

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 29

Выполнил:
Саенко Андрей Максимович
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. технических
наук, 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:

Проработаны примеры лабораторной работы:

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

Код программы:

```
#!/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(
            "№",
            "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 кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

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.

```

```

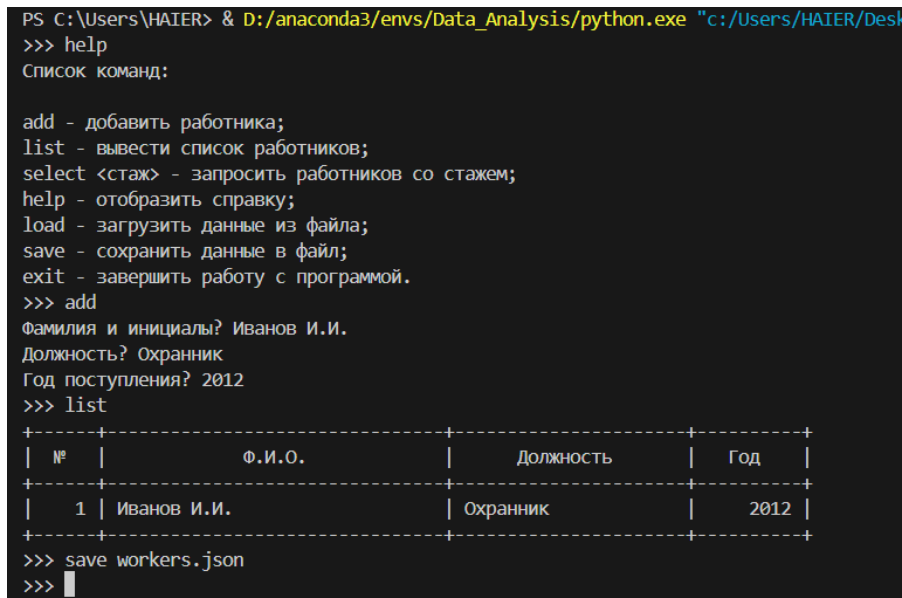
print("Список команд:\n")
print("add - добавить работника;")
print("list - вывести список работников;")
print("select <стаж> - запросить работников со стажем;")
print("help - отобразить справку;")
print("load - загрузить данные из файла;")
print("save - сохранить данные в файл;")
print("exit - завершить работу с программой.")

else:
    print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

Результат работы программы:



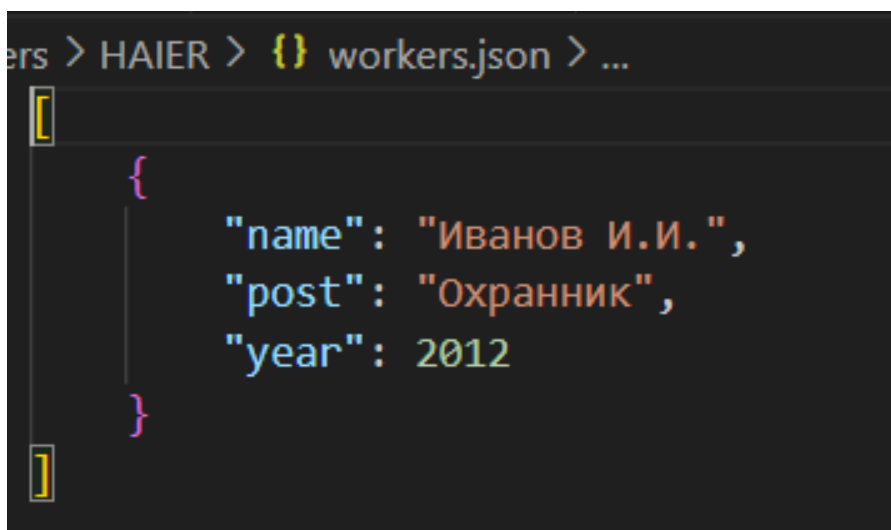
```

