

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

Кафедра инфокоммуникаций

**Отчет по лабораторной работе №14**

**Работа с файловой системой в Python3 с использованием модуля pathlib**

**По дисциплине «Программирование на Python**

Выполнил студент группы ИВТ-б-о-20-1

Харченко Б.Р. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

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

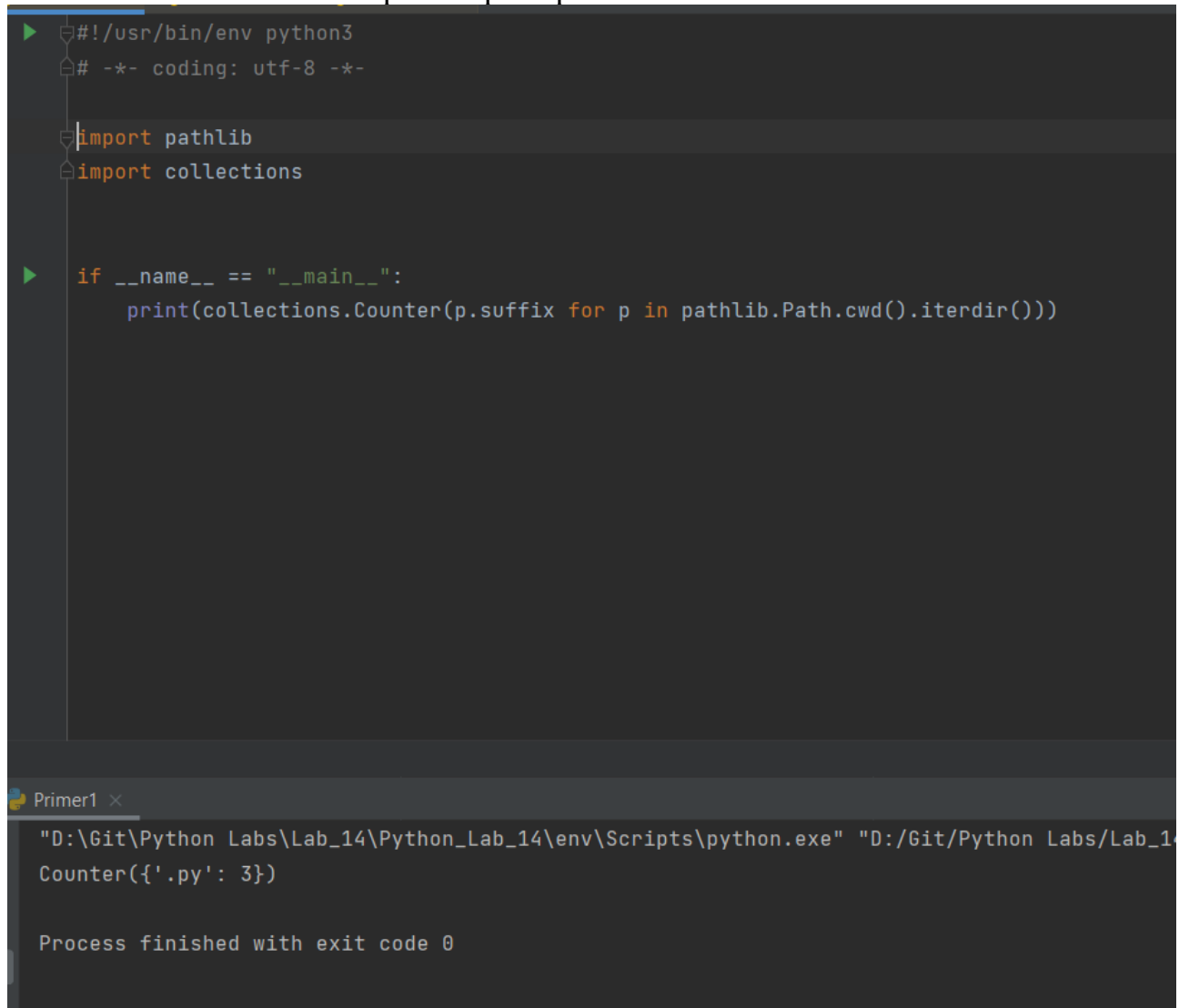
Проверил Воронкин Р. А. \_\_\_\_\_

(подпись)

**Цель работы:** приобретение навыков по работе с файловой системой с помощью библиотеки `pathlib` языка программирования Python версии 3.x.

