is no file with that name")
```

Рисунок 2.1 – Код программы



```
aliqua
ullamco
ea
```

Рисунок 2.2 – Результат программы

3. 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.

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

import json
import sys

def help():
    """
    Функция для вывода списка команд
```

```

"""
# Вывести справку о работе с программой.
print("Список команд:\n")
print("add - добавить маршрут;")
print("list - вывести список маршрутов;")
print("select <тип> - вывод на экран пунктов маршрута, используя номер
маршрута;")
print("help - отобразить справку;")
print("exit - завершить работу с программой.")
print("load - загрузить данные из файла;")
print("save - сохранить данные в файл;")

def load_point(file_name):
    with open(file_name, "r") as f:
        return json.load(f)

def save_point(file_name, point_list):
    with open(file_name, "w", encoding="utf-8") as f:
        json.dump(point_list, f, ensure_ascii=False, indent=4)

def add():
    """
    Функция для добавления информации о новых маршрутах
    """
    # Запросить данные о маршруте.
    name = input("Название начального пункта маршрута: ")
    name2 = input("Название конечного пункта маршрута: ")
    number = int(input("Номер маршрута: "))

    # Создать словарь.
    i = {'name': name, 'name2': name2, 'number': number}

    return i

def error(command):
    """
    функция для неопознанных команд
    """
    print(f"Неизвестная команда {command}")

def list(point):
    """
    Функция для вывода списка добавленных маршрутов
    """
    # Заголовок таблицы.
    line = '+--{}--{}--{}--{}--+'.format(
        '-' * 4,

```

```

'-' * 30,
'-' * 20,
'-' * 8
)
print(line)
print(
    '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
        "№",
        "Начальный пункт.",
        "Конечный пункт",
        "№ маршрута"
    )
)
print(line)

# Вывести данные о всех маршрутах.
for idx, i in enumerate(point, 1):
    print(
        '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
            idx,
            i.get('name', ''),
            i.get('name2', ''),
            i.get('number', '')
        )
    )
print(line)

def select(point):
    """
    Функция для получения маршрута по его номеру
    """

    # Разбить команду на части для выделения номера маршрута.
    parts = input("Введите значение: ")
    # Проверить сведения работников из списка.

    # Проверить сведения.
    flag = True
    for i in point:
        if i['number'] == int(parts):
            print("Начальный пункт маршрута - ", i["name"])
            print("Конечный пункт маршрута - ", i["name2"])
            flag = False
            break
    if flag:
        print("Маршрут с таким номером не найден")

def main():
    """
    Главная функция программы.

```

```

"""
print("Список команд:\n")
print("add - добавить маршрут;")
print("list - вывести список маршрутов;")
print("select <тип> - вывод на экран пунктов маршрута, используя номер
маршрута;")
print("help - отобразить справку;")
print("exit - завершить работу с программой.")
print("load - загрузить данные из файла;")
print("save - сохранить данные в файл;")

point = []

while True:
    command = (
        input("Введите команду (add, info, list, load, save, exit, help): ")
        .strip()
        .lower()
        .split(maxsplit=1)
    )

    match command:
        case ["exit"]:
            break

        case ["load", file_name]:
            new_point_list = load_point(file_name)
            if new_point_list:
                point = new_point_list

        case ["save", file_name]:
            save_point(file_name, point)

        case ["add"]:
            # Добавить словарь в список.
            i = add()
            point.append(i)
            # Отсортировать список в случае необходимости.
            if len(point) > 1:
                point.sort(key=lambda item: item.get('number', ''))

        case ["list"]:
            list(point)

        case ["select"]:
            select(point)

        case ["help"]:
            help()

        case _:

```

```

        print(f"Неизвестная команда {command[0]}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

Рисунок 3.1 – Командная строка

```

help - отобразить справку;
exit - завершить работу с программой.
load - загрузить данные из файла;
save - сохранить данные в файл;
Введите команду (add, info, list, load, save, exit, help): add
Название начального пункта маршрута: ddd
Название конечного пункта маршрута: fff
Номер маршрута: 3
Введите команду (add, info, list, load, save, exit, help): list
+-----+-----+-----+-----+
| № | Начальный пункт. | Конечный пункт | № маршрута |
+-----+-----+-----+-----+
| 1 | ddd | fff | 3 |
+-----+-----+-----+-----+
Введите команду (add, info, list, load, save, exit, help): █

```

Рисунок 3.2 – Результат программы

4. 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.

```

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

import json
import sys
from datetime import datetime

```

```

from jsonschema import validate
from jsonschema.exceptions import ValidationError

def validation(instance):
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "name1": {"type": "string"},
                "name2": {"type": "string"},
                "number": {"type": "number"},
            },
        },
        "required": ["name1", "name2", "number"],
    }

    try:
        validate(instance, schema=schema)
        return True
    except ValidationError as err:
        print(err.message)
        return False

def help():
    """
    Функция для вывода списка команд
    """
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <тип> - вывод на экран пунктов маршрута, используя номер маршрута;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
    print("load - загрузить данные из файла;")
    print("save - сохранить данные в файл;")

def load_point(file_name):
    with open(file_name, "r") as f:
        point = json.load(f)

    if validation(point):
        return point

def save_point(file_name, point_list):

```



```

with open(file_name, "w", encoding="utf-8") as f:
    json.dump(point_list, f, ensure_ascii=False, indent=4)

def add():
    """
    Функция для добавления информации о новых маршрутах
    """
    # Запросить данные о маршруте.
    name = input("Название начального пункта маршрута: ")
    name2 = input("Название конечного пункта маршрута: ")
    number = int(input("Номер маршрута: "))

