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).

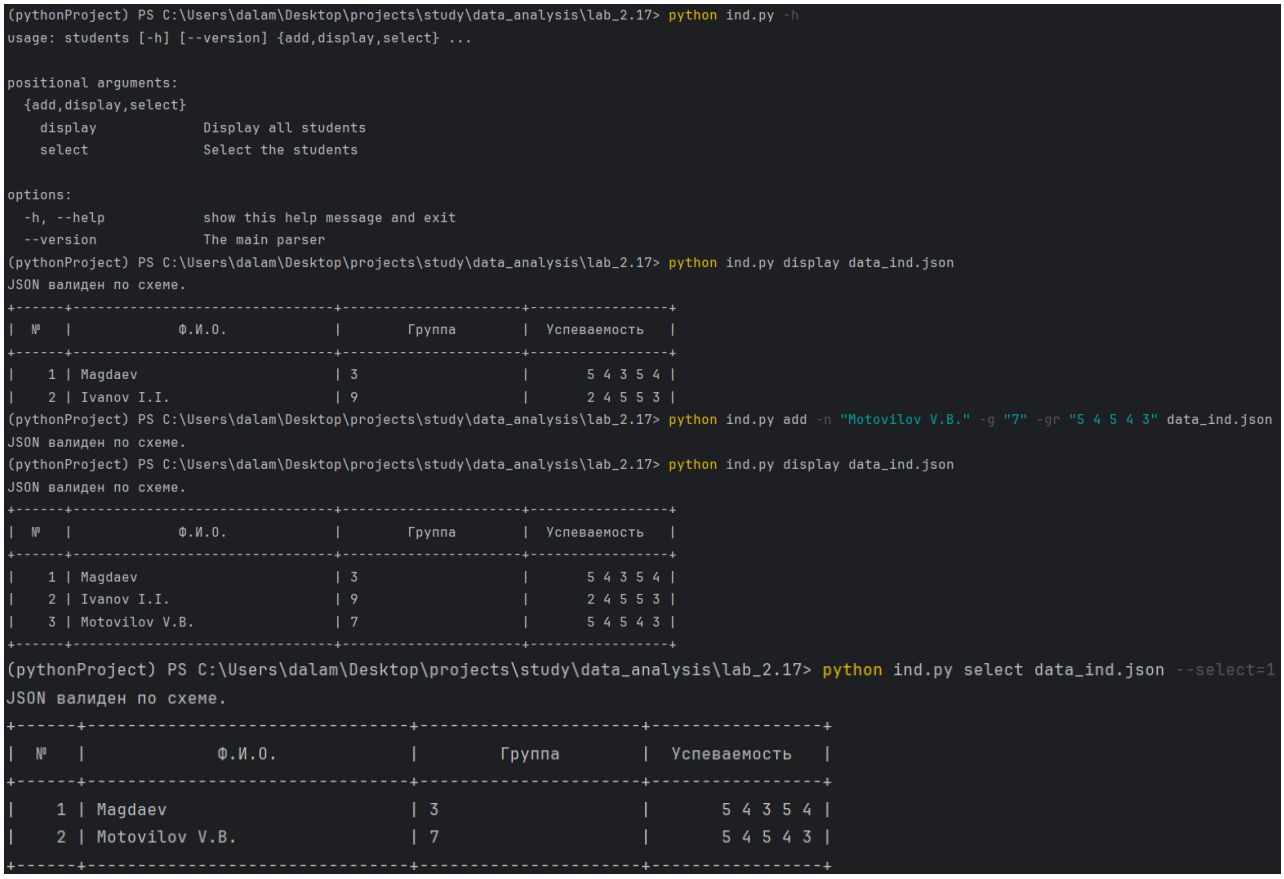

Рисунок 3. Страницы руководства и результат работы программы

Код индивидуального задания №1:

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

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


def add_student(students, name, group, grade):
    """
    Добавить данные о студенте
    """
    students.append(
        {
            'name': name,
            'group': group,
            'grade': grade,
        }
    )
    return students


def show_list(students):
```

```
    """
    Вывести список студентов
    """
    # Заголовок таблицы.
    if students:

        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 15
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Успеваемость"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(students, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group', ''),
                    student.get('grade', 0)
                )
            )
        print(line)
    else:
        print("Список студентов пуст.")


def show_selected(students):
    # Проверить сведения студентов из списка.
    result = []
    for student in students:
        grade = [int(x) for x in (student.get('grade', '').split())]
        if sum(grade) / max(len(grade), 1) >= 4.0:
            result.append(student)
    return result


def help_1():
    print("Список команд:\n")
    print("add - добавить студента;")
    print("display - вывести список студентов;")
    print("select - запросить студентов с баллом выше 4.0;")
    print("save - сохранить список студентов;")
    print("load - загрузить список студентов;")
    print("exit - завершить работу с программой.")


def save_students(file_name, students):
    with open(file_name, "w", encoding="utf-8") as fout:
```

