МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра

инфокоммуникаций

Институт цифрового

развития

ОТЧЁТ

по лабораторной работе №2.17

Дисциплина: «Основы программной инженерии»

Тема: «Разработка

приложений с интерфейсом командной

строки (CLI) в Python3»

Выполнил: студент

2 курса группы Пиж-

б-о-21-1

Рязанцев Матвей

Денисович

Ставрополь 2023

Цель работы: приобретение построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.

Ход работы

```
(lab217) D:\2kurs\!22kurs\opi_lab217>python main.py display dattta.json
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Mat                              | gen dir              |     2020 |
|    2 | Yan                              | Gen gen dir          |     2011 |
+------+----------------------------------+----------------------+----------+
```

Рисунок 1 – результата работы программы пример 1

```
D:\2kurs\!22kurs\opi_lab217>python main.py add dattta.json --name="Kai" --post="slu" --year=2009

D:\2kurs\!22kurs\opi_lab217>python main.py display dattta.json
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Mat                              | gen dir              |     2020 |
|    2 | Yan                              | Gen gen dir          |     2011 |
|    3 | Kai                              | slu                  |     2009 |
+------+----------------------------------+----------------------+----------+
```

Рисунок 2 – добавление нового служащего

Индивидуальное задание 1

Код программы

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

import json
import argparse
import os.path


def ad(pep,name,num,year):
    pep.append(
        {
            'name': name,
            'num': num,
            'year': year
        }
    )
    return pep

def li(pep):
     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
```

```
                '-' * 20,
                '-' * 8
            )
    print(line)
    print(
        '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
            "№",
            "F.I.O.",
            "NUMBER",
            "BRDAY"
        )
    )
    print(line)
    for idx, chel in enumerate(pep, 1):
        print(
            '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                idx,
                chel.get('name', ''),
                chel.get('num', ''),
                chel.get('year', 0)
            )
        )
        print(line)


def sel(pep, numb):
    ot = []
    # Проверить сведения работников из списка.
    for chel in pep:
        if numb in str(chel.values()):
            ot.append(chel)
    return ot


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", errors="ignore") 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", errors="ignore") 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("pep")
    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 человека.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new human"
    )
    add.add_argument(
        "-na",
        "--name",
        action="store",
        required=True,
        help="The human's name"
    )
    add.add_argument(
        "-n",
        "--num",
        action="store",
        type=int,
        required=True,
        help="The human's number"
    )
    add.add_argument(
        "-y",
        "--year",
        action="store",
        type=int,
        required=True,
        help="The date of human's birth"
    )

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

    # Создать субпарсер для выбора людей.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the humans"
    )
    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 людей из файла, если файл существует.
    is_dirty = False
    if os.path.exists(args.filename):
        pep = load_workers(args.filename)
    else:
        pep = []

    # Добавить человека.
    if args.command == "add":
        pep = ad(
            pep,
            args.name,
            args.num,
            args.year
        )
        is_dirty = True


    # Отобразить всех людей.
    elif args.command == "display":
        li(pep)

    # Выбрать требуемых людей.
    elif args.command == "select":
        selected = sel(pep, args.select)
        li(selected)

    # Сохранить данные в файл, если список людей был изменен.
    if is_dirty:
        save_workers(args.filename, pep)


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

```
(myenv) D:\my_downloads>python test.py select dat.json -s 43
+------+--------------------------------+--------------------+--------------------+
|  №   |             F.I.O.             |       NUMBER       |  BRDAY   |
+------+--------------------------------+--------------------+--------------------+
|    1 | mat                            | 13431              |     2003 |
+------+--------------------------------+--------------------+--------------------+
|    2 | adfd                           | 43                 |     1220 |
+------+--------------------------------+--------------------+--------------------+

(myenv) D:\my_downloads>
```

```
(myenv) D:\my_downloads>python test.py display dat.json
+------+--------------------------------+----------------------+--------------------+
|  №   |             F.I.O.             |        NUMBER        |  BRDAY   |
+------+--------------------------------+----------------------+--------------------+
|    1 | mat                            | 31531                |        0 |
+------+--------------------------------+----------------------+--------------------+
|    2 | mat                            | 13431                |     2003 |
+------+--------------------------------+----------------------+--------------------+

(myenv) D:\my_downloads>python test.py add dat.json -nm "adfd" -n 43 -y 1220

(myenv) D:\my_downloads>python test.py display dat.json
+------+--------------------------------+----------------------+--------------------+
|  №   |             F.I.O.             |        NUMBER        |  BRDAY   |
+------+--------------------------------+----------------------+--------------------+
|    1 | mat                            | 31531                |        0 |
+------+--------------------------------+----------------------+--------------------+
|    2 | mat                            | 13431                |     2003 |
+------+--------------------------------+----------------------+--------------------+
|    3 | adfd                           | 43                   |     1220 |
+------+--------------------------------+----------------------+--------------------+
```

Рисунок 1 – результат работы программы

Индивидуальное задание 2

Код программы

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

import json
import os.path
import click


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


@cli.command(help="Add a new work")
@click.option("-nm", "--name", required=True, help="Name worker's")
@click.option("-n", "--num", type=int, required=True, help="Number of the
worker")
@click.option("-y", "--year", required=True, help="burthday worker")
@click.argument("filename")
def add(name, num, year, filename):
    pep = load_workers(filename)
    pep = add_wrk(pep, name, num, year)
    save_workers(filename, pep)


