

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

Артемьев Андрей В. « _____ » 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..

Ход работы:

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

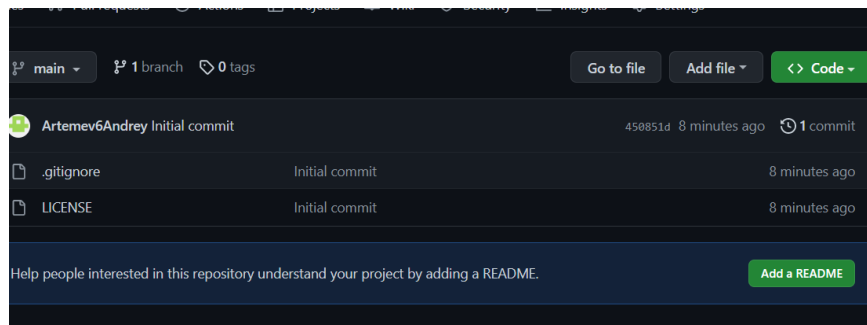


Рисунок 1.1 – Созданный репозиторий

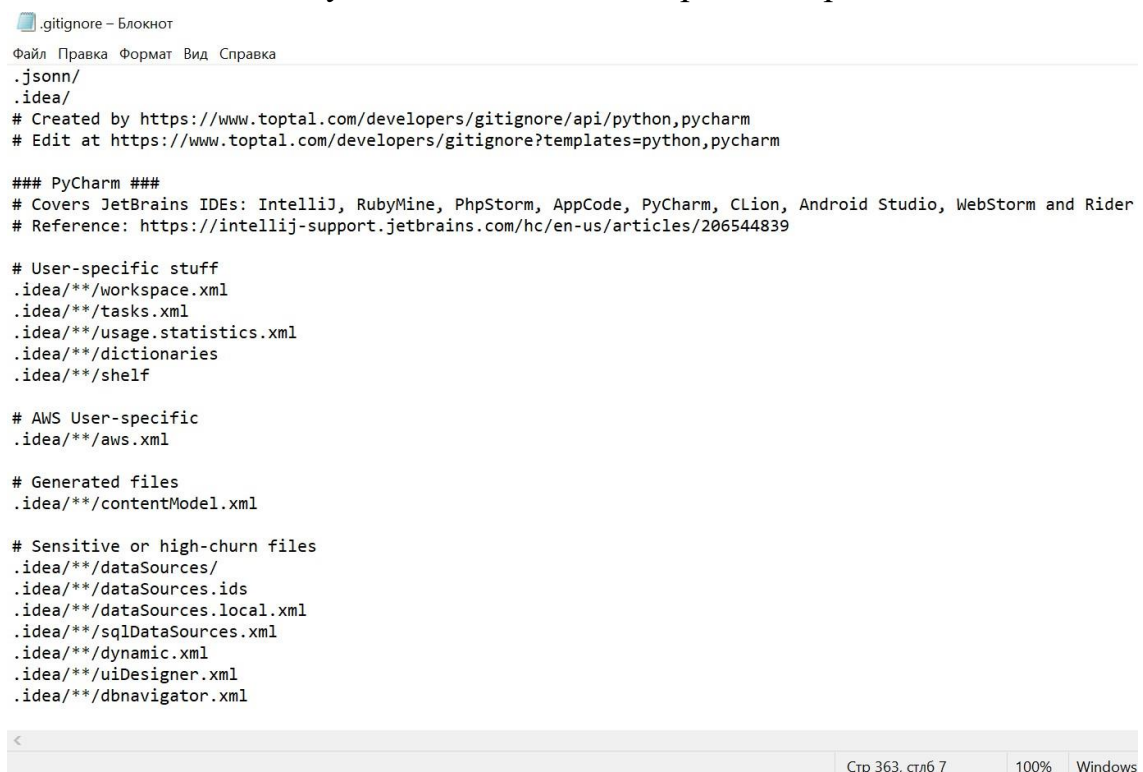


Рисунок 1.2 – Дополнил правила в .gitignore

```

aa715@Admin MINGW64 ~/OneDrive/Рабочий стол/Python14-2.19 (main)
$ git flow init

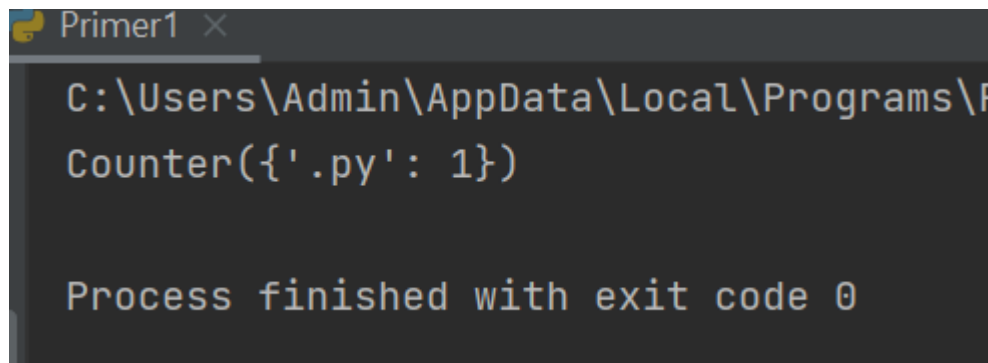
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

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? [C:/Users/aa715/OneDrive/Рабочий стол/Python14-2.19/.git/hooks]

```

Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Проработал примеры лабораторной работы.

