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

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 2.19

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

по дисциплине «Основы программной инженерии»

Выполнил студент группы
ПИЖ-б-о-21-1
Зиберов Александр
« » мая 2023 г.
Подпись студента_____
Работа защищена
« »_____20__г.
Проверил 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.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: https://github.com/afk552/lab2.19

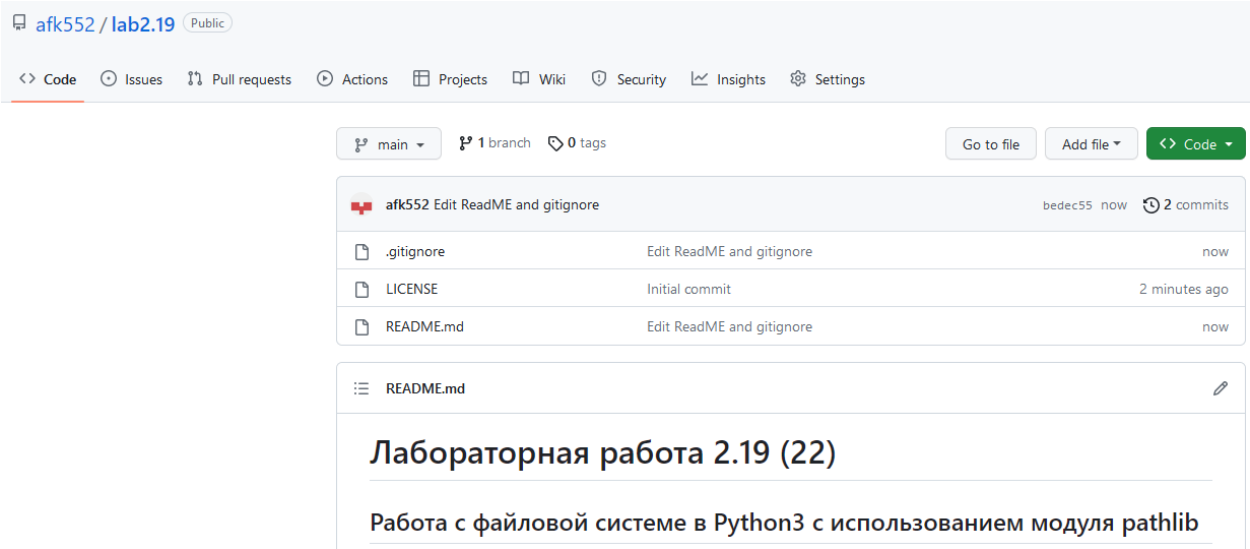

Рисунок 1 – Удаленный репозиторий на GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

