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 №3
дисциплины «Анализ данных»

Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент, 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 построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.

Ход работы

Пример 1. *Для примера 1 лабораторной работы 2.16 разработайте интерфейс командной строки.*

Код программы:

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

import argparse
import json
import os.path
import sys
from datetime import date


def add_worker(staff, name, post, year):
    """
    Добавить данные о работнике.
    """
    staff.append(
        {
            "name": name,
            "post": post,
            "year": year
        }
    )
    return staff


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 кирилицы 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(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("workers")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )

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

    # Создать субпарсер для добавления работника.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new worker"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="The worker's name"
    )
    add.add_argument(
        "-p",
        "--post",
        action="store",
        help="The worker's post"
    )
    add.add_argument(
        "-y",
        "--year",
        action="store",
        type=int,
        required=True,
```

```
        help="The year of hiring"
    )

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

    # Создать субпарсер для выбора работников.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the workers"
    )
    select.add_argument(
        "-P",
        "--period",
        action="store",
        type=int,
        required=True,
        help="The required period"
    )

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

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

    # Добавить работника.
    if args.command == "add":
        workers = add_worker(
            workers,
            args.name,
            args.post,
            args.year
        )
        is_dirty = True

    # Отобразить всех работников.
    elif args.command == "display":
        display_workers(workers)

    # Выбрать требуемых рааботников.
    elif args.command == "select":
        selected = select_workers(workers, args.period)
        display_workers(selected)
```

```
    # Сохранить данные в файл, если список работников был изменен.
    if is_dirty:
        save_workers(args.filename, workers)


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

```
(venv) PS C:\Rep\Data_analysis_3\Project> python example.py add data.json --name="Сидоров Сидор" --post="главный инженер" --year=2012
(venv) PS C:\Rep\Data_analysis_3\Project> python example.py add data.json --name="пётр петров" --post="бухгалтер" --year=2010
(venv) PS C:\Rep\Data_analysis_3\Project> python example.py add data.json --name="иванов иван" --post="директор" --year=2007
```

Рисунок 1. Добавление новых работников

```
(venv) PS C:\Rep\Data_analysis_3\Project> python example.py display data.json
+------+--------------------------------+----------------------+----------+
+------+--------------------------------+----------------------+----------+
|    1 | Сидоров Сидор                  | главный инженер      |     2012 |
+------+--------------------------------+----------------------+----------+
|    2 | пётр петров                    | бухгалтер            |     2010 |
+------+--------------------------------+----------------------+----------+
|    3 | иванов иван                    | директор             |     2007 |
+------+--------------------------------+----------------------+----------+
```

Рисунок 2. Отображение информации обо всех работниках

```
(venv) PS C:\Rep\Data_analysis_3\Project> python example.py select data.json --period=12
+------+--------------------------------+----------------------+----------+
|  №   |            Ф.И.О.              |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Сидоров Сидор                  | главный инженер      |     2012 |
+------+--------------------------------+----------------------+----------+
```

Рисунок 3. Выбор работников с заданном периодом работы

Индивидуальные задания:

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

2. Использовать словарь, содержащий следующие ключи: фамилия и инициалы; номер группы; успеваемость (список из пяти элементов). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию среднего балла; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5; если таких студентов нет, вывести соответствующее сообщение.

Код программы:

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

```python
import argparse
import json
import os.path


def add_student(staff, surname, group_number, grades):
    """
    Добавить данные о студенте.
    """
    staff.append(
        {
            "surname": surname,
            "group_number": group_number,
            "grades": grades
        }
    )

    return staff


def display_students(staff):
    """
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Оценки"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                    idx,
                    student.get('surname', ''),
                    student.get('group_number', ''),
                    ', '.join(str(el) for el in student.get('grades')[0])
                )
```

```
            )
        print(line)

    else:
        print("Список студентов пуст.")


def select_students(staff):
    # Сформировать список студентов, имеющих оценки 4 и 5.
    result = []
    for student in staff:
        if all(int(grade) >= 4 for grade in student['grades'][0]):
            result.append(student)

    # Возвратить список выбранных студентов.
    return result


def save_students(file_name, staff):
    """
    Сохранить всех учеников 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)


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


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",
        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(
    "-sn",
    "--surname",
    action="store",
    type=str,
    required=True,
    help="The student's surname"
)
add.add_argument(
    "-gn",
    "--group_number",
    action="store",
    type=int,
    help="The student's group"
)
add.add_argument(
    "-g",
    "--grades",
    action="store",
    nargs='+',
    type=list,
    required=True,
    help="grades"
)

# 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"
)

# 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.surname,
            args.group_number,
            args.grades
        )
        is_dirty = True

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

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

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


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

```
(venv) PS C:\Rep\Data_analysis_3\Project> python indv1.py add data2.json --surname="михайлова с.а" -gn=2 --grades="33434"
(venv) PS C:\Rep\Data_analysis_3\Project> python indv1.py add data2.json --surname="третьяков к.б" -gn=3 --grades="44454"
(venv) PS C:\Rep\Data_analysis_3\Project> python indv1.py add data2.json --surname="журавлёва а.в" -gn=1 --grades="55545"
```

Рисунок 4. Добавление новых студентов

```
(venv) PS C:\Rep\Data_analysis_3\Project> python indv1.py display data2.json
+------+-----------------------------+---------------------+----------------+
|  №   |           Ф.И.О.            |       Группа        |     Оценки     |
+------+-----------------------------+---------------------+----------------+
|    1 | михайлова с.а               | 2                   |  3, 3, 4, 3, 4 |
|    2 | третьяков к.б               | 3                   |  4, 4, 4, 5, 4 |
|    3 | журавлёва а.в               | 1                   |  5, 5, 5, 4, 5 |
+------+-----------------------------+---------------------+----------------+
```

Рисунок 5. Отображение информации обо всех студентах

