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»

Выполнила:

студентка 2 курса

группы Пиж-б-о-21-1

Джолдошова Мээрим

Бекболотовна

Ставрополь 2023

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

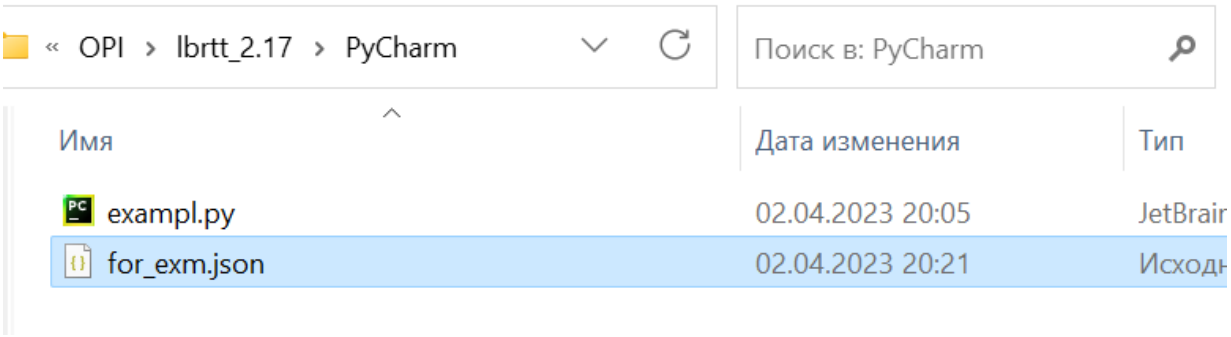

Рисунок 1 – Был проработан пример

```
(myenv) C:\OPI\lbrtt_2.17\PyCharm>python exampl.py display for_exm.json
+------+--------------------------------+----------------------+----------+
|  No  |             Ф.И.О.             |      Должность       |   Год    |
+------+--------------------------------+----------------------+----------+
|    1 | Mike                           | ceo                  |     2003 |
|    2 | Mia                            | helper               |     2005 |
+------+--------------------------------+----------------------+----------+
```

Рисунок 2 – Результат работы программы

Индивидуальное задание

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

Код программы:

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

import json
import argparse
import os.path


def add_flight(flights, dst, nmb, tpe):
    flights.append(
        {
            "destination": dst,
            "number_flight": nmb,
            "type_plane": tpe
        }
    )
    return flights


def display_flights(flights):
    """
```

```
    displaying the given information
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 18
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^18} |'.format(
                "№",
                "Destination",
                "NumberOfTheFlight",
                "TypeOfThePlane"
            )
        )
        print(line)

        for idx, flight in enumerate(flights, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>18} |'.format(
                    idx,
                    flight.get('destination', ''),
                    flight.get('number_flight', ''),
                    flight.get('type_plane', 0)
                )
            )
        print(line)
    else:
        print('list is empty')


def select_flights(flights, t):
    result = []
    for flight in flights:
        if t in str(flight.values()):
            result.append(flight)
    return result


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


def load_flights(file_name):
    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("flights")
    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 flight"
    )
    add.add_argument(
        "-d",
        "--destination",
        action="store",
        required=True,
        help="Destination of the flight"
    )
    add.add_argument(
        "-n",
        "--number",
        action="store",
        type=int,
        required=True,
        help="Number of the flight"
    )
    add.add_argument(
        "-t",
        "--type",
        action="store",
        required=True,
        help="Type of the plane"
    )

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

    # Создать субпарсер для выбора работников.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the flights"
    )
    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 работников из файла, если файл существует .
    is_dirty = False
    if os.path.exists(args.filename):
        flights = list(load_flights(args.filename))
```

```
    else:
        flights = []


    # Добавить работника.
    if args.command == "add":
        flights = add_flight(
            flights,
            args.destination,
            args.number_flight,
            args.type_plane
        )
        is_dirty = True

    # Отобразить всех работников.
    elif args.command == "display":
        display_flights(flights)

    # Выбрать требуемых рааботников.
    elif args.command == "select":
        selected = select_flights(flights, args.select)
        display_flights(selected)

    # Сохранить данные в файл, если список работников был изменен.
    if is_dirty:
        save_flights(args.filename, flights)


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

```
C:\OPI\lbrtt_2.17\indiv>python indiv_s.py display for_ind.json
+------+--------------------------------+----------------------+--------------------+
|  №   |          Destination           |  NumberOfTheFlight   |   TypeOfThePlane   |
+------+--------------------------------+----------------------+--------------------+
|    1 | japan                          | 2                    |         passangers |
|    2 | china                          | 7                    |         passangers |
|    3 | korea                          | 2                    |           official |
+------+--------------------------------+----------------------+--------------------+

C:\OPI\lbrtt_2.17\indiv>python indiv_s.py select for_ind.json -s passangers
+------+--------------------------------+----------------------+--------------------+
|  №   |          Destination           |  NumberOfTheFlight   |   TypeOfThePlane   |
+------+--------------------------------+----------------------+--------------------+
|    1 | japan                          | 2                    |         passangers |
|    2 | china                          | 7                    |         passangers |
+------+--------------------------------+----------------------+--------------------+
```