**Ход работы:**

1. Создал новый репозиторий на `github`, после клонировал его и создал в папке репозитория новый проект PyCharm.
2. Выполнил первый пример.



The image shows a PyCharm IDE window with a Python script and its terminal output. The script is a simple program that uses `pathlib` and `collections` to count the number of files with a specific suffix in the current directory. The terminal output shows the execution of the script, which counts 3 files with the suffix `.py`.

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

import pathlib
import collections

if __name__ == "__main__":
    print(collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir()))
```

Primer1 x

```
"D:\Git\Python Labs\Lab_14\Python_Lab_14\env\Scripts\python.exe" "D:/Git/Python Labs/Lab_14/Primer1.py"
Counter({' .py': 3})

Process finished with exit code 0
```

Рисунок 1. Пример 1

3. Выполнил второй пример.

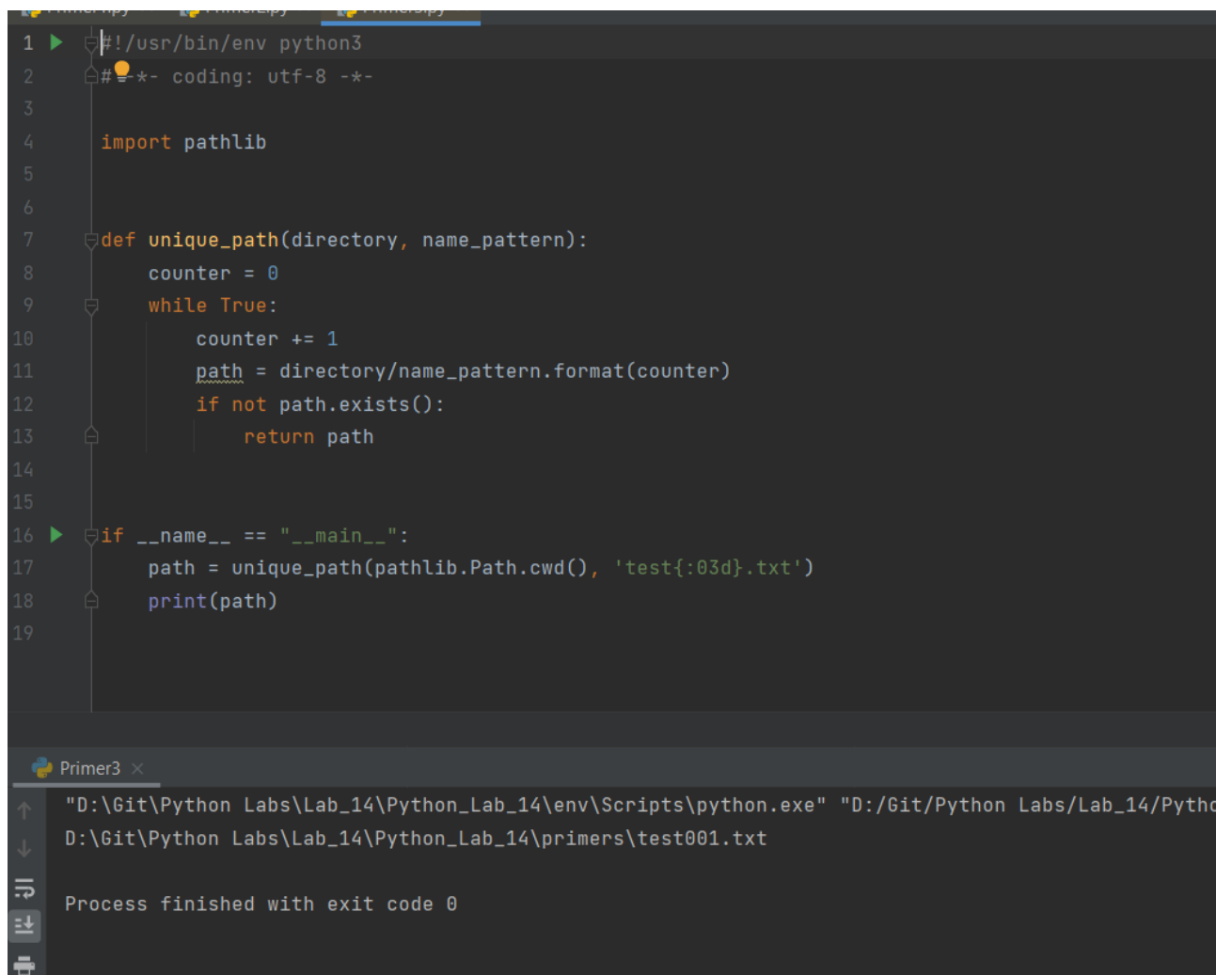
```
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import pathlib
5
6
7 def tree(directory):
8     print(f'+ {directory}')
9     for path in sorted(directory.rglob('*')):
10         depth = len(path.relative_to(directory).parts)
11         spacer = ' ' * depth
12         print(f'{spacer}+ {path.name}')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

