

**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**

Тема: «Работа с файловой системе в Python3 с использованием модуля  
pathlib»

Выполнил студент группы ИВТ-б-о-21-1

Назаров Никита Юрьевич

« » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

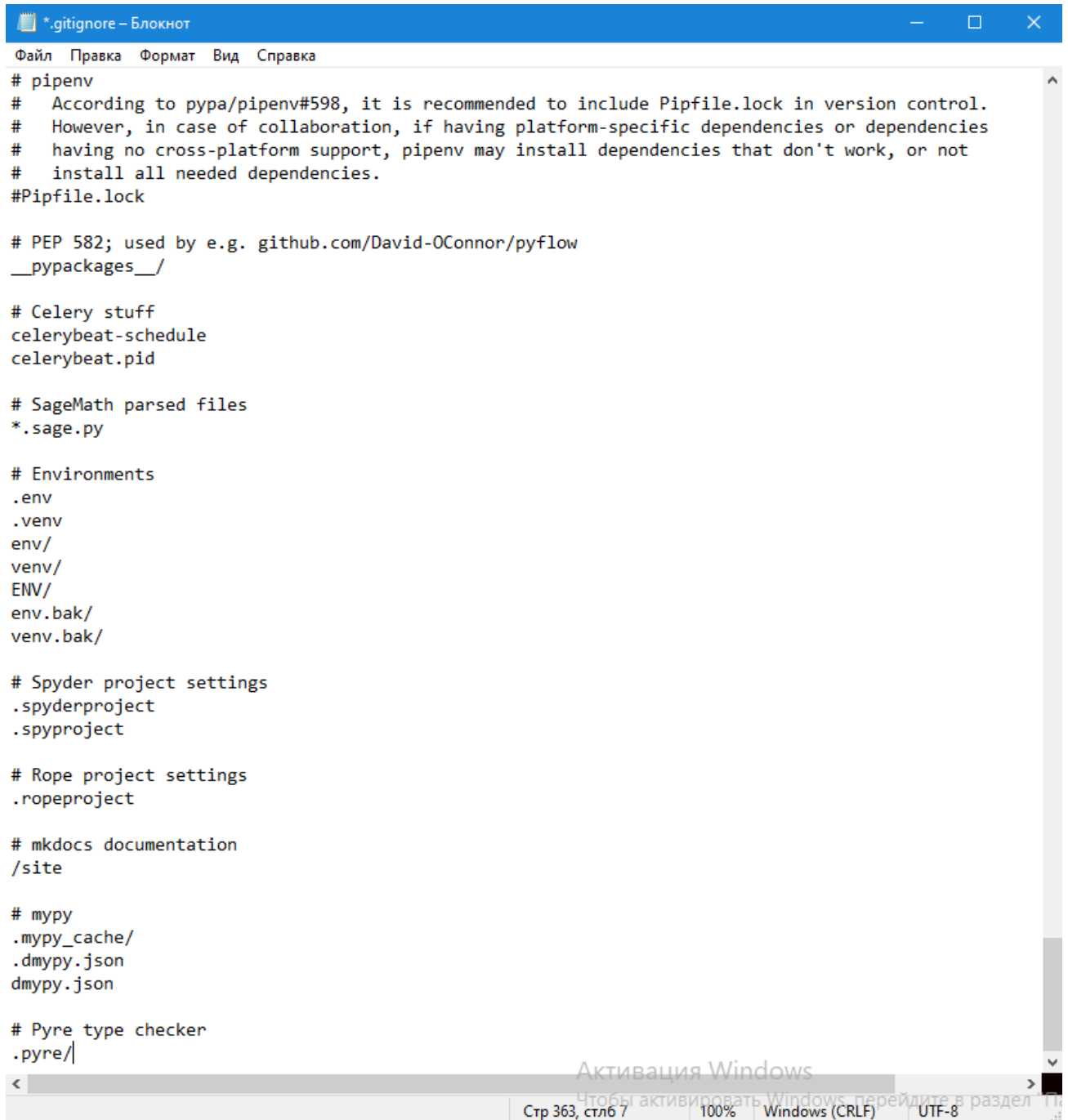
Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А. \_\_\_\_\_  
(подпись)