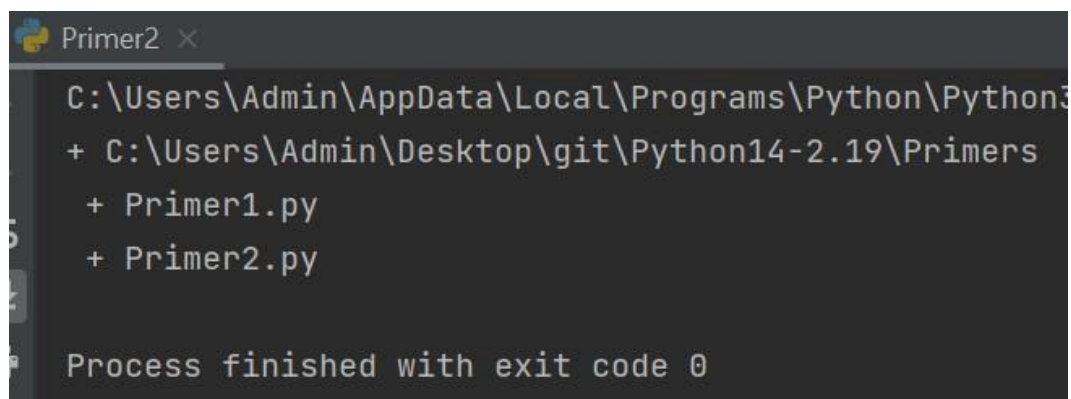


```

Primer1 x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe
Counter({' .py': 1})
Process finished with exit code 0

```

Рисунок 2 – Результат работы примера №1

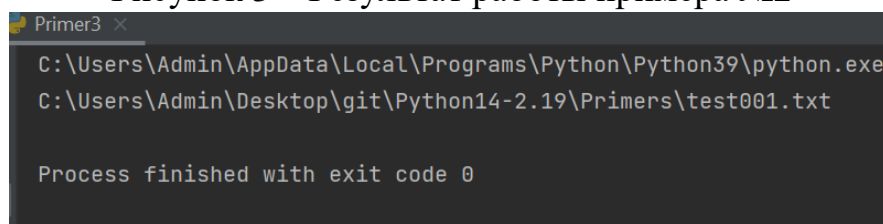


```

Primer2 x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe
+ C:\Users\Admin\Desktop\git\Python14-2.19\Primers
+ Primer1.py
+ Primer2.py
Process finished with exit code 0

```

Рисунок 3 – Результат работы примера №2



```

Primer3 x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe
C:\Users\Admin\Desktop\git\Python14-2.19\Primers\test001.txt
Process finished with exit code 0

```

Рисунок 4 – Результат работы примера №3

Задание №1. Для своего варианта лабораторной работы 2.17 добавьте возможность хранения файла данных в домашнем каталоге пользователя. Для выполнения операций с файлами необходимо использовать модуль `pathlib`.

```
def save_students(file_name, students):  
    """  
    Сохранение данных  
    """  
    with open(file_name, "w", encoding="utf-8") as fout:  
        json.dump(students, fout, ensure_ascii=False, indent=4)  
    directory = pathlib.Path.cwd().joinpath(file_name)  
    directory.replace(pathlib.Path.home().joinpath(file_name))
```

Рисунок 5 – Добавление возможности хранения файлов в домашнем каталоге

```
C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind>python individual1.py add  
new.json --name="Artemev" --group="3" --grade="2 2 3 2 2"  
C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind>
```

Рисунок 6 – Проверка работы программы

Задание 2. Разработайте аналог утилиты `tree` в Linux. Используйте возможности модуля `argparse` для управления отображением дерева каталогов файловой системы.

Проверка работоспособности кода:

```
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",  
        version=f'%(prog)s 1.0.0',
```

Рисунок 7 – Код разработанной программы

```

>>> tuqtlqngjs'bl
>>> tuqtlqngjs'bl
>>> qsfsg'zou
>>> C:/pzeiz/sgdtz/ouertile/bsqonnn stou/blfrouth-5'Id/tuq
C:/pzeiz/sgdtz/ouertile/bsqonnn stou/blfrouth-5'Id/tuq>blfrou tuqtlqngjs'bl
C:/pzeiz/sgdtz/ouertile/bsqonnn stou/blfrouth-5'Id/tuq>

```

Рисунок 8 – Запуск программы без передачи ей параметров

```

C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind>python individual2.py mkd
ir .abc
>>> C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind
>>> .abc
>>> data2.json
>>> individual1.py
>>> individual2.py

C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind>python individual2.py rmd
ir .abc
>>> C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind
>>> data2.json
>>> individual1.py
>>> individual2.py

C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind>

```

Рисунок 9 – Создание и удаление каталога

```

C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind>python individual2.py tou
ch artemev.txt
>>> C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind
>>> artemev.txt
>>> data2.json
>>> individual1.py
>>> individual2.py

C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind>python individual2.py rm
artemev.txt
>>> C:\Users\aa715\OneDrive\Рабочий стол\Python14-2.19\ind
>>> data2.json
>>> individual1.py
>>> individual2.py

```

Рисунок 10 – Создание и удаление файла

Вывод: в результате выполнения лабораторной работы были приобрете-
нены теоретические сведения и практические навыки для работы с файловой
системой с помощью библиотек pathlib и salogama языка программирования
Python версии 3.x..

Ответы на 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.