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

Институт цифрового развития
Кафедра инфокоммуникаций

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.16
дисциплины «Программирование на языке Python»

Выполнила:
Мурашко Анастасия Юрьевна
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль) «Инфокоммуникационные системы и сети», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 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:

Пример 1

Для примера 1 лабораторной работы 2.8 добавьте возможность сохранения списка в файл формата JSON и чтения данных из файла JSON.

Решение: введем следующие команды для работы с файлом формата JSON в интерактивном режиме:

1. load - загрузить данные из файла, имя файла должно отделяться от команды load пробелом. Например: load data.json.

2. save - сохранить сделанные изменения в файл, имя файла должно отделяться от команды save пробелом. Например: save data.json.

Напишем программу для решения поставленной задачи.

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

import json
import sys
from datetime import date


def get_worker():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))

    # Создать словарь.
    return {
        'name': name,
        'post': post,
        'year': year,
    }


def display_workers(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
```

```
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
        print(line)
    else:
        print("Список работников пуст.")


def select_workers(staff, period):
    """
    Выбрать работников с заданным стажем.
    """

    # Получить текущую дату.
    today = date.today()

    # Сформировать список работников.
    result = []
    for employee in staff:
        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)

    # Возвратить список выбранных работников.
    return result


def save_workers(file_name, staff):
    """
    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(staff, fout, ensure_ascii=False, indent=4)


```

```python
def load_workers(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def main():
    """
    Главная функция программы.
    """
    # Список работников.
    workers = []

    # 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 работнике.
            worker = get_worker()

            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))
        elif command == "list":
            # Отобразить всех работников.
            display_workers(workers)

        elif command.startswith("select "):
            # Разбить команду на части для выделения стажа.
            parts = command.split(maxsplit=1)
            # Получить требуемый стаж.
            period = int(parts[1])

            # Выбрать работников с заданным стажем.
            selected = select_workers(workers, period)
            # Отобразить выбранных работников.
            display_workers(selected)

        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_workers(file_name, workers)

        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.
            workers = load_workers(file_name)

        elif command == 'help':
            # Вывести справку о работе с программой.
            print("Список команд:\n")
            print("add - добавить работника;")
            print("list - вывести список работников;")
            print("select <стаж> - запросить работников со стажем;")
            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 1. Код

```
>>> add
Фамилия и инициалы? Мурашко А.Ю
Должность? Студентка
Год поступления? 2022
>>> save primer.txt
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Мурашко А.Ю                      | Студентка            |     2022 |
+------+----------------------------------+----------------------+----------+
>>> list
Список работников пуст.
>>> load primer.txt
>>> list
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Мурашко А.Ю                      | Студентка            |     2022 |
+------+----------------------------------+----------------------+----------+
>>>
```

Рисунок 2. Результат выполнения

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.*

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

import sys
import json

list_route = []
spisok_new = []

def add_route():
    name_start = input('Начало маршрута: ')
    name_finish = input('Конец маршрута: ')
    num_route_get = input('Номер маршрута: ')

    num_route = int(num_route_get)

    if type(num_route) != int:
        print("Введенные данные не верны!")

    list_route_new = {
        'name_start': name_start,
        'name_finish': name_finish,
        'num_route': num_route
    }

    list_route.append(list_route_new)

    if len(list_route) > 1:
        list_route.sort(key=lambda item: item.get('num_route', ''))

def print_routes():
    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 15,
        '-' * 30,
        '-' * 20
    )

    print(line)
    print(
        '| {:^4} | {:^15} | {:^30} | {:^20} | '.format(
            "№",
            "Начало маршрута",
```

```
            "Конец маршрута",
            "№ Маршрута"
        )
    )
    print(line)

    for idx, spisok_new in enumerate(list_route, 1):
        print(
            '| {:>4} | {:<15} | {:<30} | {:<20} | '.format(
                idx,
                spisok_new.get('name_start', ''),
                spisok_new.get('name_finish', ''),
                spisok_new.get('num_route', 0)
            )
        )

    print(line)

def search_route():
    route_sear = input('Введите пункт маршрута: ')
    search_route = []
    for route_sear_itme in list_route:
        if route_sear == route_sear_itme['name_start']:
            search_route.append(route_sear_itme)
        if route_sear == route_sear_itme['name_finish']:
            search_route.append(route_sear_itme)

    if len(search_route) > 0:
        line_new = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 15,
                '-' * 30,
                '-' * 20
            )
        print(line_new)

        print(
                '| {:^4} | {:^15} | {:^30} | {:^20} | '.format(
                    "№",
                    "Начало маршрута",
                    "Конец маршрута",
                    "№ Маршрута"
```

```
            )
        )
        print(line_new)

        for idx_new, spisok_new_new in enumerate(search_route, 1):
            print(
                '| {:>4} | {:<15} | {:<30} | {:<20} | '.format(
                    idx_new,
                    spisok_new_new.get('name_start', ''),
                    spisok_new_new.get('name_finish', ''),
                    spisok_new_new.get('num_route', '')
                )
            )

        print(line_new)
    else:
        print('Таких маршрутов не найдено', file=sys.stderr)

def show_help():
    print('Список команд:\n')
    print('add - добавить маршрут.')
    print('list - вывести список маршрутов.')
    print('route <Пункуты маршрутов> - запросить Начало или конечные пункты маршрутов.')
    print('help - Справочник.')
    print('exit - Завершить работу программы.')
    print('save - сохранить в файл json.')
    print('load - загрузить список маршрутов из файла json.')
def main():
    while True:
        command = input('>>> ').lower()

        if command == 'exit':
            break

        elif command == 'add':
            add_route()

        elif command == 'list':
            print_routes()
        elif command == 'route':
            search_route()

```

