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 №2.17
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: Разработка приложений с интерфейсом командной строки (CLI) в Python3

Цель: приобретение навыков построения приложений с интерфейсом командной строки 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:

```
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py -h
usage: workers [-h] [--version] {add,display,select} ...

positional arguments:
  {add,display,select}
    add                 Add a new worker
    display             Display all workers
    select              Select the workers

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py add -h
usage: workers add [-h] -n NAME [-p POST] -y YEAR filename

positional arguments:
  filename              The data file name
```

Рисунок 1. Страницы руководства

```
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py add -n Магдаев -p Студент -y 2022 data.json
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py add -n Иванов -p Директор -y 2012 data.json
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py add -n Сидоров -p Бухгалтер -y 1988 data.json
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py display data.json
+------+--------------------------------+----------------------+----------+
|  №   |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Магдаев                        | Студент              |     2022 |
+------+--------------------------------+----------------------+----------+
|    2 | Иванов                         | Директор             |     2012 |
+------+--------------------------------+----------------------+----------+
|    3 | Сидоров                        | Бухгалтер            |     1988 |
+------+--------------------------------+----------------------+----------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py select -h
usage: workers select [-h] -P PERIOD filename

positional arguments:
  filename              The data file name

options:
  -h, --help            show this help message and exit
  -P PERIOD, --period PERIOD
                        The required period
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python prim.py select -P 5 data.json
+------+--------------------------------+----------------------+----------+
|  №   |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Иванов                         | Директор             |     2012 |
+------+--------------------------------+----------------------+----------+
|    2 | Сидоров                        | Бухгалтер            |     1988 |
+------+--------------------------------+----------------------+----------+
```

Рисунок 2. Ввод, вывод и выбор работников в консоли

3. Выполнил индивидуальное задание: для своего варианта лабораторной работы 2.16 необходимо дополнительно реализовать интерфейс командной строки (CLI).

```
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind.py -h
usage: routes [-h] [--version] {add,display,select} ...

positional arguments:
  {add,display,select}
    add                 Add a new route
    display             Display all routes
    select              Select the route

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind.py display data_ind.json
JSON валиден по схеме.
+------+--------------------------------+----------------------+----------------+
|  №   |        Начальный пункт         |    Конечный пункт    | Номер маршрута |
+------+--------------------------------+----------------------+----------------+
|    1 | russia                         | usa                  |             12 |
|    2 | russia                         | italy                |             45 |
+------+--------------------------------+----------------------+----------------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind.py add -s usa -f russia -n 12 data_ind.json
JSON валиден по схеме.
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind.py add -s paris -f brazil -n 12 data_ind.json
JSON валиден по схеме.
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind.py display data_ind.json
JSON валиден по схеме.
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind.py display data_ind.json
JSON валиден по схеме.
+------+--------------------------------+----------------------+----------------+
|  №   |        Начальный пункт         |    Конечный пункт    | Номер маршрута |
+------+--------------------------------+----------------------+----------------+
|    1 | russia                         | usa                  |             12 |
|    2 | russia                         | italy                |             45 |
|    3 | usa                            | russia               |             12 |
|    4 | paris                          | brazil               |             12 |
+------+--------------------------------+----------------------+----------------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind.py select -N 12 data_ind.json
JSON валиден по схеме.
+------+--------------------------------+----------------------+----------------+
|  №   |        Начальный пункт         |    Конечный пункт    | Номер маршрута |
+------+--------------------------------+----------------------+----------------+
|    1 | russia                         | usa                  |             12 |
|    2 | usa                            | russia               |             12 |
|    3 | paris                          | brazil               |             12 |
+------+--------------------------------+----------------------+----------------+
```

Рисунок 3. Страницы руководства и результат работы программы

Код индивидуального задания №1:

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

# Для своего варианта лабораторной работы 2.16 необходимо дополнительно реализовать
# интерфейс командной строки (CLI).

import argparse
import json
import os.path
from jsonschema import validate, ValidationError


def add_route(routes, start, finish, number):
```

```
    """
    Добавить данные о маршруте
    """
    routes.append(
        {
            'start': start,
            'finish': finish,
            'number': number
        }
    )
    return routes


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, routes):
    """
        Сохранить данные в файл JSON
        """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(routes, 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(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )
```

