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 №2.19

Работа с файловой системой в Python3 с использованием модуля pathlib.

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 по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.

Задание 1.

Код программы:

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

import json
import os.path
import argparse
import pathlib


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)
    directory = pathlib.Path.cwd().joinpath(file_name_1)
    directory.replace(pathlib.Path.home().joinpath(file_name_1))


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:

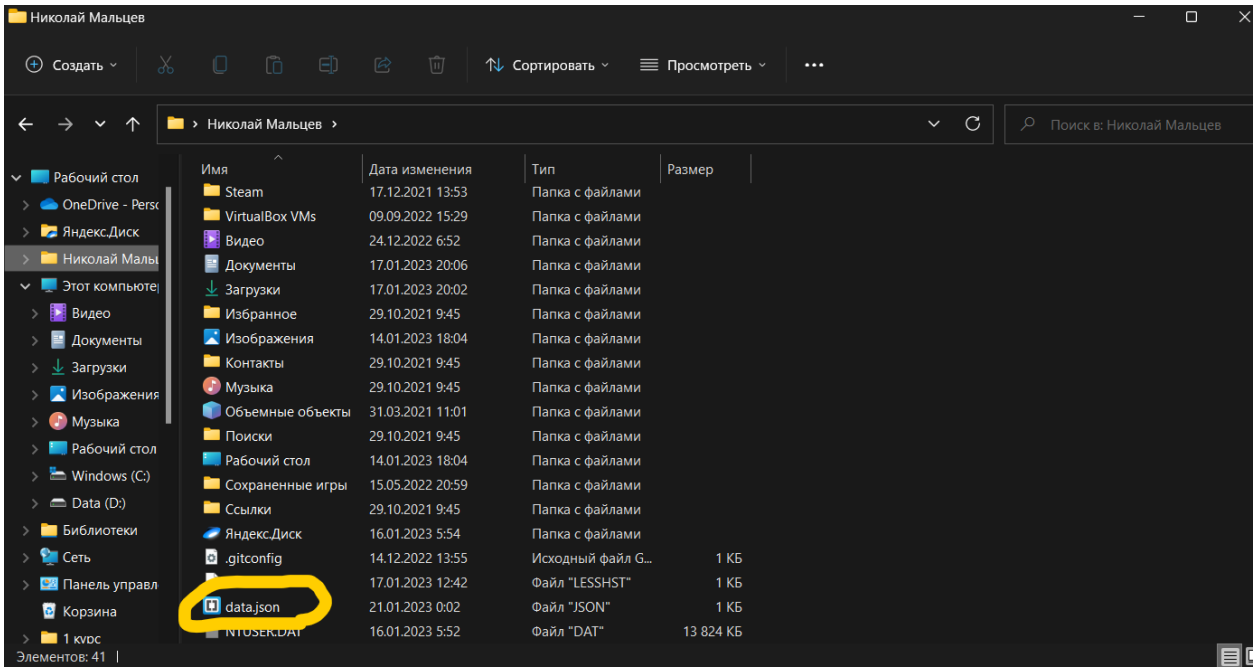

Рисунок 1. Результат выполнения

Задание 2.

Код программы:

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

```python
import argparse
import pathlib
import colorama
from colorama import Fore, Style


def tree(directory):
    print(Fore.RED + f'>>> {directory}')
    for path in sorted(directory.rglob('*')):
        depth = len(path.relative_to(directory).parts)
        spacer = ' ' * depth
        print(Fore.GREEN + Style.BRIGHT + f'{spacer} >> {path.name}')
        for new_path in sorted(directory.joinpath(path).glob('*')):
            depth = len(new_path.relative_to(directory.joinpath(path)).parts)
            spacer = '\t' * depth
            print(Fore.BLUE + f'{spacer} > {new_path.name}')


def main(command_line=None):
    colorama.init()
    current = pathlib.Path.cwd()
    file_parser = argparse.ArgumentParser(add_help=False)

    # Создаем основной парсер командной строки
    parser = argparse.ArgumentParser("tree")
    parser.add_argument(
        "--version",
        action="version",
        help="The main parser",
        version="%(prog)s 0.1.0"
    )

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

    # Создаем субпарсер для создания новой папки
    create = subparsers.add_parser(
        "mkdir",
        parents=[file_parser]
    )
    create.add_argument(
        "filename",
        action="store"
    )

    # Субпарсер для удаления папок
    create = subparsers.add_parser(
        "rmdir",
        parents=[file_parser]
    )
    create.add_argument(
        "filename",
        action="store"
    )

    # Субпарсер для создания файлов
    create = subparsers.add_parser(
        "touch",
        parents=[file_parser]
    )
    create.add_argument(
        "filename",
        action="store"
    )
    # Субпарсер для удаления файлов
```

```
    create = subparsers.add_parser(
        "rm",
        parents=[file_parser]
    )
    create.add_argument(
        "filename",
        action="store"
    )
    args = parser.parse_args(command_line)
    if args.command == 'mkdir':
        directory_path = current / args.filename
        directory_path.mkdir()
        tree(current)
    elif args.command == "rmdir":
        directory_path = current / args.filename
        directory_path.rmdir()
        tree(current)
    elif args.command == "touch":
        directory_path = current / args.filename
        directory_path.touch()
        tree(current)
    elif args.command == "rm":
        directory_path = current / args.filename
        directory_path.unlink()
        tree(current)
    else:
        tree(current)


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

Результат выполнения программы:

```
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания> python Ind_task_2.py mkdir .abc
>>> C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания
  >> .abc
  >> Ind_task_1.py
  >> Ind_task_2.py
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания> python Ind_task_2.py rmdir .abc
>>> C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания
  >> Ind_task_1.py
  >> Ind_task_2.py
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания> python Ind_task_2.py touch MyFile.txt
>>> C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания
  >> Ind_task_1.py
  >> Ind_task_2.py
  >> MyFile.txt
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания> python Ind_task_2.py rm MyFile.txt
>>> C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания
  >> Ind_task_1.py
  >> Ind_task_2.py
(env) PS C:\Users\Николай Мальцев\PycharmProjects\Lab_2.19\Индивидуальные задания>
```

Рисунок 2. Результат работы программы

Вывод: в ходе работы были приобретены навыки работы с файловой системой с использованием библиотеки языка программирования Python версии 3.x.