```
(venv) PS C:\Rep\Data_analysis_3\Project> python indv1.py select data2.json
+------+--------------------------------+----------------------+----------------+
|  №   |            Ф.И.О.              |        Группа        |     Оценки     |
+------+--------------------------------+----------------------+----------------+
|    1 | третьяков к.б                  | 3                    | 4, 4, 4, 5, 4  |
|    2 | журавлёва а.в                  | 1                    | 5, 5, 5, 4, 5  |
+------+--------------------------------+----------------------+----------------+
```

Рисунок 6. Выбор студентов, имеющих оценки 4 и 5

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`.

Код программы:

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

import json
import os.path
import click


def add_student(staff, surname, group_number, grades):
    """
    Добавить данные о студенте.
    """
    staff.append(
        {
            "surname": surname,
            "group_number": group_number,
            "grades": grades
        }
    )

    return staff


def display_students(staff):
    """
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
```

```
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Оценки"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                    idx,
                    student.get('surname', ''),
                    student.get('group_number', ''),
                    ', '.join(str(el) for el in student.get('grades')[0])
                )
            )
        print(line)

    else:
        print("Список студентов пуст.")


def select_students(staff):
    # Сформировать список студентов, имеющих оценки 4 и 5.
    result = []
    for student in staff:
        if all(int(grade) >= 4 for grade in student['grades'][0]):
            result.append(student)

    # Возвратить список выбранных студентов.
    return result


def save_students(file_name, staff):
    """
    Сохранить всех учеников 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)


def load_students(file_name):
    """
    Загрузить всех учеников из файла JSON.
    """
```

```
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


@click.command()
@click.argument("filename")
@click.option("--surname", "-sn", type=str, help="The student's surname")
@click.option("--group_number", "-gn", type=int, help="The student's group")
@click.option("--grades", "-g", multiple=True, type=list, help="grades")
@click.argument('command', type=click.Choice(['add', 'display', 'select']))
def main(filename, surname, group_number, grades, command):
    is_dirty = False

    students = load_students(filename) if os.path.exists(filename) else []

    if command == "add":
        add_student(students, surname, group_number, grades)
        is_dirty = True
    elif command == "display":
        display_students(students)
    elif command == "select":
        selected = select_students(students)
        display_students(selected)

    if is_dirty:
        save_students(filename, students)


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

```
(venv) PS C:\Rep\Data_analysis_3\Project> python indv2.py --surname="третьяков к.б" -gn=3 --grades="44454" data3.json add
(venv) PS C:\Rep\Data_analysis_3\Project> python indv2.py --surname="михайлова с.а" -gn=2 --grades="33434" data3.json add
(venv) PS C:\Rep\Data_analysis_3\Project> python indv2.py data3.json display
+-----+------------------------------+--------------------+---------------+
| №   |           Ф.И.О.             |       Группа       |    Оценки     |
+-----+------------------------------+--------------------+---------------+
|   1 | журавлёва а.в                | 1                  | 5, 5, 5, 4, 5 |
|   2 | третьяков к.б                | 3                  | 4, 4, 4, 5, 4 |
|   3 | михайлова с.а                | 2                  | 3, 3, 4, 3, 4 |
+-----+------------------------------+--------------------+---------------+
```

Рисунок 7. Добавление новых студентов и вывод информации

```
(venv) PS C:\Rep\Data_analysis_3\Project> python indv2.py data3.json select
+-----+------------------------------+--------------------+---------------+
| №   |           Ф.И.О.             |       Группа       |    Оценки     |
+-----+------------------------------+--------------------+---------------+
|   1 | журавлёва а.в                | 1                  | 5, 5, 5, 4, 5 |
|   2 | третьяков к.б                | 3                  | 4, 4, 4, 5, 4 |
+-----+------------------------------+--------------------+---------------+
```

Рисунок 8. Выбор студентов, имеющих оценки 4 и 5

Контрольные вопросы:

1. Терминал - это программное обеспечение, которое позволяет пользователю взаимодействовать с операционной системой через текстовый интерфейс. Терминал обычно предоставляет доступ к командной строке.

Консоль - это окно или экран, где отображается интерфейс командной строки. Оно предоставляет пользователю возможность вводить команды и видеть вывод программ.

2. Консольное приложение - это приложение, которое работает в текстовом режиме, без графического интерфейса. Пользователь взаимодействует с таким приложением через командную строку.

3. Средства Python для построения приложений командной строки:

- sys.argv - для работы с аргументами командной строки.
- Модуль getopt - для парсинга аргументов командной строки.
- Модуль argparse - для создания гибких и удобных интерфейсов командной строки.

4. Особенности построения CLI с использованием модуля sys:

- sys.argv содержит список аргументов командной строки, переданных скрипту.
- При использовании sys.argv, разработчику нужно самостоятельно обрабатывать и проверять переданные аргументы.

5. Особенности построения CLI с использованием модуля getopt:

- Модуль getopt позволяет более гибко и структурированно обрабатывать аргументы командной строки.
- С помощью getopt можно определять опции с короткими и длинными флагами.

6. Особенности построения CLI с использованием модуля argparse:

- argparse предоставляет более высокоуровневый интерфейс для создания CLI приложений.
- С помощью argparse можно определять аргументы, опции, поддерживать автогенерацию справки и другие удобные функции.

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