

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

**«Работа с файловой системе в Python3 с использованием модуля
pathlib»**

Отчет по лабораторной работе № 2.19
по дисциплине «Основы программной инженерии»

Выполнил студент группы

ПИЖ-б-о-21-1

Трушева В. О. _____. « » 2023г.

Подпись студента _____

Работа защищена « _____ 20 __ г.

Проверила Воронкин Р.А. _____

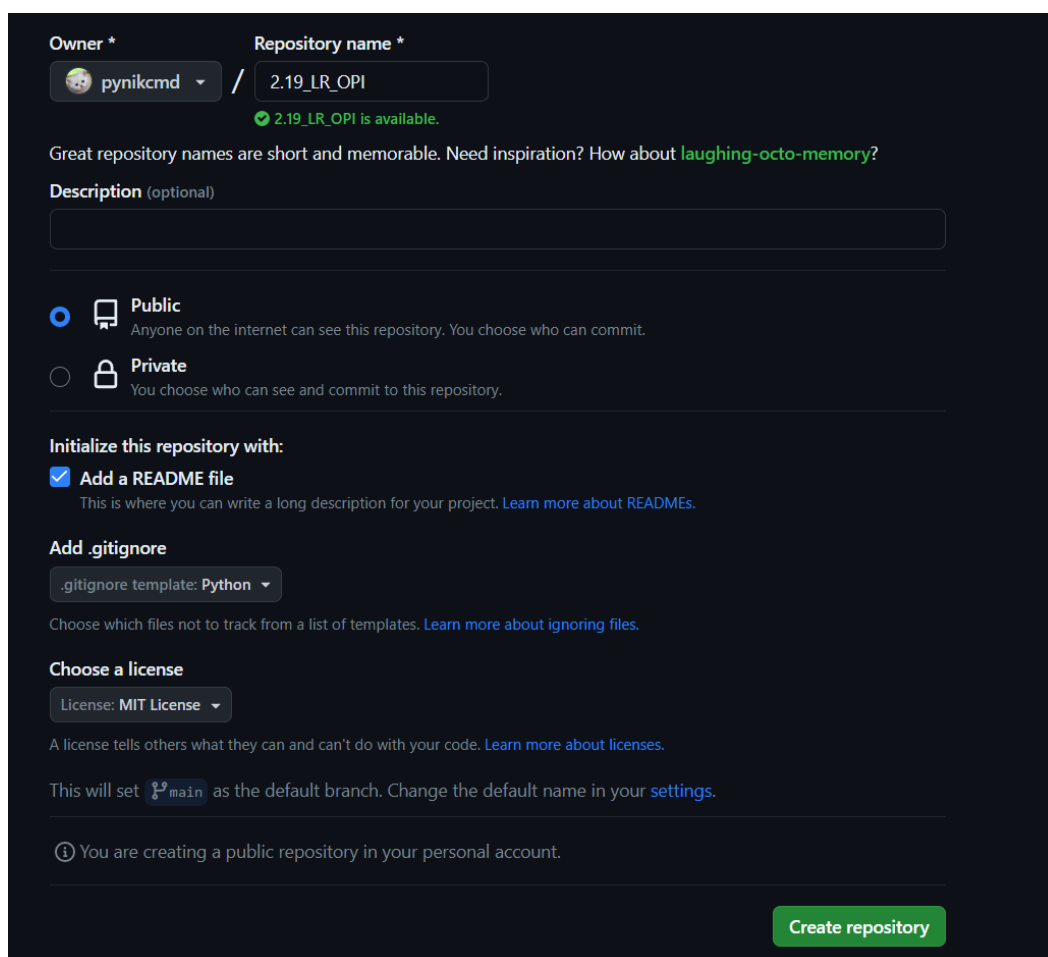
(подпись)

Ставрополь 2023


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

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.




Owner * / Repository name *


 pynikcmd / 2.19_LR_OPI

✔ 2.19_LR_OPI is available.

Great repository names are short and memorable. Need inspiration? How about [laughing-octo-memory](#)?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:


☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)


Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

[Create repository](#)

Рисунок – Создание репозитория

3. Выполните клонирование созданного репозитория.

```
D:\fgit>git clone https://github.com/pynikcmd/2.19_LR_OPI.git
Cloning into '2.19_LR_OPI'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 5.12 KiB | 524.00 KiB/s, done.
```

Рисунок – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.