    # Создать словарь.
    i = {'name': name, 'name2': name2, 'number': number}

    return i

def error(command):
    """
    функция для неопознанных команд
    """
    print(f"Неизвестная команда {command}")

def list(point):
    """
    Функция для вывода списка добавленных маршрутов
    """
    # Заголовок таблицы.
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 8
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
            "№",
            "Начальный пункт.",
            "Конечный пункт",
            "№ маршрута"
        )
    )
    print(line)

    # Вывести данные о всех маршрутах.
    for idx, i in enumerate(point, 1):
        print(

```

```

        | {:>4} | {:<30} | {:<20} | {:>8} |'.format(
            idx,
            i.get('name', ''),
            i.get('name2', ''),
            i.get('number', '')
        )
    )
    print(line)

def select(point):
    """
    Функция для получения маршрута по его номеру
    """
    # Разбить команду на части для выделения номера маршрута.
    parts = input("Введите значение: ")
    # Проверить сведения работников из списка.

    # Проверить сведения.
    flag = True
    for i in point:
        if i['number'] == int(parts):
            print("Начальный пункт маршрута - ", i["name"])
            print("Конечный пункт маршрута - ", i["name2"])
            flag = False
            break
    if flag:
        print("Маршрут с таким номером не найден")

def main():
    """
    Главная функция программы.
    """
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("list - вывести список маршрутов;")
    print("select <тип> - вывод на экран пунктов маршрута, используя номер маршрута;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")
    print("load - загрузить данные из файла;")
    print("save - сохранить данные в файл;")

    point = []

    while True:
        command = (
            input("Введите команду (add, info, list, load, save, exit, help): ")
            .strip()
            .lower()

```

```

        .split(maxsplit=1)
    )

    match command:
        case ["exit"]:
            break

        case ["load", file_name]:
            new_point_list = load_point(file_name)
            if new_point_list:
                point = new_point_list

        case ["save", file_name]:
            save_point(file_name, point)

        case ["add"]:
            # Добавить словарь в список.
            i = add()
            point.append(i)
            # Отсортировать список в случае необходимости.
            if len(point) > 1:
                point.sort(key=lambda item: item.get('number', ''))

        case ["list"]:
            list(point)

        case ["select"]:
            select(point)

        case ["help"]:
            help()

        case _:
            print(f"Неизвестная команда {command[0]}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

```

(191aba) C:\Ksen\191aba>python hard.py
Список команд:

add - добавить маршрут;
list - вывести список маршрутов;
select <тип> - вывод на экран пунктов маршрута, используя номер маршрута;
help - отобразить справку;
exit - завершить работу с программой.
load - загрузить данные из файла;
save - сохранить данные в файл;
Введите команду (add, info, list, load, save, exit, help): load data.json
Введите команду (add, info, list, load, save, exit, help): list
+-----+-----+-----+-----+
| № | Начальный пункт. | Конечный пункт | № маршрута |
+-----+-----+-----+-----+
| 1 | aaaaaa | ddd | 2 |
| 2 | dddsd | sdsdss | 4 |
+-----+-----+-----+-----+
Введите команду (add, info, list, load, save, exit, help): _

```

Рисунок 4.1 – Вывод программы

Контрольные вопросы:

1. JSON (JavaScript Object Notation) используется для обмена данными между различными приложениями. Он часто используется в веб-разработке для передачи данных между клиентом и сервером, а также для сохранения и передачи конфигурационных данных, настроек и других структурированных данных.

2. В JSON используются следующие типы значений:

- Строки (Strings)
- Числа (Numbers)
- Логические значения (Boolean: true или false)
- Массивы (Arrays)
- Объекты (Objects)
- Null (значение, представляющее отсутствие данных)

3. Работа со сложными данными в JSON организована путем вложения массивов и объектов друг в друга. Это позволяет создавать структурированные данные с любым уровнем вложенности и иерархии.

4. Формат данных JSON5 является расширением формата JSON и предоставляет дополнительные возможности и улучшения. Основные отличия JSON5 от JSON включают в себя поддержку комментариев, возможность использования одиночных кавычек для строковых значений, возможность использования необязательных запятых в конце массивов и объектов, а также поддержку расширенного синтаксиса чисел и ключевых слов.

5. Для работы с данными в формате JSON5 в Python можно использовать сторонние библиотеки, например, ``json5``, которая обеспечивает поддержку JSON5 в Python.

6. В языке Python для сериализации данных в формате JSON используются функции ``json.dump()`` и ``json.dumps()`` из модуля ``json``.

7. Основное отличие между ``json.dump()`` и ``json.dumps()`` заключается в том, что ``json.dump()`` записывает сериализованные данные в файл, в то время как ``json.dumps()`` возвращает строку JSON, которую можно использовать дальше в программе.

8. Для десериализации данных из формата JSON в Python используется функция ``json.load()`` для чтения из файла или ``json.loads()`` для чтения из строки JSON.

9. Для работы с данными формата JSON, содержащими кириллицу, необходимо обеспечить правильную кодировку при чтении и записи файлов,

а также при использовании строковых значений в программе. Обычно используется кодировка UTF-8.

10. JSON Schema - это спецификация для описания формата данных в формате JSON. Она определяет структуру данных, типы значений, ограничения и правила валидации для данных JSON. С помощью JSON Schema можно проверять соответствие данных определенной структуре и формату, а также автоматизировать проверку валидности данных.