## Выполнение работы.



```
*.gitignore - Блокнот
Файл  Правка  Формат  Вид  Справка

# pipenv
# According to pya/pipenv#598, it is recommended to include Pipfile.lock in version control.
# However, in case of collaboration, if having platform-specific dependencies or dependencies
# having no cross-platform support, pipenv may install dependencies that don't work, or not
# install all needed dependencies.
#Pipfile.lock

# PEP 582; used by e.g. github.com/David-OConnor/pyflow
__pypackages__/

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

Стр 363, столб 7  100%  Windows (CRLF)  UTF-8
```

Рисунок 1 – редактирование gitignore

```

h:\cross\git\nazarov\2.19>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [H:/cross/git/nazarov/2.19/.git/hooks]

h:\cross\git\nazarov\2.19>

```

Рисунок 2 – организация репозитория в соответствии с gitflow

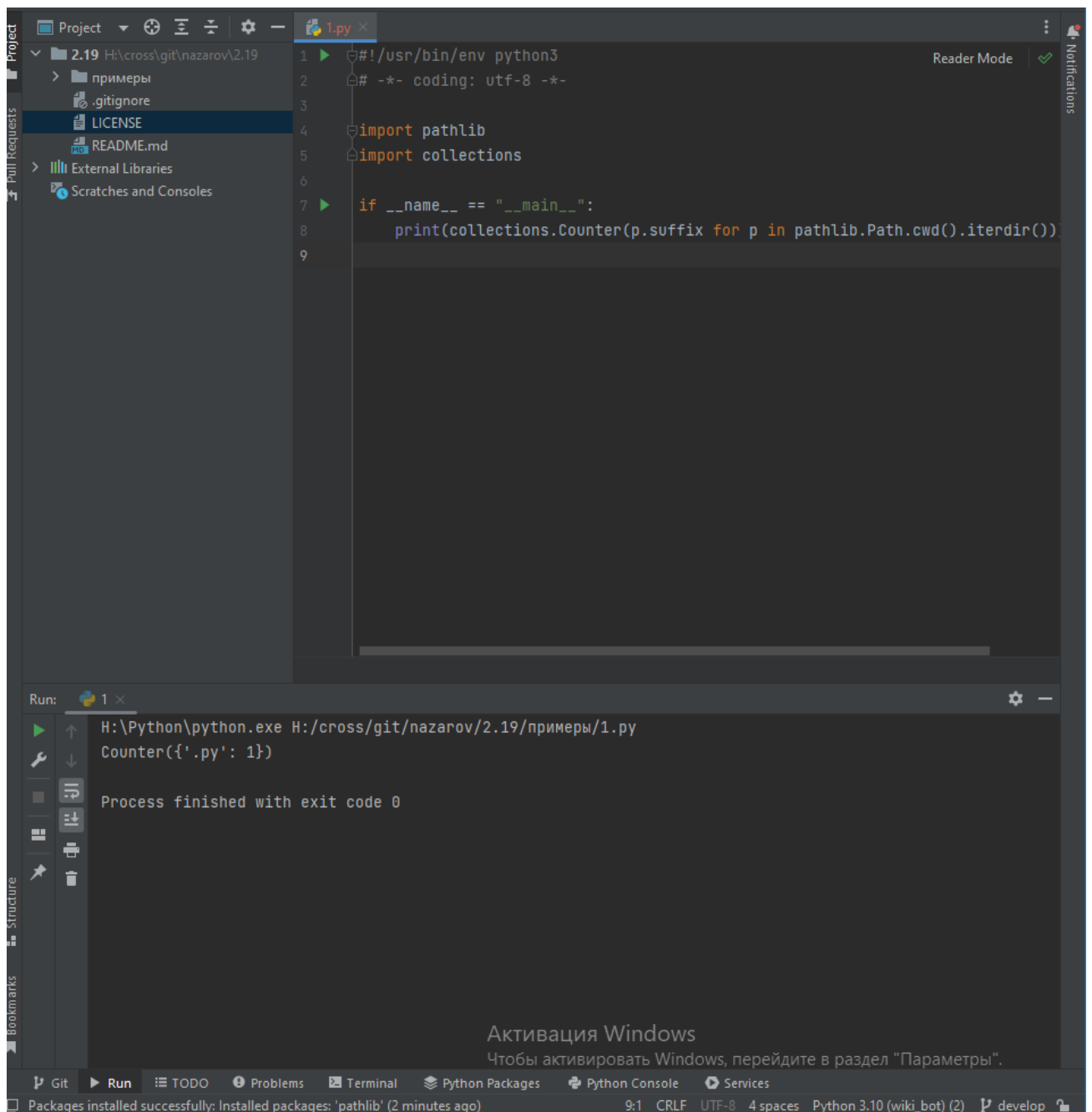
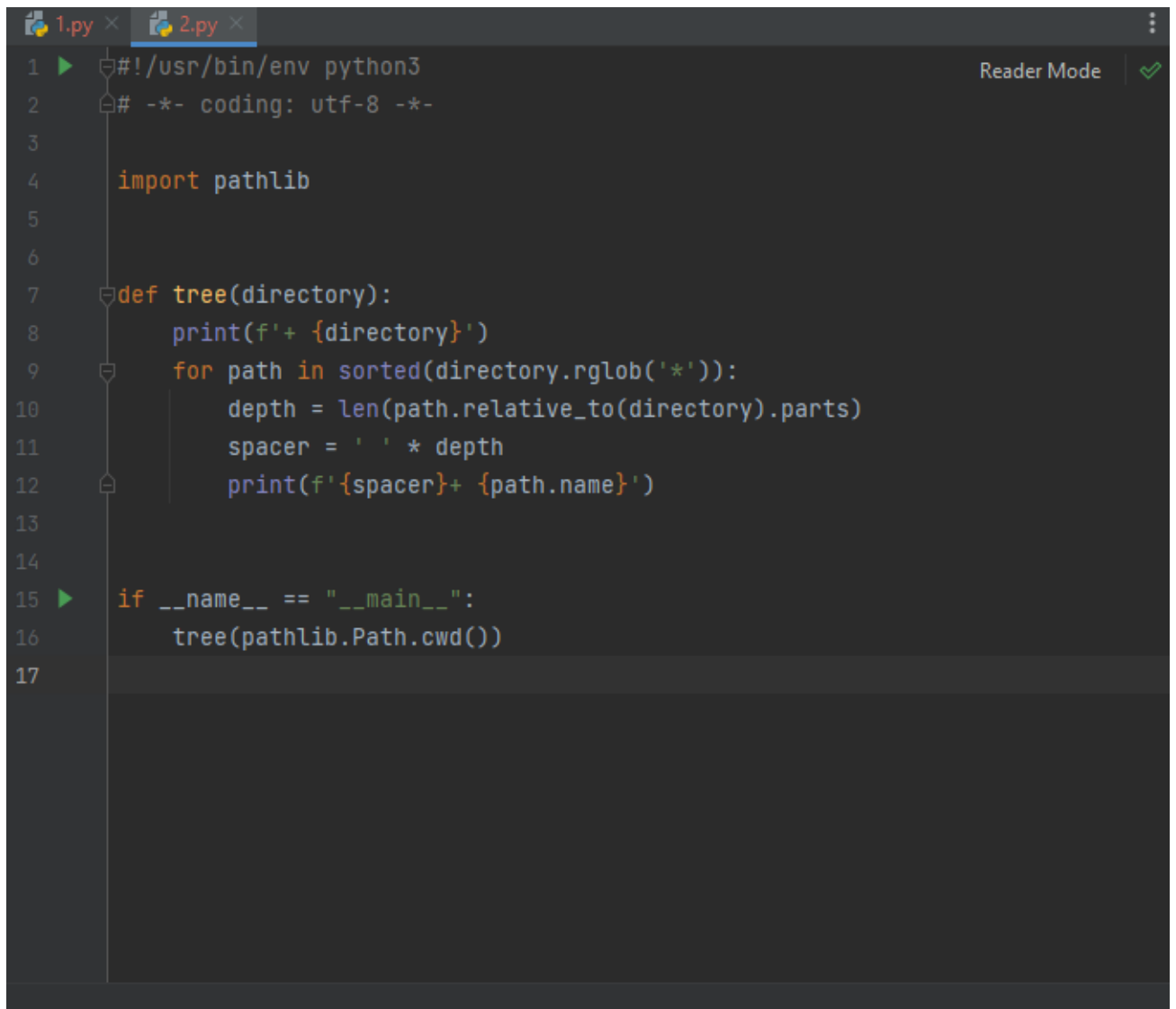


