

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

Кафедра инфокоммуникаций

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

Отчет по лабораторной работе № 2.19
по дисциплине «Программирование на Python»

Выполнил студент группы ИВТ-б-о-21-1

Уланбекова Айканыш.

« » _____ 2022г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

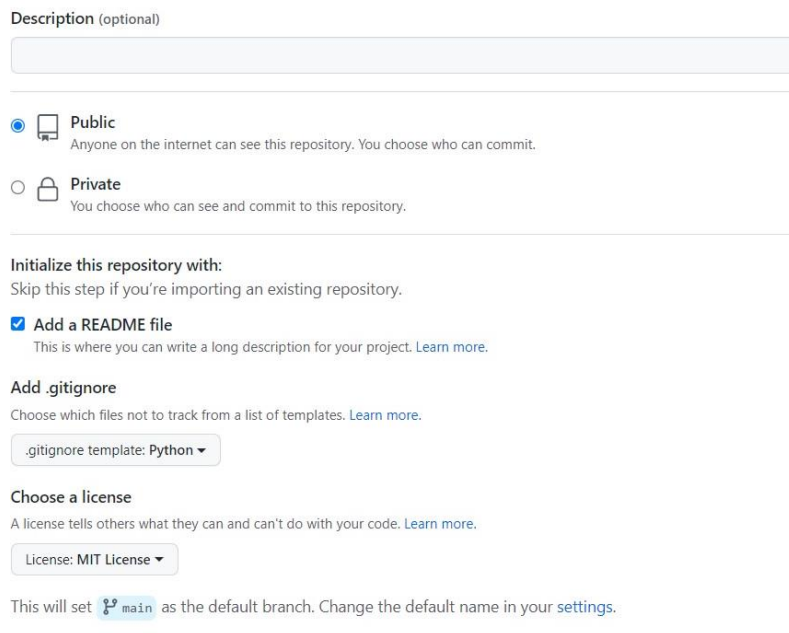
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

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

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.



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:
Skip this step if you're importing an existing repository.

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

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

.gitignore template: Python ▼

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

License: MIT License ▼

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

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория.

```
C:\Users\User>cd C:\Users\User\Desktop\2 кypc Python\lab 22
C:\Users\User\Desktop\2 кypc Python\lab 22>git clone https://github.com/aikanyshkaukanbekova/2.19.git
Cloning into '2.19'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 4.51 KiB | 1.13 MiB/s, done.
Resolving deltas: 100% (1/1), done.
C:\Users\User\Desktop\2 кypc Python\lab 22>
```

Рисунок 2 - Клонирование репозитория

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

```

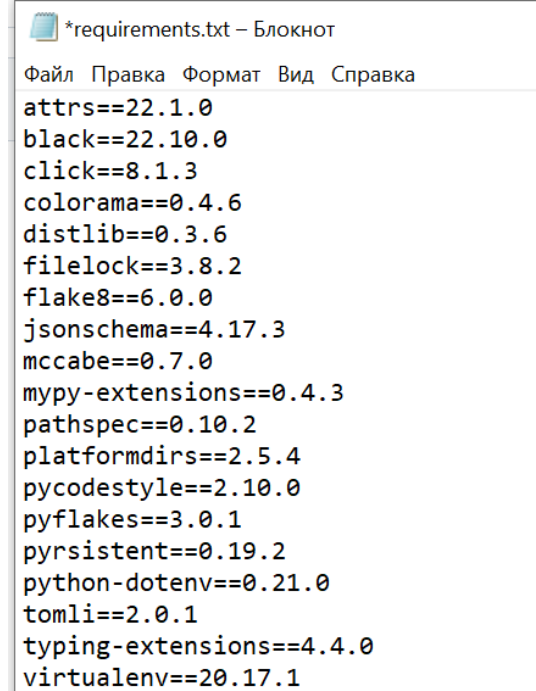
C:\Users\User\Desktop\2 кypc Python\lab 10\lab-10>git flow init
which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

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/User/Desktop/2 кypc Python/lab 10/lab-10/.git/hooks]
C:\Users\User\Desktop\2 кypc Python\lab 10\lab-10>

```

Рисунок 3 - Ветвление по модели git-flow

4. Формирование файла requirements.txt.



```

attrs==22.1.0
black==22.10.0
click==8.1.3
colorama==0.4.6
distlib==0.3.6
filelock==3.8.2
flake8==6.0.0
jsonschema==4.17.3
mccabe==0.7.0
mypy-extensions==0.4.3
pathspec==0.10.2
platformdirs==2.5.4
pycodestyle==2.10.0
pyflakes==3.0.1
pyrsistent==0.19.2
python-dotenv==0.21.0
tomli==2.0.1
typing-extensions==4.4.0
virtualenv==20.17.1

```

Рисунок 4 - Файл requirements.txt

5. Проработать примеры лабораторной работы.

```

#!/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()))

```

Рисунок 5 – Результат выполнения примера 1

```

#!/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__":
    tree(pathlib.Path.cwd())
```

