

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.17

Разработка приложений с интерфейсом командной строки (CLI) в Python3.

Выполнил студент группы

ИВТ-б-о-21-1

Мальцев Н.А. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2022

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

Проработка примера.

Код программы:

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

import argparse
import json
import os.path
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(
                "No",
                "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", errors="ignore") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы 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("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 работников.
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()

```

Результат выполнения программы:

```

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Примеры> python Example.py add data.json --name="Сидоров Сидор" --post="Главный инженер" --year=2012
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Примеры> python Example.py add data.json --name="Иванов Иван" --post="Директор" --year=2007
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Примеры> python Example.py add data.json --name="Петров Петр" --post="Бухгалтер" --year=2010
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Примеры> python Example.py display data.json

```

№	Ф.И.О.	Должность	Год
1	Сидоров Сидор	Главный инженер	2012
2	Иванов Иван	Директор	2007
3	Петров Петр	Бухгалтер	2010

```

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Примеры> python Example.py select data.json --period=12

```

№	Ф.И.О.	Должность	Год
1	Иванов Иван	Директор	2007
2	Петров Петр	Бухгалтер	2010

```

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Примеры>

```

Рисунок 1. Результат работы программы

Выполнение задания.

Код программы:

```

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

import json
import os.path
import argparse

def add_mans(list_man, name, number, date_):
    # Добавление данных
    list_man.append(
        {

```

```

        "name": name,
        "number": number,
        "date": date_

    }

)
return list_man

def list_d(list_man):
    """
    Отображение списка людей
    """
    # Проверить, что список не пуст
    if list_man:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-"
" * 20)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "No",
                "Имя",
                "Номер телефона",
                "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о человеке.
        for idx, man in enumerate(list_man, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:<20} |".format(
                    idx,
                    man.get("name", ""),
                    man.get("number", 0),
                    man.get("date", 0)
                )
            )

            print(line)
    else:
        print("Список работников пуст: ")

def select_mans(mans_list, numb):
    """
    Отбор пользователей с заданным номером телеофна
    """
    count = 0
    if mans_list:
        for man in mans_list:
            if man.get("number") == numb:
                count += 1
                print("{:>4}: {}".format(count, man.get("name", "")))
                print("Номер телефона:", man.get("number", ""))
                print("Дата рождения:", man.get("date", ""))
    else:
        print("Список пользователей пуст.")
    # Если счетчик равен 0, то человек не найден.
    if count == 0:
        print("Человек не найден.")

```

```

def save_workers(file_name_1, staff):
    """
    Сохранить всех пользователей в файл JSON.
    """
    with open(file_name_1, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4, default=str)

def load_workers(file_name_2):
    """
    Загрузить всех работников из файла JSON.
    """
    with open(file_name_2, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    # Создаем родительский парсер для определения имени файла
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The date file name"
    )
    # Основной парсер командной строки
    parser = argparse.ArgumentParser("workers")
    parser.add_argument(
        "--version",
        action="version",
        version="% (prog)s 0.1.0"
    )

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

    # Субпарсер для добавления пользователей
    add = subparser.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new worker"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="The worker name"
    )
    add.add_argument(
        "-N",
        "--number",
        action="store",
        type=str,
        help="The workers phone number"
    )
    add.add_argument(
        "-y",
        "--year",
        action="store",
        type=int,
        required=True,
        help="Man's birthdate"
    )
    # Субпарсер для отображения всех пользователей
    _ = subparser.add_parser(

```

```

        "display",
        parents=[file_parser],
        help="Display information about users"
    )
    # Субпарсер для выбора пользователей
    select = subparser.add_parser(
        "select",
        parents=[file_parser],
        help="Select users"
    )
    select.add_argument(
        "-p",
        "--phone",
        action="store",
        type=str,
        help="The required period"
    )
    # Разбор аргументов командной строки
    args = parser.parse_args(command_line)
    # Загрузить всех пользователей из файла, если он существует
    is_dirty = False
    if os.path.exists(args.filename):
        mans = load_workers(args.filename)
    else:
        mans = []
    # Добавление пользователя
    if args.command == "add":
        mans = add_mans(
            mans,
            args.name,
            args.number,
            args.year
        )
        is_dirty = True
    # Отображение всех пользователей
    elif args.command == "display":
        list_d(mans)
    # Отбор требуемых пользователей
    elif args.command == "select":
        select_mans(mans, args.phone)
    if is_dirty:
        save_workers(args.filename, mans)

if __name__ == "__main__":
    main()

```

Результат выполнения программы:

```

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Ind_task.py add data.json --name="Мальцев" --number="89620287596" --year="2003"
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Ind_task.py display data.json
-----
| No |      Имя      | Номер телефона | Дата рождения |
-----+-----+-----+-----+
| 1 | Мальцев      | 89620287596   | 2003          |
-----+-----+-----+-----+
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Ind_task.py add data.json --name="Иванов Иван" --number="123456789" --year="2005"
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Ind_task.py display data.json
-----
| No |      Имя      | Номер телефона | Дата рождения |
-----+-----+-----+-----+
| 1 | Мальцев      | 89620287596   | 2003          |
| 2 | Иванов Иван  | 123456789     | 2005          |
-----+-----+-----+-----+
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Ind_task.py select data.json --phone=89620287596
1: Мальцев
Номер телефона: 89620287596
Дата рождения: 2003
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Ind_task.py select data.json --phone=123456789
1: Иванов Иван
Номер телефона: 123456789
Дата рождения: 2005
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание>

```

Рисунок 2. Результат выполнения

Задание повышенной сложности.

Код программы:

```
#!/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("-p", "--phone")
@click.option("-d", "--date")
def add(filename, name, phone, date):
    mans = load_workers(filename)
    mans.append(
        {
            "name": name,
            "number": phone,
            "date": date,
        }
    )
    with open(filename, "w", encoding="utf-8") as fout:
        json.dump(mans, fout, ensure_ascii=False, indent=4)
    click.secho("Пользователь добавлен")

@cli.command("list")
@click.argument("filename")
def list_d(filename):
    list_man = load_workers(filename)
    if list_man:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-" * 20)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "No", "Имя", "Номер телефона", "Дата рождения"
            )
        )
        print(line)

        # Вывести данные о человеке.
        for idx, man in enumerate(list_man, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:<20} |".format(
                    idx, man.get("name", ""), man.get("number", ""),
                    man.get("date", "")
                )
            )
            print(line)
    else:
        print("Список работников пуст: ")
```

```

@cli.command("select")
@click.argument("filename")
@click.option("-s", "--select")
def select(filename, select):
    mans_list = load_workers(filename)
    count = 0
    for man in mans_list:
        if man.get("number") == select:
            count += 1
            print("{:>4}: {}".format(count, man.get("name", "")))
            print("Номер телефона:", man.get("number", ""))
            print("Дата рождения:", man.get("date", ""))

    # Если счетчик равен 0, то человек не найден.
    if count == 0:
        print("Человек не найден.")

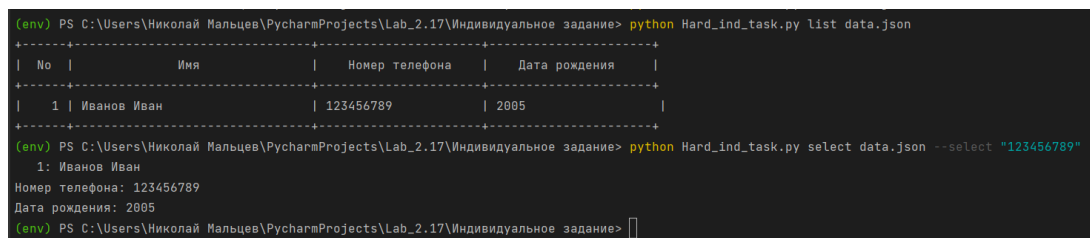
def save_workers(file_name_1, staff):
    with open(file_name_1, "w", encoding="utf-8") as fout:
        json.dump(staff, fout, ensure_ascii=False, indent=4, default=str)

def load_workers(file_name_2):
    with open(file_name_2, "r", encoding="utf-8") as fin:
        return json.load(fin)

if __name__ == "__main__":
    cli()

```

Результат выполнения программы:



```

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Hard_ind_task.py list data.json
+-----+-----+-----+-----+
| No |      Имя      | Номер телефона | Дата рождения |
+-----+-----+-----+-----+
| 1 | Иванов Иван | 123456789 | 2005 |
+-----+-----+-----+-----+

(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание> python Hard_ind_task.py select data.json --select "123456789"
1: Иванов Иван
Номер телефона: 123456789
Дата рождения: 2005
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.17\Индивидуальное задание>

```

Рисунок 3. Результат работы программы

Ответы на контрольные вопросы:

1. В чем отличие терминала от 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`). Вторым способом — это модуль `getopt`, который обрабатывает как короткие, так и длинные параметры, включая оценку значений параметров.

4. Какие особенности построения 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 построения CLI с использованием модуля `getopt`?

Как вы могли заметить ранее, модуль `sys` разбивает строку командной строки только на отдельные фасы. 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 построения 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

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