Рисунок 3 – проработал 1 пример



The image shows a code editor with two tabs: '1.py' and '2.py'. The '2.py' tab is active, displaying a Python script. The script defines a function 'tree(directory)' that recursively prints the contents of a directory. It uses 'pathlib' for path manipulation. The main block calls 'tree(pathlib.Path.cwd())' to print the current directory's structure. The editor includes a line number margin on the left (1-17), a 'Reader Mode' toggle with a checkmark on the right, and a vertical scrollbar on the left side of the code area.

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import pathlib
5
6
7      def tree(directory):
8          print(f'+ {directory}')
9          for path in sorted(directory.rglob('*')):
10             depth = len(path.relative_to(directory).parts)
11             spacer = ' ' * depth
12             print(f'{spacer}+ {path.name}')
13
14
15  ▶  if __name__ == "__main__":
16         tree(pathlib.Path.cwd())
17
```

Рисунок 4 – проработал 2 пример

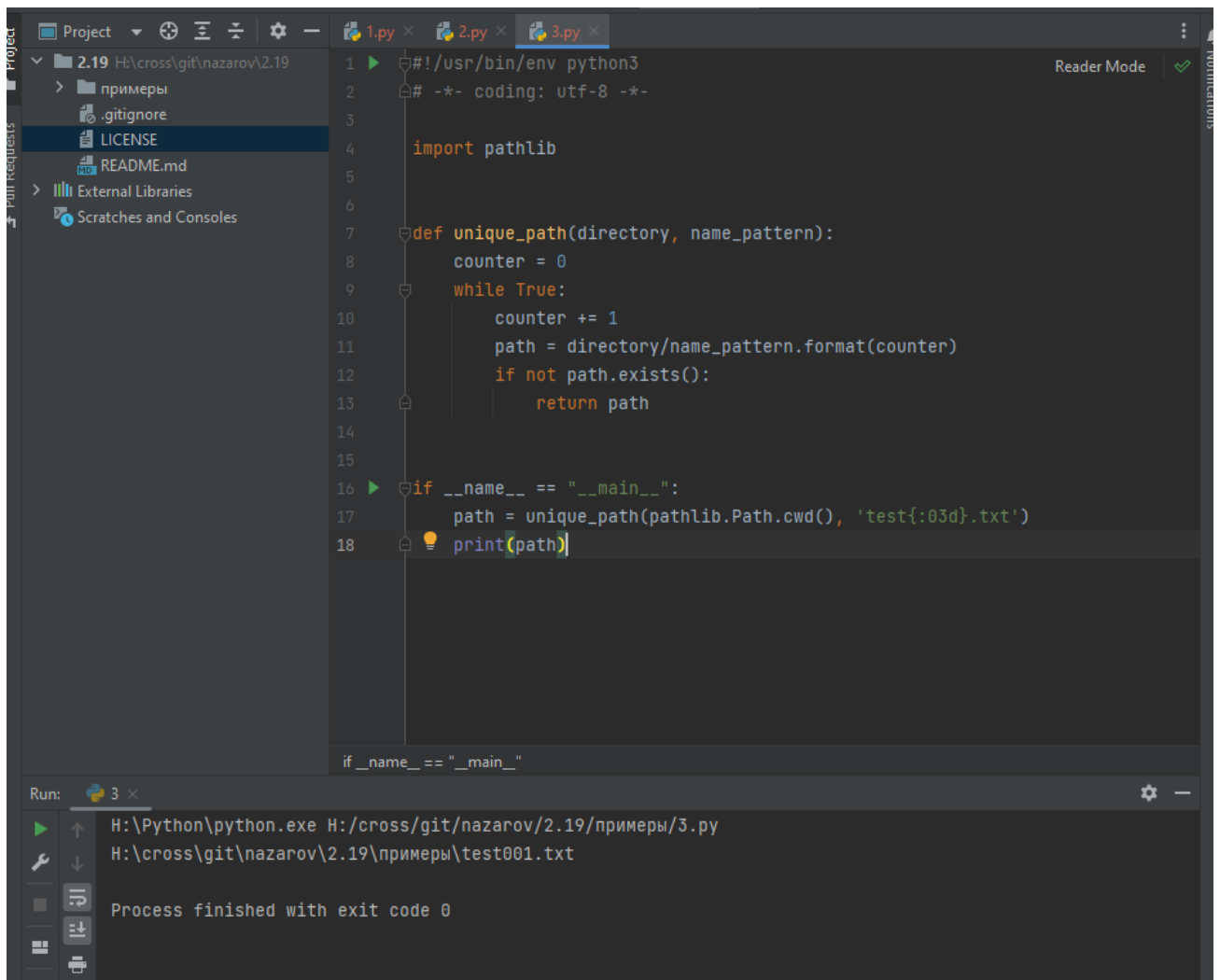


Рисунок 5 – проработал 3 пример