Рисунок – Дополнен gitignore

5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
D:\fgit\2.19_LR_OPI>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? [D:/fgit/2.19_LR_OPI/.git/hooks]
```

Рисунок – Модель gitflow

6. Создайте проект PyCharm в папке репозитория.

7. Проработайте примеры лабораторной работы. Создайте для них отдельные модули языка Python. Зафиксируйте изменения в репозитории.

8. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.

Пример 1



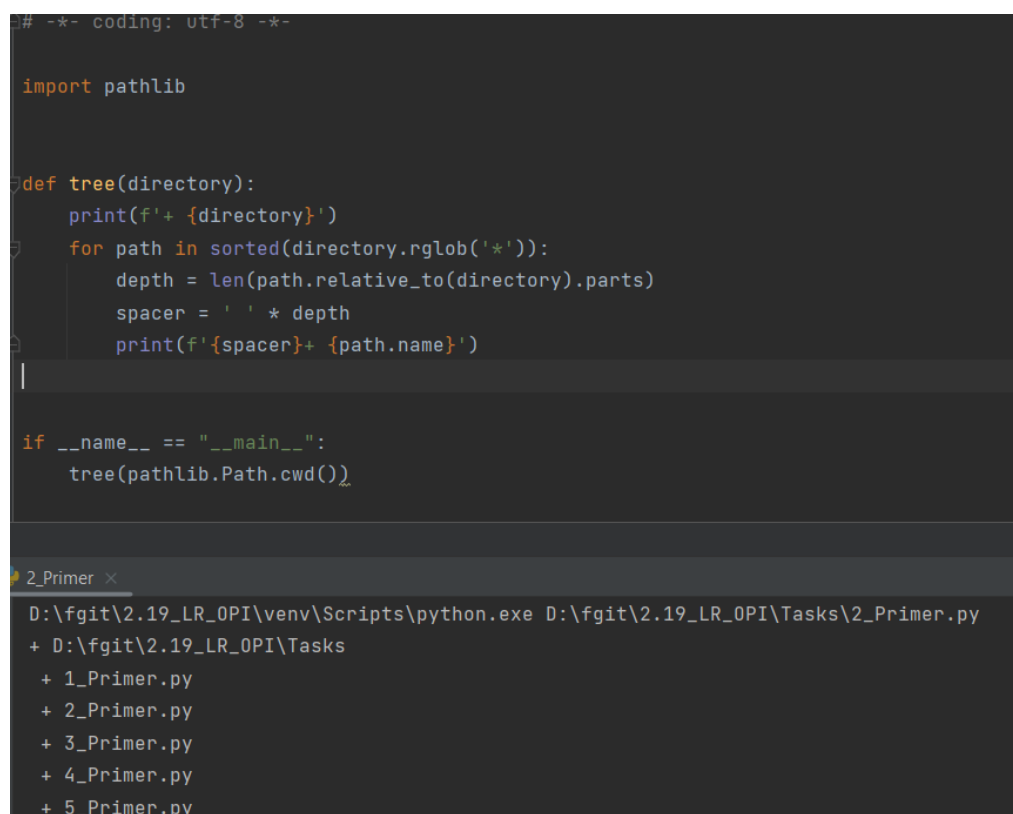
```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import pathlib
5  import collections
6
7  if __name__ == "__main__":
8      print(collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir()))
9
if __name__ == "__main__":
    print(collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir()))
```

Run: 1_Primer x

D:\fgit\2.19_LR_OPI\venv\Scripts\python.exe D:\fgit\2.19_LR_OPI\Tasks\1_Primer.py
Counter({'py': 1})

Рисунок – Результат работы программы

Пример 2



```
# -*- coding: utf-8 -*-

import pathlib

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())
```

2_Primer x

D:\fgit\2.19_LR_OPI\venv\Scripts\python.exe D:\fgit\2.19_LR_OPI\Tasks\2_Primer.py
+ D:\fgit\2.19_LR_OPI\Tasks
+ 1_Primer.py
+ 2_Primer.py
+ 3_Primer.py
+ 4_Primer.py
+ 5_Primer.py

Рисунок – Результат работы программы

Пример 3

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

from datetime import datetime
import pathlib

if __name__ == "__main__":
    time, file_path = max((f.stat().st_mtime, f) for f in pathlib.Path.cwd().iterdir())
    print(datetime.fromtimestamp(time), file_path)
```