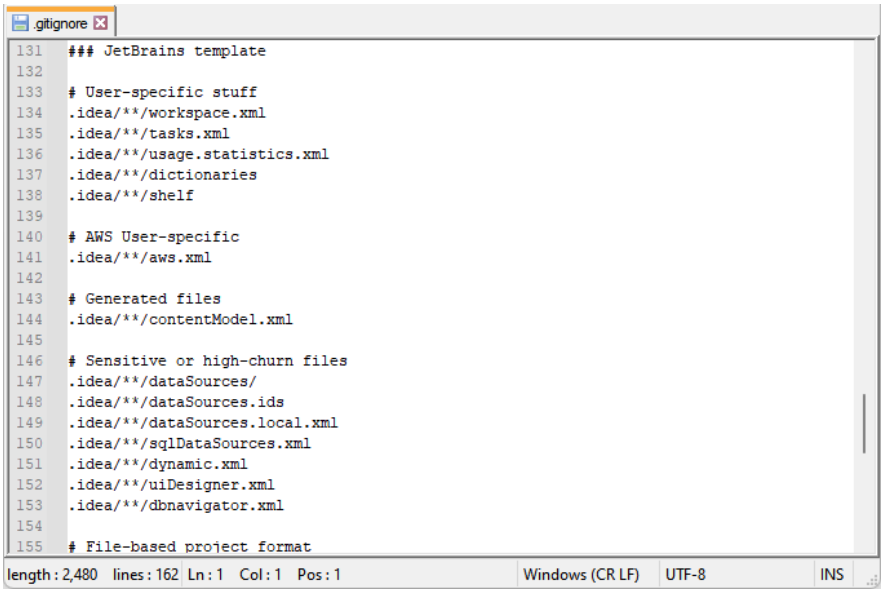

Рисунок 2 – Окно блокнота

Организуйте свой репозиторий в соответствие с моделью ветвления git-flow, рисунок 3.

```
C:\git\lab2.19>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.19>git
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

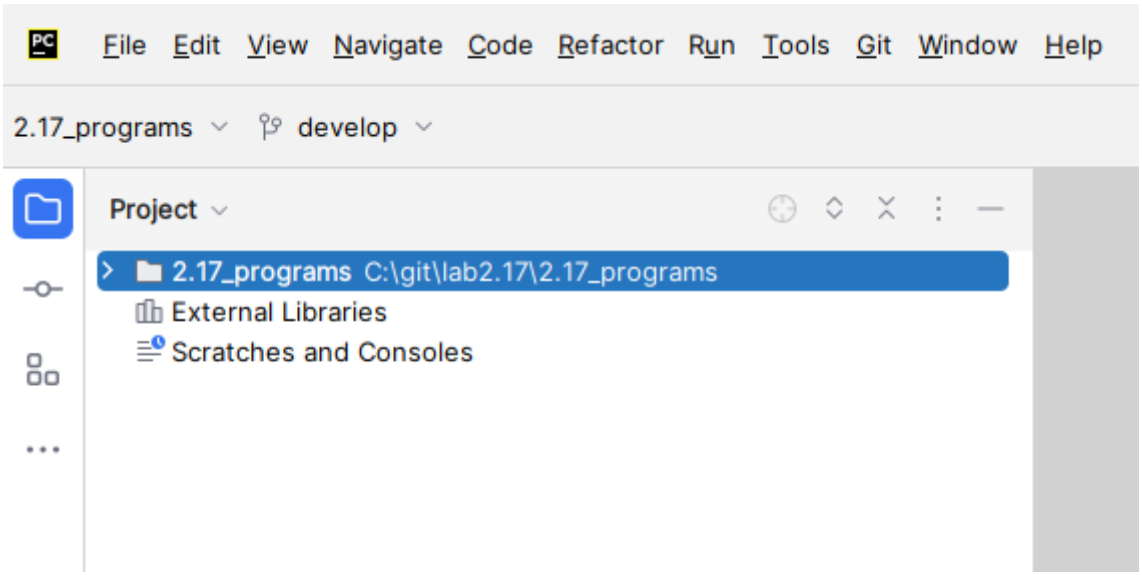

Рисунок 4 – Окно проекта в PyCharm

Примеры

example_1.py

```
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
```

Рисунок 5 – Код примера 1

```
Counter({'.py': 6})

Process finished with exit code 0
```

Рисунок 6 – Выполнение примера 1

example_2.py

```python
#!/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().glob('*.p*')))

```

Рисунок 7 – Код примера 2

```
Counter({'.py': 6})

Process finished with exit code 0
```

Рисунок 8 – Выполнение примера 2

example_3.py

```python
#!/usr/bin/env python3
# -*- 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__":
    print(tree(pathlib.Path.cwd()))

```

Рисунок 9 – Код примера 3

```
+ C:\git\lab2.19\examples
 + example_1.py
 + example_2.py
 + example_3.py
 + example_4.py
 + example_5.py
 + example_6.py
None

Process finished with exit code 0
```

Рисунок 10 – Выполнение примера 3

example_4.py

```python
#!/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)

```

Рисунок 11 – Код примера 4

```
2023-05-15 02:59:15.915895 C:\git\lab2.19\examples\example_1.py

Process finished with exit code 0
```

Рисунок 12 – Выполнение примера 4

example_5.py

```python
#!/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)

```

Рисунок 13 – Код примера 5

```
C:\git\lab2.19\examples\test001.txt

Process finished with exit code 0
```

Рисунок 14 – Выполнение примера 5

example_6.py

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

import pathlib

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

```

Рисунок 15 – Код примера 6

```
test.txt
C:\
False

Process finished with exit code 0
```

Рисунок 16 – Выполнение примера 6

Индивидуальное задание.