```
128             elif command == 'help':
129                 show_help()
130             elif command == 'save':
131                 save_to_json('routes.json')
132             elif command == 'load':
133                 load_from_json('routes.json')
134 
135             else:
136                 print(f'Команда <{command}> не существует.', file=sys.stderr)
137                 print('Введите <help> для просмотра доступных команд')
138 
139 # Функция для сохранения данных в файл JSON
140 def save_to_json(filename):
141     with open(filename, 'w') as f:
142         json.dump(list_route, f)
143 
144 # Функция для загрузки данных из файла JSON
145 def load_from_json(filename):
146     global list_route
147     try:
148         with open(filename, 'r') as f:
149             list_route = json.load(f)
150     except FileNotFoundError:
151         list_route = []
152 
153 if __name__ == "__main__":
154     # хранится файл JSON
155     json_filename = 'routes.json'
156 
157     # Загрузка данных из файла JSON
158     load_from_json('routes.json')
159 
160     try:
161         main()
162     finally:
163         # Сохранение данных в файл JSON перед завершением программы
164         save_to_json('routes.json')
165 
```

Рисунок 3. Код

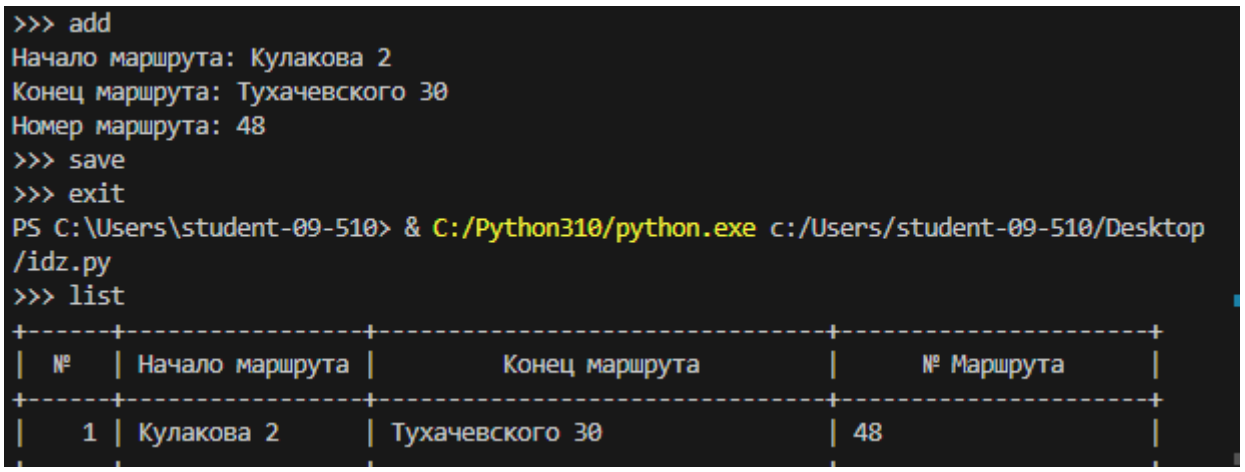

Рисунок 4. Результат выполнения индивидуального задания

Ответы на контрольные вопросы:

1. Для чего используется JSON?

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?

Запись, массив, число, литералы, строка.

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 - предложенное расширение формата json в соответствии с синтаксисом ECMAScript 5, вызванное тем, что json используется не только для общения между программами, но и создаётся/редактируется вручную. Файл JSON5 всегда является корректным кодом ECMAScript 5. 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.

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

JSON5 расширяет формат обмена данными JSON, чтобы сделать его немного более удобным в качестве языка конфигурации:

1. Комментарии в стиле JavaScript (как однострочные, так и

многострочные) являются законными.

2. Ключи объектов могут быть без кавычек, если они являются законными идентификаторами ECMAScript.

3. Объекты и массивы могут заканчиваться запятыми.

4. Строки могут заканчиваться в одинарные кавычки, и допускаются многострочные строковые литералы.

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

Модуль json предоставляет удобный метод dump() для записи данных в файл. Существует также метод dumps() для записи данных в обычную строку. Типы данных Python кодируются в формат JSON в соответствии с интуитивно понятными правилами преобразования.

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

Функции json.dump() и json.dumps() - это две разные функции из модуля json в Python, и они выполняют разные задачи:

1) json.dump(): Эта функция используется для записи данных в формате JSON в файл. Она принимает два аргумента: данные, которые вы хотите записать, и файловый объект (или объект, поддерживающий запись, такой как io.TextIOBase). Эта функция записывает данные в указанный файл в формате JSON.

2) json.dumps(): Эта функция используется для преобразования данных в формате JSON в строку (текстовое представление). Она принимает один аргумент - данные, которые вы хотите преобразовать. Эта функция возвращает строку, содержащую данные в формате JSON.

Итак, основное отличие между ними заключается в том, что json.dump() записывает данные в файл, а json.dumps() возвращает JSON-строку. Выбор между ними зависит от того, какие действия вы хотите выполнить: сохранить данные в файл или получить JSON-строку для дальнейшей обработки.

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

В модуле json определены методы load() и loads(), предназначенные для преобразования кодированных в формате JSON данных в объекты Python.

Подобно операции сериализации, также существует таблица преобразования типов, определяющая правила для обратного декодирования данных.

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

Параметр ensure_ascii.

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 лабораторной работы приобретены навыки по работе с данными формата JSON с помощью языка программирования Python версии 3.x.