

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

Отчет по лабораторной работе № 2.19

Работа с файловой системе в Python3 с использованием модуля pathlib

Выполнил студент группы ИВТ-б-о-20-1

Погорелов Д.Н « » \_\_\_\_\_ 20\_\_ г.

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

Проверил 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 общедоступный репозиторий и клонировал его на локальный сервер.
1. Ознакомившись с теоретическим материалом, выполнил примеры, создав для них отдельный модуль.



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

import pathlib
import collections

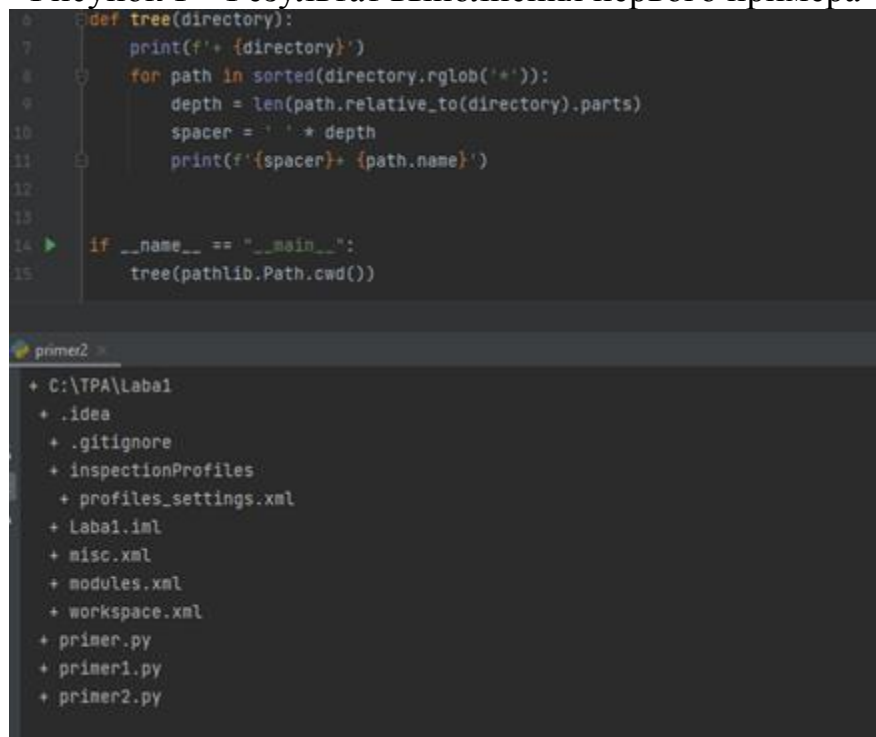
print(collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir()))
```

primer1

```
Counter({'py': 2, '': 1})

Process finished with exit code 0
```

Рисунок 1 – Результат выполнения первого примера



```
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}')

if __name__ == '__main__':
    tree(pathlib.Path.cwd())
```

primer2

```
+ C:\TPA\Lab1
+ .idea
+ .gitignore
+ inspectionProfiles
+ profiles_settings.xml
+ Lab1.iml
+ misc.xml
+ modules.xml
+ workspace.xml
+ primer.py
+ primer1.py
+ primer2.py
```

Рисунок 2 – Результат выполнения второго примера

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

import pathlib

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')
print(path)

unique_path() while True
```

primer

C:\TPA\Lab1\test001.txt

Process finished with exit code 0

Рисунок 3 – Результат выполнения третьего примера

Рисунок 4 – Результат выполнения четвертого примера

```
>>> import pathlib
>>> path = pathlib.PureWindowsPath(r'C:\TPA\Lab1\primer3.py')
>>> path.name
'primer3.py'
>>> path.parent
```

2. Приступил к выполнению индивидуальных заданий своего варианта.
3. Добавил возможность хранения файла json в домашнем каталоге пользователя.

```
103 def save_students(file_name, students):
104     """
105     Сохранить данные о студенте
106     """
107     with open(file_name, "w", encoding="utf-8") as fout:
108         json.dump(students, fout, ensure_ascii=False, indent=4)
109     directory = pathlib.Path.cwd().joinpath(file_name)
110     directory.replace(pathlib.Path.home().joinpath(file_name))
```

Рисунок 5 – Возможность хранения файла в домашнем каталоге

4. Затем приступил к выполнению второго задания.

**Условие:** разработайте аналог утилиты tree в Linux. Используйте возможности модуля argparse для управления отображением дерева каталогов файловой системы. Добавьте дополнительные уникальные возможности в данный программный продукт.

5. Написал код для реализации задачи.

```
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).rglob('*')):
            depth = len(new_path.relative_to(directory.joinpath(path)).parts)
            spacer = '\t' * depth
            print(Fore.BLUE + f'{spacer} > {new_path.name}')
```

Рисунок 7 – Функция def tree

```
23 def main(command_line=None):
24     colorama.init()
25     current = pathlib.Path.cwd()
26     file_parser = argparse.ArgumentParser(add_help=False)
27
28     # Создаем основной парсер командной строки
29     parser = argparse.ArgumentParser("tree")
30     parser.add_argument(
31         "--version",
32         action="version",
33         help="The main parser",
34         version="% (prog)s 0.1.0"
35     )
36
37     subparsers = parser.add_subparsers(dest="command")
38
39     # Создаем субпарсер для создания новой папки
40     create = subparsers.add_parser(
41         "mkdir",
42         parents=[file_parser]
43     )
44     create.add_argument(
45         "filename",
46         action="store"
47     )
48
49 if __name__ == "__main__":
```

Рисунок 8 – Функция def main

```

77     args = parser.parse_args(command_line)
78     if args.command == 'mkdir':
79         directory_path = current / args.filename
80         directory_path.mkdir()
81         tree(current)
82     elif args.command == "rmdir":
83         directory_path = current / args.filename
84         directory_path.rmdir()
85         tree(current)
86     elif args.command == "touch":
87         directory_path = current / args.filename
88         directory_path.touch()
89         tree(current)
90     elif args.command == "rm":
91         directory_path = current / args.filename
92         directory_path.unlink()
93         tree(current)
94     else:
95         tree(current)
96
97
98 if __name__ == "__main__":
99     main()

```

Рисунок 9 – Код программы

6. Сделал проверку разработанной программы через Anaconda.

Вывод: в ходе занятия были приобретены навыки по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.

### 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?

Когда мы создаем экземпляр `pathlib.Path`, возвращается либо объект `WindowsPath`, либо `PosixPath`. Тип объекта будет зависеть от операционной системы. Эта функция позволяет довольно легко писать кроссплатформенный код. Можно явно запросить `WindowsPath` или `PosixPath`, но код будет ограничен только этой системой без каких-либо преимуществ.