3_Primer x

D:\fgit\2.19_LR_OPI\venv\Scripts\python.exe D:\fgit\2.19_LR_OPI\Tasks\3_Primer.py
2023-05-01 21:34:32.939523 D:\fgit\2.19_LR_OPI\Tasks\2_Primer.py

Рисунок – Результат работы программы

Пример 4

```
#!/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

if __name__ == "__main__":
    path = unique_path(pathlib.Path.cwd(), 'test{:03d}.txt')
    print(path)
```

4_Primer x

D:\fgit\2.19_LR_OPI\venv\Scripts\python.exe D:\fgit\2.19_LR_OPI\Tasks\4_Primer.py
D:\fgit\2.19_LR_OPI\Tasks\test001.txt

Рисунок – Результат работы программы

Пример 5

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

import pathlib

if __name__ == "__main__":
    path = pathlib.Path(r"D:\fgit\test.txt")
    print(path.name)
    print(path.parent)
    print(path.exists())
```

5_Primer x

D:\fgit\2.19_LR_OPI\venv\Scripts\python.exe D:\fgit\test.txt

D:\fgit

True

Рисунок – Результат работы программы

9. Зафиксируйте сделанные изменения в репозитории.

10. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.

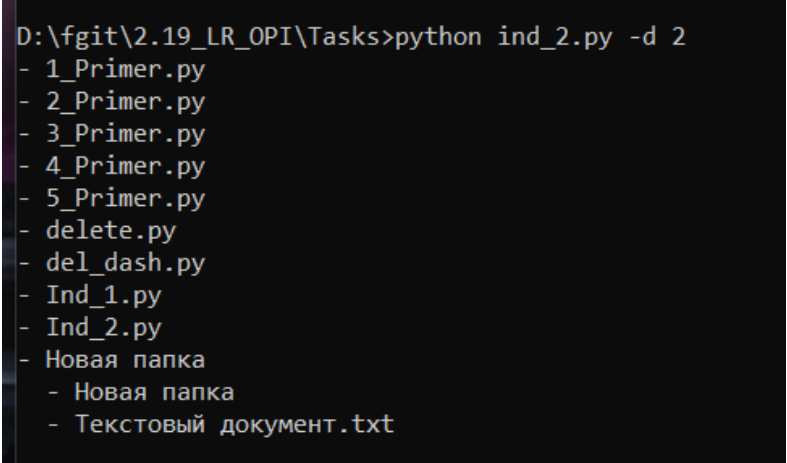
Задание 1. Для своего варианта лабораторной работы 2.17 добавьте возможность хранения файла данных в домашнем каталоге пользователя. Для выполнения операций с файлами необходимо использовать модуль `pathlib`.

```
D:\fgit\2.19_LR_OPI\Tasks>python Ind_1.py add 1.json --name="Bd fyyn" --number=12345 --bday=01.01.2023
D:\fgit\2.19_LR_OPI\Tasks>python Ind_1.py display 1.json
```

№	Фамилия и имя	Телефон	День рождения
1	Bd fn	12345	2023-01-01
2	Bd fyyn	12345	2023-01-01

Рисунок – Результат работы программы

Задание 2. 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.



```
D:\fgit\2.19_LR_OPI\Tasks>python ind_2.py -d 2
- 1_Primer.py
- 2_Primer.py
- 3_Primer.py
- 4_Primer.py
- 5_Primer.py
- delete.py
- del_dash.py
- Ind_1.py
- Ind_2.py
- Новая папка
  - Новая папка
  - Текстовый документ.txt
```

Рисунок – Результат работы программы

11. Зафиксируйте сделанные изменения в репозитории.
12. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
13. Выполните слияние ветки для разработки с веткой master/main.
14. Отправьте сделанные изменения на сервер GitHub.
15. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Контрольные вопросы

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

До Python 3.4 работа с путями файловой системы осуществлялась либо с помощью методов строк:

```
path.rsplit('\\', maxsplit=1)[0]
```

либо с помощью модуля `os.path` :

```
os.path.isfile(os.path.join(os.path.expanduser('~'), 'realpython.txt'))
```

2. Что регламентирует PEP 428?

PEP 428 - "The pathlib module - representing file system paths as objects" регламентирует стандартизированный интерфейс для работы с путями файловой системы в Python, представленный модулем `pathlib`. PEP описывает основные концепции и интерфейсы, предоставляемые модулем, и детально описывает класс `Path`, который представляет собой кросс-платформенный объект пути, совместимый с операционными системами Windows, Linux и macOS. PEP 428 был принят в Python 3.4 и выше.

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

Создание путей в модуле `pathlib` осуществляется с помощью класса `Path`.