PS C:\Users\HAIER> & D:/anaconda3/envs/Data_Analysis/python.exe "c:/Users/HAIER/Desktop/workers.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.
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? Охранник
Год поступления? 2012
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Иванов И.И. | Охранник | 2012 |
+-----+-----+-----+-----+
>>> save workers.json
>>>

```

Рисунок 1 – Добавление работника и создание файла workers.json



```

[
  {
    "name": "Иванов И.И.",
    "post": "Охранник",
    "year": 2012
  }
]

```

Рисунок 2 – Содержимое файла workers.json

```

>>> list
Список работников пуст.
>>> load workers.json
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Иванов И.И. | Охранник | 2012 |
+-----+-----+-----+-----+
>>> 

```

Рисунок 3 – Загрузка данных из файла workers.json

Выполнены задания из лабораторной работы:

Задание 1. Для своего варианта лабораторной работы 11 (Программирование на python) 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.

Код программы:

```

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

import sys
import json

def add()::
    """
    Добавить маршрут
    """
    name_start = input("Начальный пункт маршрута? ")
    name_end = input("Конечный пункт маршрута? ")
    number = int(input("Номер маршрута? "))

    route = {
        'name_start': name_start,
        'name_end': name_end,
        'number': number
    }
    return route

def list(routes):
    """
    Вывести список маршрутов
    """
    if routes:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 30,
            '-' * 30,

```

```

        '_' * 8
    )
    print(line)

    print("| {:^4} | {:^30} | {:^30} | {:^8} |".format(
        "№",
        "Начальный пункт",
        "Конечный пункт",
        "Номер"
    )
    )
    print(line)

    for idx, route in enumerate(routes, 1):
        print("| {:>4} | {:<30} | {:<30} | {:>8} |".format(
            idx,
            route.get('name_start', ""),
            route.get('name_end', ""),
            route.get('number', 0)
        )
        )
        print(line)
    else:
        print("Список работников пуст.")

def save_routes(file_name, staff):
    """
    Сохранить все маршруты в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_routes(file_name):
    """
    Загрузить все маршруты из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def select(routes, command):
    """
    Вывести выбранные маршруты
    """
    parts = command.split(' ', maxsplit=1)
    station = parts[1]
    count = 0

    for route in routes:
        if (station == route["name_start"].lower() or
            station == route["name_end"].lower()):

```

```

        count += 1
        print('{:>4}: {}-{}, номер маршрута: {}'.format(count,
            route["name_start"], route["name_end"], route["number"]))

    if count == 0:
        print("Маршрут не найден.")

def help():
    """
    Вывести список команд
    """
    print("Список команд:\n")
    print("add - добавить маршрут;")
    print("load - загрузить данные из файла;")
    print("save - сохранить данные в файл;")
    print("list - вывести список маршрутов;")
    print("select <пункт> - запросить информацию" +
        " о маршруте с указанным пунктом;")
    print("help - отобразить справку;")
    print("exit - завершить работу с программой.")

def main():
    """
    Основная функция
    """
    routes = []

    while True:

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

        if command == 'exit':
            break

        elif command == 'add':
            route = add()
            routes.append(route)

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

        elif command == 'list':
            list(routes)

        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            routes = load_routes(file_name)

        elif command.startswith("save "):
            parts = command.split(maxsplit=1)

```



```

    file_name = parts[1]
    save_routes(file_name, routes)

elif command.startswith('select '):
    select(routes, command)

elif command == 'help':
    help()

else:
    print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

Результат работы программы:

```

>>> help
Список команд:

add - добавить маршрут;
load - загрузить данные из файла;
save - сохранить данные в файл;
list - вывести список маршрутов;
select <пункт> - запросить информацию о маршруте с указанным пунктом;
help - отобразить справку;
exit - завершить работу с программой.
>>> add
Начальный пункт маршрута? Ставрополь
Конечный пункт маршрута? Краснодар
Номер маршрута? 1
>>> list
+-----+-----+-----+
| № | Начальный пункт | Конечный пункт | Номер |
+-----+-----+-----+
| 1 | Ставрополь | Краснодар | 1 |
+-----+-----+-----+
>>> save routes.json
>>>