@cli.command(help="Display all workers")
@click.argument("filename")
def display(filename):
    pep = load_workers(filename)
    li(pep)


@cli.command(help="Select the worker")
@click.option("-s", "--select", required=True, help="The required select")
@click.argument("filename")
```

```
def select(select, filename):
    pep = load_workers(filename)
    select = sel(pep, select)
    li(select)


def add_wrk(pep, name, num, year):
    pep.append(
        {
            'name': name,
            'num': num,
            'year': year
        }
    )
    return pep


def li(pep):
     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
     print(line)
     print(
          '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
              "№",
              "F.I.O.",
              "NUMBER",
              "BRDAY"
          )
     )
     print(line)
     for idx, chel in enumerate(pep, 1):
        print(
             '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                 idx,
                 chel.get('name', ''),
                 chel.get('num', ''),
                 chel.get('year', 0)
             )
        )
        print(line)


def sel(pep, numb):
    ot = []
    # Проверить сведения работников из списка.
    for chel in pep:
        if numb in str(chel.values()):
            ot.append(chel)
    return ot


def save_workers(file_name, staff):
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4)


def load_workers(file_name):
    if os.path.exists(file_name):
        with open(file_name, "r", encoding="utf-8", errors="ignore") as fin:
            return json.load(fin)
```

```
    else:
        return []


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

```
D:\2kurs\!22kurs\opi_lab217>python test.py display dat.json
+------+--------------------------------+----------------------+--------------------+
|  №   |             F.I.O.             |        NUMBER        |  BRDAY   |
+------+--------------------------------+----------------------+--------------------+
|    1 | mat                            | 31531                |        0 |
+------+--------------------------------+----------------------+--------------------+
|    2 | mat                            | 13431                |     2003 |
+------+--------------------------------+----------------------+--------------------+
|    3 | li                             | 153                  |     2003 |
+------+--------------------------------+----------------------+--------------------+

D:\2kurs\!22kurs\opi_lab217>python idz2.py add dat.json -nm "Jarg" -n 89414 -y 2000

D:\2kurs\!22kurs\opi_lab217>python idz2.py display dat.json
+------+--------------------------------+----------------------+----------+
|  №   |             F.I.O.             |        NUMBER        |  BRDAY   |
+------+--------------------------------+----------------------+----------+
|    1 | mat                            | 31531                |        0 |
+------+--------------------------------+----------------------+----------+
|    2 | mat                            | 13431                |     2003 |
+------+--------------------------------+----------------------+----------+
|    3 | li                             | 153                  |     2003 |
+------+--------------------------------+----------------------+----------+
|    4 | Jarg                           | 89414                |     2000 |
+------+--------------------------------+----------------------+----------+

D:\2kurs\!22kurs\opi_lab217>python idz2.py select dat.json -s 89414
+------+--------------------------------+----------------------+----------+
|  №   |             F.I.O.             |        NUMBER        |  BRDAY   |
+------+--------------------------------+----------------------+----------+
|    1 | Jarg                           | 89414                |     2000 |
+------+--------------------------------+----------------------+----------+

D:\2kurs\!22kurs\opi_lab217>
```

Рисунок 2 – результат работы программы

Контрольные вопросы

1. В чем отличие терминала и консоли?

Терминал (от лат. terminus — граница) — устройство или ПО, выступающее посредником между человеком и вычислительной системой.

Обычно данный термин используется, когда точка доступа к системе вынесена в отдельное физическое устройство и предоставляет свой пользовательский интерфейс на основе внутреннего интерфейса (например, сетевых протоколов).

Консоль console — исторически реализация терминала с клавиатурой и текстовым дисплеем. В настоящее время это слово часто используется как синоним сеанса работы или окна оболочки командной строки. В том же смысле иногда применяется и слово “терминал”.

2. Что такое консольное приложение?

Консольное приложение console application — вид ПО, разработанный с расчётом на работу внутри оболочки командной строки, т.е. опирающийся на текстовый ввод-вывод.

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

Python 3 поддерживает несколько различных способов обработки аргументов командной строки. Встроенный способ – использовать модуль sys. С точки зрения имен и использования, он имеет прямое отношение к библиотеке C (libc). Второй способ – это модуль getopt, который обрабатывает как короткие, так и длинные параметры, включая оценку значений параметров.

4. Какие особенности построение CLI с использованием модуля sys?

Это базовый модуль, который с самого начала поставлялся с Python. Он использует подход, очень похожий на библиотеку C, с использованием argc и argv для доступа к аргументам. Модуль sys реализует аргументы командной строки в простой структуре списка с именем sys.argv

5. Какие особенности построение CLI с использованием модуля getopt?

Как вы могли заметить ранее, модуль sys разбивает строку командной строки только на отдельные фасеты. Модуль getopt в Python идет немного дальше и расширяет разделение входной строки проверкой параметров. Основанный на функции C getopt, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений.

6. Какие особенности построение CLI с использованием модуля argparse?

Начиная с версий Python 2.7 и Python 3.2, в набор стандартных библиотек была включена библиотека argparse для обработки аргументов (параметров, ключей) командной строки.

Для начала рассмотрим, что интересного предлагает argparse:

* анализ аргументов sys.argv;
* конвертирование строковых аргументов в объекты вашей программы и работа с ними;

* форматирование и вывод информативных подсказок