Рисунок 6 - Результат выполнения примера 2

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

import pathlib
from datetime import datetime

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

Рисунок 7 - Результат выполнения примера 3

```
#!/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 8 - Результат выполнения примера 4

6. Выполнить индивидуальные задания.

Задание 1

Для своего варианта лабораторной работы 2.17 добавьте возможность получения имени файла данных, используя соответствующую переменную окружения.

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

import argparse
import json
import os.path
import sys

def add_route(routes, start, finish, number):
    """
    Добавить данные о маршруте
    """
```

```

        routes.append(
            {
                'start': start,
                'finish': finish,
                'number': number
            }
        )
    return routes

def display_route(routes):
    """
    Отобразить список маршрутов
    """
    if routes:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Начальный пункт",
                "Конечный пункт",
                "Номер маршрута"
            )
        )
        print(line)

        # Вывести данные о всех рейсах.
        for idx, worker in enumerate(routes, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('start', ''),
                    worker.get('finish', ''),
                    worker.get('number', 0)
                )
            )
            print(line)
    else:
        print("Список маршрутов пуст.")

def select_route(routes, period):
    """
    Выбрать маршрут
    """
    result = []
    for employee in routes:
        if employee.get('number') == period:
            result.append(employee)

    return result

def save_routes(file_name, routes):
    """
    Сохранить всех работников в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(routes, fout, ensure_ascii=False, indent=4)
        directory = pathlib.Path.cwd().joinpath(file_name)
        directory.replace(pathlib.Path.home().joinpath(file_name))

```

```

def load_routes(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main(command_line=None):
    # Создать родительский парсер для определения имени файла.
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "-d",
        "--data",
        action="store",
        required=False,
        help="The data file name"
    )
    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("routes")
    parser.add_argument(
        "--version",
        action="version",
        version="% (prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    # Создать субпарсер для добавления маршрута.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new route"
    )
    add.add_argument(
        "-s",
        "--start",
        action="store",
        required=True,
        help="The start of the route"
    )
    add.add_argument(
        "-f",
        "--finish",
        action="store",
        help="The finish of the route"
    )
    add.add_argument(
        "-n",
        "--number",
        action="store",
        type=int,
        required=True,
        help="The number of the route"
    )
    # Создать субпарсер для отображения всех маршрутов.
    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all routes"
    )
    # Создать субпарсер для выбора маршрута.
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the route"
    )
    select.add_argument(
        "-N",

```

```

        "--numb",
        action="store",
        type=int,
        required=True,
        help="The route"
    )
    # Выполнить разбор аргументов командной строки.
    args = parser.parse_args(command_line)
    # Загрузить все маршруты из файла, если файл существует.
    data_file = args.data
    if not data_file:
        data_file = os.environ.get("WORKERS_DATA")
    if not data_file:
        print("The data file name is absent", file=sys.stderr)
        sys.exit(1)
    # Загрузить всех работников из файла, если файл существует.
    is_dirty = False
    if os.path.exists(data_file):
        routes = load_routes(data_file)
    else:
        routes = []
    # Добавить маршрут.
    if args.command == "add":
        routes = add_route(
            routes,
            args.start,
            args.finish,
            args.number
        )
        is_dirty = True
    # Отобразить все маршруты.
    elif args.command == "display":
        display_route(routes)
    # Выбрать требуемые маршруты.
    elif args.command == "select":
        selected = select_route(routes, args.numb)
        display_route(selected)
    # Сохранить данные в файл, если список маршрутов был изменен.
    if is_dirty:
        save_routes(data_file, routes)

if __name__ == '__main__':
    main()

```

Рисунок 9 - Результат выполнения задания 1

Задание 2

Самостоятельно изучите работу с пакетом `python-dotenv`. Модифицируйте программу задания 1 таким образом, чтобы значения необходимых переменных окружения считывались из файла `.env`.

```

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

import argparse
import pathlib
import colorama
from colorama import Fore

def tree(directory):
    print(Fore.BLUE + f'>>> {directory}')

```

```

for path in sorted(directory.rglob('*')):
    print(Fore.YELLOW + f' >> {path.name}')
    for new_path in sorted(directory.joinpath(path).glob('*')):
        print(Fore.GREEN + f' > {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",
        help="The main parser",
        version="%(prog)s 0.1.0"
    )

    subparsers = parser.add_subparsers(dest="command")

    # Субпарсер для создания папок
    create = subparsers.add_parser(
        "mkfl",
        parents=[file_parser]
    )
    create.add_argument(
        "filename",
        action="store"
    )

    # Субпарсер для удаления папок
    create = subparsers.add_parser(
        "rmfl",
        parents=[file_parser]
    )
    create.add_argument(
        "filename",
        action="store"
    )

    args = parser.parse_args(command_line)

    if args.command == 'mkfl':
        directory_path = current / args.filename
        directory_path.mkdir()
        tree(current)
    elif args.command == "rmfl":
        directory_path = current / args.filename
        directory_path.rmdir()
        tree(current)
    else:
        tree(current)

if __name__ == "__main__":
    main()

```

Рисунок 10 - Результат выполнения задания 2

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

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