Primer2 x

```
"D:\Git\Python Labs\Lab_14\Python_Lab_14\env\Scripts\python.exe" "D:/G
+ D:\Git\Python Labs\Lab_14\Python_Lab_14\primers
+ Primer1.py
+ Primer2.py
+ Primer3.py
```

Рисунок 2. Пример 2

4. Выполнил третий пример.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import pathlib
5
6
7 def unique_path(directory, name_pattern):
8     counter = 0
9     while True:
10         counter += 1
11         path = directory/name_pattern.format(counter)
12         if not path.exists():
13             return path
14
15
16 if __name__ == "__main__":
17     path = unique_path(pathlib.Path.cwd(), 'test{:03d}.txt')
18     print(path)
19
```

Primer3 x

"D:\Git\Python Labs\Lab\_14\Python\_Lab\_14\env\Scripts\python.exe" "D:/Git/Python Labs/Lab\_14/Python\_Lab\_14\primers\test001.txt"

Process finished with exit code 0

Рисунок 3. Пример 3

5. Выполнил первое индивидуальное задание



```

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

import argparse
import pathlib
import colorama
import collections

def tree(directory):
    print('\033[1;31m' + f'>>> {directory}')
    for path in sorted(directory.rglob('*')):
        depth = len(path.relative_to(directory).parts)
        spacer = '\t' * depth
        print('\033[1;32m' + f'{spacer} >> {path.name}')
        for new_path in sorted(directory.joinpath(path).rglob('*')):
            depth = len(new_path.relative_to(directory.joinpath(path)).parts)
            spacer = '\t\t' * depth
            print('\033[1;33m' + f'{spacer} > {new_path.name}')

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

    parser = argparse.ArgumentParser("tree")
    parser.add_argument(
        "--version",
        action="version",
    )

```

Рисунок 6. Второе индивидуальное задание

```

(env) D:\Git\Python Labs\Lab_14\Python_Lab_14\individual>python individual2.py
>>> D:\Git\Python Labs\Lab_14\Python_Lab_14\individual
>> individual1.py
>> individual2.py

```

Рисунок 7. Результат работы кода

### Контрольные вопросы:

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

- Методы строк, например `path.rsplitlet("\\", 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 ходе выполнения лабораторной работы приобрел навыки по работе с файловой системой с помощью библиотеки `pathlib` языка программирования Python версии 3.x.