```
# Создать основной парсер командной строки.
parser = argparse.ArgumentParser("routes")
parser.add_argument(
    "--version",
    action="version",
    version="%(prog)s 0.1.0"
)
subparsers = parser.add_subparsers(dest="command")
# Создать субпарсер для добавления маршрута.
add = subparsers.add_parser(
    "add",
    parents=[file_parser],
    help="Add a new route"
)
add.add_argument(
    "-s",
    "--start",
    action="store",
    required=True,
    help="The start of the route"
)
add.add_argument(
    "-f",
    "--finish",
    action="store",
    help="The finish of the route"
)
add.add_argument(
    "-n",
    "--number",
    action="store",
    type=int,
    required=True,
    help="The number of the route"
)
# Создать субпарсер для отображения всех маршрутов.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all routes"
)
# Создать субпарсер для выбора маршрута.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the route"
)
select.add_argument(
    "-N",
    "--numb",
    action="store",
    type=int,
    required=True,
    help="The route"
)
```

```
    # Выполнить разбор аргументов командной строки.
    args = parser.parse_args(command_line)

    # Загрузить все маршруты из файла, если файл существует.
    is_dirty = False
    if os.path.exists(args.filename):
        routes = load_routes(args.filename)
    else:
        routes = []

    # Добавить маршрут.
    if args.command == "add":
        routes = add_route(
            routes,
            args.start,
            args.finish,
            args.number
        )
        is_dirty = True

    # Отобразить все маршруты.
    elif args.command == "display":
        display_route(routes)

    # Выбрать требуемые маршруты.
    elif args.command == "select":
        selected = select_route(routes, args.numb)
        display_route(selected)

    # Сохранить данные в файл, если список маршрутов был изменен.
    if is_dirty:
        save_routes(args.filename, routes)


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

4. Выполнил задание повышенной сложности: Самостоятельно изучите работу с пакетом click для построения интерфейса командной строки (CLI). Для своего варианта лабораторной работы 2.16 необходимо реализовать интерфейс командной строки с использованием пакета click.