```

Рисунок 4 – Добавление маршрута и создание файла routes.json

```

>>> list
Список маршрутов пуст.
>>> load routes.json
>>> list
+-----+-----+-----+
| № | Начальный пункт | Конечный пункт | Номер |
+-----+-----+-----+
| 1 | Ставрополь | Краснодар | 1 |
+-----+-----+-----+
>>>

```

Рисунок 5 – Загрузка маршрута из файла routes.json

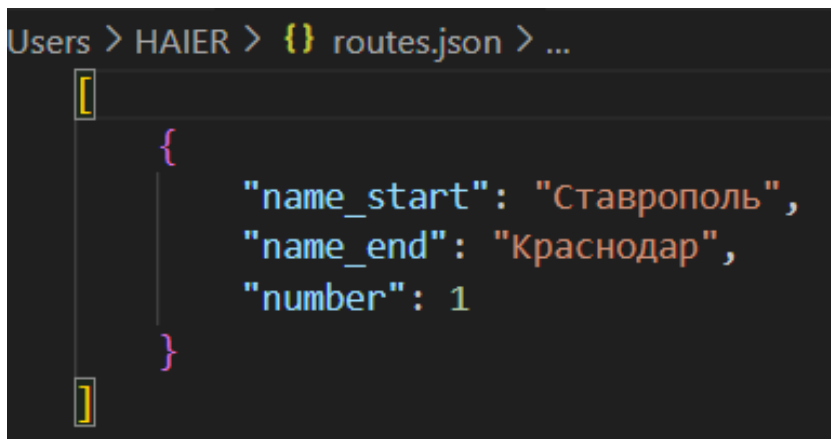


Рисунок 6 – Содержимое файла routes.json

Выполнено задание повышенной сложности:

Необходимо реализовать валидацию загруженных данных в примере 1 и в индивидуальном задании с помощью спецификации JSON Schema

Изменён пример 1:

Код программы:

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

import json
import sys
import jsonschema

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(
    "№",
    "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 кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_workers(file_name):
    """

```

Загрузить всех работников из файла JSON.

"""

Открыть файл с заданным именем для чтения.

```
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:
    data = json.load(fin)
```

try:

```
    jsonschema.validate(data, schema)
    print(">>> Данные получены")
```

except jsonschema.exceptions.ValidationError as e:

```
    print(">>> Ошибка:")
    print(e.message)
```

return data

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 соответствии 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()

```

Результат работы программы:

```
Users > HAIER > {} workers.json > ...  
1  [  
2      {  
3          "name": "Иванов И.И.",  
4          "post": "Охранник",  
5          "y": 2012  
6      }  
7  ]
```

Рисунок 7 – Содержимое файла workers.json

```
>>> load workers.json  
>>> Ошибка:  
'year' is a required property  
>>>
```

Рисунок 8 – Ошибка при загрузке данных из файла

Изменён код программы из индивидуального задания:

Код программы:

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

```
#Необходимо реализовать валидацию загруженных данных  
#в примере 1 и в индивидуальном задании с помощью спецификации JSON Schema
```

```
import sys  
import json  
import jsonschema
```

```
def add():  
    """  
    Добавить маршрут  
    """  
    name_start = input("Начальный пункт маршрута? ")  
    name_end = input("Конечный пункт маршрута? ")  
    number = int(input("Номер маршрута? "))  
  
    route = {
```

```

        'name_start': name_start,
        'name_end': name_end,
        'number': number
    }
    return route

def list(routes):
    """
    Вывести список маршрутов
    """
    if routes:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 30,
            '-' * 8
        )
        print(line)

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

    for idx, route in enumerate(routes, 1):
        print('| {:>4} | {:<30} | {:<30} | {:>8} |'.format(
            idx,
            route.get('name_start', ''),
            route.get('name_end', ''),
            route.get('number', 0)
        ))
    )
    print(line)
else:
    print("Список маршрутов пуст.")

def save_routes(file_name, staff):
    """
    Сохранить все маршруты в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_routes(file_name):
    """
    Загрузить все маршруты из файла JSON.
    """