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_workers(file_name, people_list):
    """
    Сохранение списка людей в json
    """
    # Сохранение файла данных в домашнем каталоге пользователя
    file_name = f"{str(Path.home())}/{str(file_name)}"
    # Проверка заданного имени файла
    if file_name.split(".", maxsplit=1)[-1] != "json":
        print("Заданный формат файла не .json", file=sys.stderr)
        return False

    # Делаем копию списка, чтобы его не затронуть
    lst = copy.deepcopy(people_list)
    # Сериализация даты в строку для записи в файл
    list(lst)
    print(lst)
    for i in lst:
        i["birth"] = i["birth"].strftime("%d.%m.%Y")

    # Дамп в json списка
    with open(file_name, "w", encoding="utf-8") as f_out:
        json.dump(lst, f_out, ensure_ascii=False, indent=4)
    lst.clear()
```

Рисунок 17 – Код программы индивидуального задания 1

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.

indiv2_tree.py

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

from pathlib import *
import click


@click.command()
@click.option("--pth", default=Path.cwd(), help="Директория", type=str)
@click.option("--level", default=-1, help="Уровень вложенности", type=int)
@click.option(
    "--option",
    default="1",
    help=(
        "1 - Показать файлы, папки и подпапки, "
        "2 - Показать папки и подпапки, "
        "3 - Показать только папки."
    ),
    type=int,
)
def tree(pth, level, option):
    print(f"{pth}")
    match option:
        case 1:
            contents = Path(pth).rglob("*")
        case 2:
            contents = Path(pth).glob("**")
        case 3:
            contents = Path(pth).glob("*/")
        case _:
            contents = []

    for path in contents:
        depth = len(path.relative_to(pth).parts)
        if depth <= level or level == -1:
            spacer = "-" * depth
            print(f">{spacer}{path.name}")


if __name__ == "__main__":
    tree()
```

Рисунок 18 – Код программы индивидуального задания 2

```
(venv) PS C:\git\lab2.19> py indiv2_tree.py --help
Usage: indiv2_tree.py [OPTIONS]

Options:
  --pth TEXT        Директория
  --level INTEGER   Уровень вложенности
  --option INTEGER  1 - Показать файлы, папки и подпапки, 2 - Показать папки и
                    подпапки, 3 - Показать только папки.
  --help            Show this message and exit.
(venv) PS C:\git\lab2.19> py indiv2_tree.py --pth C:\Folder --level 1 --option 1
C:\Folder
>-Subfolder
(venv) PS C:\git\lab2.19> py indiv2_tree.py --pth C:\Folder --level 1 --option 2
C:\Folder
>Folder
>-Subfolder
(venv) PS C:\git\lab2.19> py indiv2_tree.py --pth C:\Folder --level 1 --option 3
C:\Folder
>-Subfolder
(venv) PS C:\git\lab2.19> py indiv2_tree.py --pth C:\Folder --level 2 --option 1
C:\Folder
>-Subfolder
>--FileInSubfolder.txt
>--SubSubFolder
(venv) PS C:\git\lab2.19> py indiv2_tree.py --pth C:\Folder --level 2 --option 2
C:\Folder
>Folder
>-Subfolder
>--SubSubFolder
(venv) PS C:\git\lab2.19> py indiv2_tree.py --pth C:\Folder --level 2 --option 3
C:\Folder
>-Subfolder
(venv) PS C:\git\lab2.19>
```

Рисунок 19 – Результат выполнения индивидуального задания 2

Вывод: В результате выполнения работы был изучен модуль pathlib языка Python и его основные функции для работы с файловой системой.