```
        json.dump(students, fout, ensure_ascii=False, indent=4)


def load_students(file_name):
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "name": {"type": "string"},
                "group": {"type": "integer"},
                "grade": {"type": "string"},
            },
            "required": [
                "name",
                "group",
                "grade",
            ],
        },
    }
    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("students")
    parser.add_argument(
        "--version",
        action="version",
        help="The main parser",
        version="%(prog)s 0.1.0"
    )

    subparsers = parser.add_subparsers(dest="command")

    # Создать субпарсер для добавления студента.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new student"
    )
    add.add_argument(
        "-n",
```

```
    "--name",
    action="store",
    required=True,
    help="The student's name"
)
add.add_argument(
    "-g",
    "--group",
    type=int,
    action="store",
    help="The student's group"
)
add.add_argument(
    "-gr",
    "--grade",
    action="store",
    required=True,
    help="The student's grade"
)

# Создать субпарсер для отображения всех студентов.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all students"
)

# Создать субпарсер для выбора студентов.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the students"
)
select.add_argument(
    "-s",
    "--select",
    action="store",
    required=True,
    help="The required select"
)

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

# Загрузить всех студентов из файла, если файл существует.
is_dirty = False
if os.path.exists(args.filename):
    students = load_students(args.filename)
else:
    students = []

# Добавить студента.
if args.command == "add":
    students = add_student(
        students,
        args.name,
        args.group,
        args.grade
    )
    is_dirty = True
```

```
    # Отобразить всех студентов.
    elif args.command == "display":
        show_list(students)

    # Выбрать требуемых студентов.
    elif args.command == "select":
        selected = show_selected(students)
        show_list(selected)

    # Сохранить данные в файл, если список студентов был изменен.
    if is_dirty:
        save_students(args.filename, students)


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.

```
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind_hard.py --help
Usage: ind_hard.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.
Commands:
  add      Добавить данные о студенте
  display  Отобразить список студентов
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind_hard.py display data_hard.json
+------+--------------------------------+----------------------+-----------------+
|  №   |            Ф.И.О.              |        Группа        |  Успеваемость   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Ivanov I.I.                    | 4                    |       3 3 4 3 4 |
+------+--------------------------------+----------------------+-----------------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind_hard.py add -n "Magdaev D.M." -g 5 -gr "3 3 4 3 4" data_hard.json
Студент добавлен
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind_hard.py add -n "Motovilov V.B." -g 7 -gr "5 5 4 3 4" data_hard.json
Студент добавлен
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind_hard.py display data_hard.json
+------+--------------------------------+----------------------+-----------------+
|  №   |            Ф.И.О.              |        Группа        |  Успеваемость   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Ivanov I.I.                    | 4                    |       3 3 4 3 4 |
|    2 | Magdaev D.M.                   | 5                    |       3 3 4 3 4 |
|    3 | Motovilov V.B.                 | 7                    |       5 5 4 3 4 |
+------+--------------------------------+----------------------+-----------------+
(pythonProject) PS C:\Users\dalam\Desktop\projects\study\data_analysis\lab_2.17> python ind_hard.py display data_hard.json -s 1
+------+--------------------------------+----------------------+-----------------+
|  №   |            Ф.И.О.              |        Группа        |  Успеваемость   |
+------+--------------------------------+----------------------+-----------------+
|    1 | Motovilov V.B.                 | 7                    |       5 5 4 3 4 |
+------+--------------------------------+----------------------+-----------------+
```

Рисунок 4. Страницы руководства и результат работы программы

Код индивидуального задания №2:

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


import json
import click
```

```
@click.group()
def cli():
    pass


@cli.command("add")
@click.argument('filename')
@click.option("-n", "--name")
@click.option("-g", "--group")
@click.option("-gr", "--grade")
def add(filename, name, group, grade):
    """
    Добавить данные о студенте
    """
    # Запросить данные о студенте.
    students = load_students(filename)
    students.append(
        {
            'name': name,
            'group': group,
            'grade': grade,
        }
    )
    with open(filename, "w", encoding="utf-8") as fout:
        json.dump(students, fout, ensure_ascii=False, indent=4)
    click.secho("Студент добавлен")


@cli.command("display")
@click.argument('filename')
@click.option('--select', '-s', type=int)
def display(filename, select=None):
    """
    Отобразить список студентов
    """
    students = load_students(filename)
    if select == 1:
        students = selected(students)

    line = '+-{}-+-{}-+-{}-+-{}-+'.format(
        '-' * 4,
        '-' * 30,
        '-' * 20,
        '-' * 15
    )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
            "№",
            "Ф.И.О.",
            "Группа",
            "Успеваемость"
        )
    )
    print(line)

    # Вывести данные о всех студентах.
    for idx, student in enumerate(students, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>15} |'.format(
```

```
                idx,
                student.get('name', ''),
                student.get('group', ''),
                student.get('grade', 0)
            )
        )
    print(line)


def selected(list):
    # Проверить сведения студентов из списка.
    students = []
    for student in list:
        result = [int(x) for x in (student.get('grade', '').split())]
        if sum(result) / max(len(result), 1) >= 4.0:
            students.append(student)
    return students


def load_students(filename):
    with open(filename, "r", encoding="utf-8") as fin:
        return json.load(fin)


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

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.