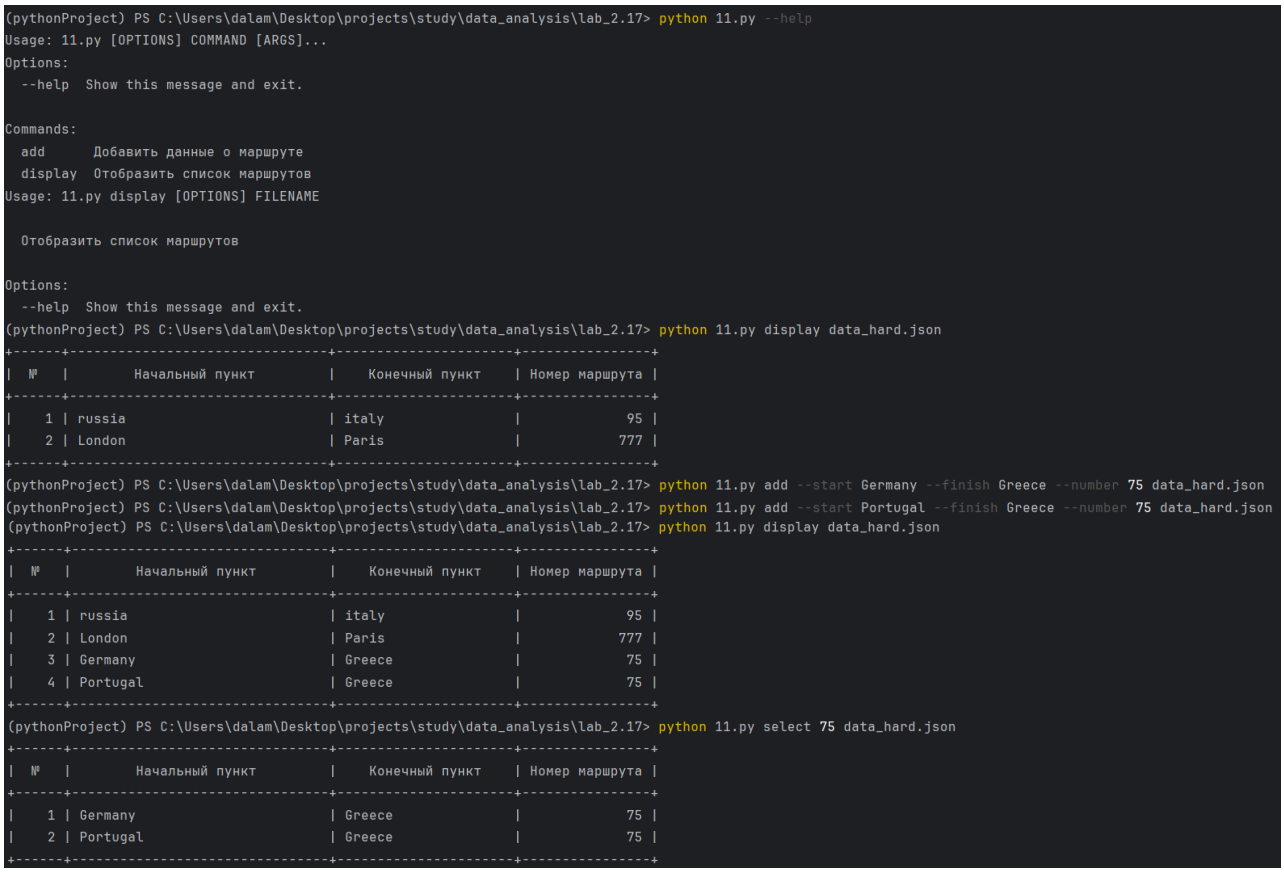

Рисунок 4. Страницы руководства и результат работы программы

Код индивидуального задания №2:

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

import json
import click


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


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)

@click.group()
def commands():
    pass

@commands.command("add")
@click.argument("filename")
@click.option("--start", help="Start")
@click.option("--finish", help="Finish")
@click.option("--number", help="Number")
def add(filename, start, finish, number):
    """
    Добавить данные о маршруте
    """
    routes = load_routes(filename)
    route = {
        "start": start,
        "finish": finish,
        "number": number,
    }
    routes.append(route)
    save_routes(filename, routes)

@commands.command("display")
@click.argument("filename")
```

```
def display(filename):
    """
    Отобразить список маршрутов
    """
    routes = load_routes(filename)
    display_routes(routes)

@commands.command("select")
@click.argument("number")
@click.argument("filename")
def select(filename, number):
    """
    Выбрать маршрут с заданным номером
    """
    routes = load_routes(filename)
    result = []
    for route in routes:
        if route.get("number") == number:
            result.append(route)

    display_routes(result)

def main():
    commands()

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

Ответы на контрольные вопросы:

1) Чем отличаются терминал и консоль?

Ответ: терминал – программа-оболочка, запускающая оболочку и позволяющая вводить команды. Консоль – разновидность терминала, это окно, в котором активны программы текстового режима.

2) Что такое консольное приложение?

Ответ: консольное приложение – программа, не имеющая графического интерфейса (окон), и которая работает в текстовом режиме в консоли. Команды в такой программе нужно вводить с клавиатуры, результаты работы консольные приложения также выводят на экран в текстовом виде.

3) Какие существуют средства языка программирования Python для построения приложений командной строки?

Ответ: модуль sys (предоставляет доступ к некоторым переменным и функциям, взаимодействующим с интерпретатором Python) и модуль argparse (Позволяет создавать красивые и гибкие интерфейсы командной строки с

автоматической генерацией справки и поддержкой нескольких параметров командной строки).

4) Какие особенности построения CLI с использованием модуля sys?

Ответ: sys.argv – позволяет получить список аргументов командной строки. Эквивалент argc – количество элементов в списке (Получается от len()).

5) Какие особенности построения CLI с использованием модуля getopt?

Ответ: Модуль getopt в Python расширяет разделение входной строки проверкой параметров. Основанный на функции C getopt, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений. Удобен для простых CLI, но может быть не так гибок и мощен, как argparse.

6) Какие особенности построения CLI с использованием модуля argparse?

Ответ: особенности построения CLI с использованием модуля argparse: Поддержка создания позиционных аргументов и флагов.

a) Возможность создания подкоманд для более сложных CLI.

b) Автоматическая генерация справки.

c) Поддержка типизации аргументов и их ограничений.

d) Гибкая конфигурация для обработки различных сценариев использования.

e) Часто используется для создания профессиональных и гибких CLI-интерфейсов.

Вывод: в ходе выполнения лабораторной работы, приобретены навыки построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.