Контрольные вопросы:

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

Через модуль os.path – os.path.isfile(os.path.join(os.path.expanduser('~'), 'realpython.txt')) или с помощью методов строк path.rsplit('\\', maxsplit=1)[0].

2. Что регламентирует PEP 428?

PEP 428 - "The pathlib module - representing file system paths as objects" (Модуль pathlib — объектно-ориентированный подход к путям в файловых системах).

Цель этой библиотеки - предоставить простую иерархию классов для работы с путями файловой системы и обычными операциями, которые пользователи выполняют над ними.

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):

```
import pathlib
pathlib.Path.cwd()
PosixPath('/home/gahjelle/realpython/')
```

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

Чтобы получить путь дочернего элемента файловой системы с помощью модуля pathlib в Python, можно использовать метод joinpath().

Допустим, есть объект Path, представляющий путь к родительскому каталогу, и вы хотите получить путь к дочернему элементу child_dir в этом каталоге. Для этого можно вызвать метод joinpath() на объекте Path, передав в него имя дочернего элемента в качестве аргумента.

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

Для получения пути к родительским элементам файловой системы с помощью модуля pathlib в Python, можно использовать атрибут parent объекта

Path. Этот атрибут возвращает объект Path, представляющий родительский каталог текущего элемента.

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

```
# создание файла
file_path = Path('/path/to/myfile.txt')
file_path.touch()
# чтение содержимого файла
file_path = Path('/path/to/myfile.txt')
with file_path.open() as f:
contents = f.read()
# запись в файл
file_path = Path('/path/to/myfile.txt')
with file_path.open(mode='w') as f:
f.write('Hello, world!')
# переименование файла
file_path = Path('/path/to/myfile.txt')
new_file_path = Path('/path/to/newfile.txt')
file_path.rename(new_file_path)
# удаление файла
file_path = Path('/path/to/myfile.txt')
file_path.unlink()
```

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

Модуль pathlib в Python позволяет выделить различные компоненты пути файловой системы, такие как имя файла, расширение файла, родительский каталог и т.д.

```
# получение имени файла
file_path = Path('/path/to/myfile.txt')
file_name = file_path.name
print(file_name) # 'myfile.txt'
# получение расширения файла
file_path = Path('/path/to/myfile.txt')
file_ext = file_path.suffix
print(file_ext) # '.txt'
# получение родительского каталога
file_path = Path('/path/to/myfile.txt')
parent_dir = file_path.parent
print(parent_dir) # '/path/to'
# получение всех компонентов пути
file_path = Path('/path/to/myfile.txt')
components = file_path.parts
print(components) # ('/', 'path', 'to', 'myfile.txt')
```

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.

9. Как выполнить подсчет файлов в файловой системе?

Есть несколько разных способов перечислить много файлов. Самым простым является метод. iterdir() , который перебирает все файлы в данном каталоге. Более гибкие списки файлов могут быть созданы с помощью методов .glob() и .rglob() (рекурсивный глоб).

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?

Сначала укажите шаблон для имени файла с местом для счетчика. Затем проверьте существование пути к файлу, созданного путем соединения каталога и имени файла (со значением счетчика). Если он уже существует, увеличьте счетчик и попробуйте снова:

```
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:

```
>>> pathlib.WindowsPath('test.md')
```

В некоторых случаях может потребоваться представление пути без доступа к базовой файловой системе (в этом случае также может иметь смысл представлять путь Windows в системе, отличной от Windows, или наоборот). Это можно сделать с помощью объектов PurePath:

```
>>> path = pathlib.PureWindowsPath(r'C:\Users\gahjelle\realpython\file.txt')
>>> path.name
'file.txt'
>>> path.parent
PureWindowsPath('C:/Users/gahjelle/realpython')
>>> path.exists()
AttributeError: 'PureWindowsPath' object has no attribute 'exists'
```

Можно напрямую создать экземпляр PureWindowsPath или PurePosixPath во всех системах. Создание экземпляра PurePath вернет один из этих объектов в зависимости от используемой операционной системы.