Рисунок 3 – Результат работы программы

```
C:\OPI\lbrtt_2.17\indiv>python indiv_s.py add for_ind.json -d "canada" -n 3 -t "official"

C:\OPI\lbrtt_2.17\indiv>python indiv_s.py display for_ind.json
+------+--------------------------------+----------------------+----------------------+
|  №   |          Destination           |  NumberOfTheFlight   |   TypeOfThePlane     |
+------+--------------------------------+----------------------+----------------------+
|    1 | japan                          | 2                    |          passangers  |
|    2 | china                          | 7                    |          passangers  |
|    3 | korea                          | 2                    |            official  |
|    4 | canada                         | 3                    |            official  |
+------+--------------------------------+----------------------+----------------------+
```

Рисунок 4 – Результат работы программы (продолжение)

Индивидуальное задание повышенной сложности

Самостоятельно изучите работу с пакетом click для построения интерфейса командной строк (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 os
import json
import click


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


@cli.command(help="Add a new flight")
@click.option("-d", "--destination", required=True, help="Destination of the
flight")
@click.option("-n", "--number", type=int, required=True, help="Number of the
flight")
@click.option("-t", "--type", required=True, help="Type of the plane")
@click.argument("filename")
def add(destination, number, type, filename):
    flights = load_flights(filename)
    flights = add_flight(flights, destination, number, type)
    save_flights(filename, flights)


@cli.command(help="Display all flights")
@click.argument("filename")
def display(filename):
    flights = load_flights(filename)
    display_flights(flights)


@cli.command(help="Select the flights")
```

```python
@click.option("-s", "--select", required=True, help="The required select")
@click.argument("filename")
def select(select, filename):
    flights = load_flights(filename)
    selected = select_flights(flights, select)
    display_flights(selected)


def add_flight(flights, dst, nmb, tpe):
    flights.append(
        {
            "destination": dst,
            "number_flight": nmb,
            "type_plane": tpe
        }
    )
    return flights


def display_flights(flights):
    """
    displaying the given information
    """
    if flights:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 18
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^18} |'.format(
                "№",
                "Destination",
                "NumberOfTheFlight",
                "TypeOfThePlane"
            )
        )
        print(line)

        for idx, flight in enumerate(flights, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>18} |'.format(
                    idx,
                    flight.get('destination', ''),
                    flight.get('number_flight', ''),
                    flight.get('type_plane', 0)
                )
            )
        print(line)
    else:
        print('list is empty')


def select_flights(flights, t):
    result = []
    for flight in flights:
        if t in str(flight.values()):
            result.append(flight)
    return result


def save_flights(file_name, flights):
```

```
    with open(file_name, "w", encoding="utf-8", errors="ignore") as fout:
        json.dump(flights, fout, ensure_ascii=False, indent=4)


def load_flights(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()
```

```
(myenv) C:\OPI\lbrtt_2.17\indiv>python indiv_advanced.py display for_ind.json
+------+----------------------------------+----------------------+--------------------+
|  №   |           Destination            |  NumberOfTheFlight   |   TypeOfThePlane   |
+------+----------------------------------+----------------------+--------------------+
|    1 | japan                            | 2                    |         passangers |
|    2 | china                            | 7                    |         passangers |
|    3 | korea                            | 2                    |           official |
|    4 | canada                           | 3                    |           official |
|    5 | canada                           | 3                    |           official |
+------+----------------------------------+----------------------+--------------------+
```

```
(myenv) C:\OPI\lbrtt_2.17\indiv>python indiv_advanced.py add for_ind.json -d "america" -n 1 -t "official"

(myenv) C:\OPI\lbrtt_2.17\indiv>python indiv_advanced.py display for_ind.json
+------+----------------------------------+----------------------+--------------------+
|  №   |           Destination            |  NumberOfTheFlight   |   TypeOfThePlane   |
+------+----------------------------------+----------------------+--------------------+
|    1 | japan                            | 2                    |         passangers |
|    2 | china                            | 7                    |         passangers |
|    3 | korea                            | 2                    |           official |
|    4 | canada                           | 3                    |           official |
|    5 | canada                           | 3                    |           official |
|    6 | america                          | 1                    |           official |
+------+----------------------------------+----------------------+--------------------+

(myenv) C:\OPI\lbrtt_2.17\indiv>python indiv_advanced.py select for_ind.json -s official
+------+----------------------------------+----------------------+--------------------+
|  №   |           Destination            |  NumberOfTheFlight   |   TypeOfThePlane   |
+------+----------------------------------+----------------------+--------------------+
|    1 | korea                            | 2                    |           official |
|    2 | canada                           | 3                    |           official |
|    3 | canada                           | 3                    |           official |
|    4 | america                          | 1                    |           official |
+------+----------------------------------+----------------------+--------------------+

(myenv) C:\OPI\lbrtt_2.17\indiv>
```

Рисунок 5 – 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