```

def save_products(file_name, products):
    """
    Сохранить все товары 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
        json.dump(products, fout, ensure_ascii=False, indent=4)
    directory = pathlib.Path.cwd().joinpath(file_name)
    directory.replace(pathlib.Path.home().joinpath(file_name))

```

Рисунок 6 – 1 индивидуальное задание

```

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
)

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("mk", 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")

```

Рисунок 7 – 2 индивидуальное задание.

### Контрольные вопросы:

1. Какие существовали средства для работы с файловой системой до Python 3.4?

- Методы строк, например `path.split("\\", maxsplit=1)[0]`
- Модуль `os.path`

2. Что регламентирует PEP 428?

Модуль `Pathlib` – Объектно-ориентированные пути файловой системы

3. Как осуществляется создание путей средствами модуля `pathlib`?

Есть несколько разных способов создания пути. Прежде всего, существуют classmethods наподобие `.cwd()` (текущий рабочий каталог) и `.home()` (домашний каталог вашего пользователя)

4. Как получить путь дочернего элемента файловой системы с помощью модуля `pathlib`?

При помощи метода `resolve()`.

5. Как получить путь к родительским элементам файловой системы с помощью модуля `pathlib`?

При помощи свойства `parent`.

6. Как выполняются операции с файлами с помощью модуля `pathlib`?

- перемещение;
- удаление файлов;
- подсчёт файлов;
- найти последний изменённый файл;
- создать уникальное имя файла;
- чтение и запись файлов.

7. Как можно выделить компоненты пути файловой системы с помощью модуля pathlib?

```
.name
.parent
.stem
.suffix
.anchor
```

8. Как выполнить перемещение и удаление файлов с помощью модуля pathlib?

```
.replace() – метод перемещения файлов
.unlink() – метод удаления файлов
```

9. Как выполнить подсчет файлов в файловой системе?

Метод `.iterdir()`

10. Как отобразить дерево каталогов файловой системы?

```
def tree(directory):
    print(f'+ {directory}')
    for path in sorted(directory.rglob('*')):
        depth = len(path.relative_to(directory).parts)
        spacer = ' ' * depth
        print(f'{spacer}+ {path.name}')
```

11. Как создать уникальное имя файла?

```
def unique_path(directory, name_pattern):
    counter = 0
    while True:
        counter += 1
        path = directory/name_pattern.format(counter)
```



```
if not path.exists():  
    return path  
path = unique_path(pathlib.Path.cwd(), 'test{:03d}.txt')
```

12. Каковы отличия в использовании модуля `pathlib` для различных операционных систем?

Ранее мы отмечали, что когда мы создавали экземпляр `pathlib.Path`, возвращался либо объект `WindowsPath`, либо `PosixPath`. Тип объекта будет зависеть от операционной системы, которую вы используете. Эта функция позволяет довольно легко писать кроссплатформенный код. Можно явно запросить `WindowsPath` или `PosixPath`, но вы будете ограничивать свой код только этой системой без каких-либо преимуществ. Такой конкретный путь не может быть использован в другой системе.

**Вывод:** в результате выполнения лабораторной работы были приобретены теоретические сведения и практические навыки для работы с файловой системой с помощью библиотек `pathlib` и `calorama` языка программирования Python версии 3.x..