```
# абсолютный путь
```

```
path = Path('/home/user/file.txt')
```

```
# относительный путь
```

```
path = Path('folder/subfolder/file.txt')
```

```
# создание пути из другого объекта pathlib
```

```
path1 = Path('/home/user')
```

```
path2 = path1 / 'file.txt'
```



```
# создание пути из нескольких частей
path = Path().joinpath('folder', 'subfolder', 'file.txt')
```

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

```
# получение абсолютного пути дочерней папки
child_dir_path = Path('.').resolve() / 'child_folder'
# получение абсолютного пути дочернего файла
child_file_path = Path('.').resolve() / 'child_folder' / 'file.txt'
```

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

Атрибут `parent` возвращает объект пути, представляющий родительскую директорию текущего элемента файловой системы.

получение пути к родительской директории текущего файла или директории

```
parent_dir_path = Path('.').parent
```

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

Например, чтобы прочитать содержимое файла, можно использовать метод `read_text()`:

```
# создание объекта пути для файла
file_path = Path('/path/to/file.txt')
# чтение содержимого файла в виде строки
file_content = file_path.read_text()
```

Аналогично можно записать данные в файл с помощью методов `write_text()`:

```
# запись строки в файл
```

```
file_path.write_text('Hello, world!')
```

Чтобы удалить файл, можно использовать метод `unlink()`:

```
# удаление файла
```

```
file_path.unlink()
```

Для переименования файла можно использовать метод `rename()`:

```
# переименование файла
```

```
file_path.rename('/path/to/new_file.txt')
```

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

Различные части пути удобно доступны как свойства. Основные примеры включают в себя:

`.name` : имя файла без какого-либо каталога

`.parent` : каталог, содержащий файл, или родительский каталог, если путь является каталогом

`.stem` : имя файла без суффикса

`.suffix` : расширение файла

`.anchor` : часть пути перед каталогами

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

Чтобы переместить файл, используйте `.replace()`. Обратите внимание, что если место назначения уже существует, `.replace()` перезапишет его. К сожалению, `pathlib` явно не поддерживает безопасное перемещение файлов. Чтобы избежать возможной перезаписи пути назначения, проще всего проверить, существует ли место назначения перед заменой:

```
if not destination.exists():
    source.replace(destination)
```

Когда вы переименовываете файлы, полезными методами могут быть `.with_name()` и `.with_suffix()`. Они оба возвращают исходный путь, но с замененным именем или суффиксом соответственно.

```
path
PosixPath('/home/gahjelle/realpython/test001.txt')
path.with_suffix('.py')
PosixPath('/home/gahjelle/realpython/test001.py')
path.replace(path.with_suffix('.py'))
```

Каталоги и файлы могут быть удалены с помощью `.rmdir()` и `.unlink()` соответственно.

9. Как выполнить подсчет файлов в файловой системе?

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

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

Одно из главных различий между операционными системами заключается в символах, используемых для разделения каталогов в путях. Например, на Windows используется обратный слеш `\`, а на Unix-подобных системах (Linux, macOS) используется прямой слеш `/`. Чтобы справиться с этой разницей, модуль `pathlib` предоставляет два класса, `WindowsPath` и `PosixPath`, которые могут использоваться для работы с путями на соответствующих операционных системах.

Еще одно различие связано с регистром букв в именах файлов и директорий. Например, на Windows имена файлов и директорий регистрозависимы, т.е. два пути, отличающиеся только регистром букв, будут считаться разными. На Unix-подобных системах имена файлов и директорий регистронезависимы.

Кроме того, есть и другие небольшие различия в использовании `pathlib` на различных операционных системах. Например, на Windows могут возникать проблемы при работе с путями, содержащими зарезервированные имена (например, `CON`, `PRN`, `LPT1`), а на Unix-подобных системах могут возникать проблемы при работе с символическими ссылками.