```

```

"""
# Открыть файл с заданным именем для чтения.
schema = {
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "name_start": {"type": "string"},
            "name_end": {"type": "string"},
            "number": {"type": "integer"}
        },
        "required": ["name_start", "name_end", "number"]
    }
}

# Открыть файл с заданным именем для чтения.
with open(file_name, "r", encoding="utf-8") as fin:
    data = json.load(fin)

try:
    jsonschema.validate(data, schema)
    print(">>> Данные получены")

except jsonschema.exceptions.ValidationError as e:
    print(">>> Ошибка:")
    print(e.message)
return data

def select(routes, command):
    """
    Вывести выбранные маршруты
    """
    parts = command.split(' ', maxsplit=1)
    station = parts[1]
    count = 0

    for route in routes:
        if (station == route["name_start"].lower() or
            station == route["name_end"].lower()):

            count += 1
            print('{:>4}: {}-{}, номер маршрута: {}'.format(count,
                route["name_start"], route["name_end"], route["number"]))

    if count == 0:
        print("Маршрут не найден.")

def help():
    """
    Вывести список команд
    """
    print("Список команд:\n")

```



```

print("add - добавить маршрут;")
print("load - загрузить данные из файла;")
print("save - сохранить данные в файл;")
print("list - вывести список маршрутов;")
print("select <пункт> - запросить информацию" +
      " о маршруте с указанным пунктом;")
print("help - отобразить справку;")
print("exit - завершить работу с программой.")

def main():
    """
    Основная функция
    """
    routes = []

    while True:

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

        if command == 'exit':
            break

        elif command == 'add':
            route = add()
            routes.append(route)

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

        elif command == 'list':
            list(routes)

        elif command.startswith("load "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            routes = load_routes(file_name)

        elif command.startswith("save "):
            parts = command.split(maxsplit=1)
            file_name = parts[1]
            save_routes(file_name, routes)

        elif command.startswith('select '):
            select(routes, command)

        elif command == 'help':
            help()

        else:
            print(f'Неизвестная команда {command}', file=sys.stderr)

if __name__ == '__main__':

```

main()

Результат работы программы:

```
>>> load routes.json
>>> Ошибка:
'number' is a required property
>>> █
```

Рисунок 9 – Ошибка при загрузке данных из файла

```
sers > HAIER > {} routes.json > {} 0 > [abc] name_st
[
  {
    "name_start": "Ставрополь",
    "name_end": "Краснодар",
    "n": 1
  }
]
```

Рисунок 10 – Содержимое файла routes.json

Ответы на контрольные вопросы:

1. Для чего используется JSON?

JSON используется для хранения структурированных данных и обмена ими.

2. Какие типы значений используются в JSON?

В качестве значений в JSON могут быть использованы: запись, массив, число (целое или вещественное), литералы, строка.

3. Как организована работа со сложными данными в JSON?

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

Формат обмена данными JSON5 (JSON5) — это надмножество JSON, цель которого — облегчить некоторые ограничения JSON за счет расширения его синтаксиса за счет включения некоторых продуктов из ECMAScript 5.1

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

Для работы с данными в формате JSON5 может использоваться 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)

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))

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

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

JSON - это инструмент для проверки структуры данных JSON.

Схема данных для примера 1:

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

Выводы: получены навыки использования json при написании программ на языке программирования Python