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

Пуценко И.А. « » 2023г.

Подпись студента_____

Работа защищена « »_____2023г.

Проверил Воронкин Р.А. _____

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

Выполнение работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработать примеры лабораторной работы. Создайте для

8. 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.

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

import json
import os
import jsonschema
from jsonschema import validate
import argparse
from pathlib import Path


def load_data():
    data = []
    data_path = Path(data_file)
    if data_path.exists():
        with open(data_path, "r") as file:
            data = json.load(file)
            validate(data, schema)
    return data


def save_data(data):
    validate(data, schema)
    with open(data_file, "w") as file:
        json.dump(data, file, indent=4)


def exit_to_program():
    print('всего доброго')
    save_data(lst_planes)
    return exit(1)


def help_program():
    print("add - добавление рейса\n"
          "help - помощь по командам\n"
          "select \"пункт назначения\" - вывод самолетов летящих в п.н.\n"
          "display_plane - вывод всех самолетов\n"
          "exit - выход из программы")


def add_program(planes):
    plane = dict()
    plane["destination"] = input("Пункт назначения:\n")
    plane["flight_number"] = int(input("Номер рейса:\n"))
    plane["type_plane"] = input("Тип самолета\n")
    planes.append(plane)
    planes.sort(key=lambda key_plane: key_plane.get("flight_number"))
    return planes


def select_program(planes):
    lst = list(map(lambda x: x.get("destination"), planes))
    point = input('выберите нужное вам место\n')
    print("результаты поиска")
    if point in lst:
        print('рейсы в эту точку')
        for i in planes:
            if point == i["destination"]:
                print(f"{i['flight_number']}.....{i['type_plane']}")
    else:
        print("рейсов не найдено")


def error():
    print('неверная комманда')
```

```
def display_plane(staff):
    if staff:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
                '-' * 4,
                '-' * 30,
                '-' * 20,
                '-' * 8
            )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "№",
                "Направление",
                "Тип самолета",
                "рейс"
            )
        )
        print(line)

        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('destination', ''),
                    worker.get('type_plane', ''),
                    worker.get('flight_number', 0)
                )
            )
            print(line)

    else:
        print("рейсов не найдено")


def menu(lst_plane):
    command = input('введите команду("help" - руководство по командам)\n>>>').lower()
    if command == 'exit':
        exit_to_program()
    elif command == 'help':
        help_program()
    elif command == 'add':
        lst_plane = add_program(lst_plane)
    elif command == 'select':
        select_program(lst_plane)
    elif command == 'display_plane':
        display_plane(lst_plane)
    else:
        error()


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Программа управления рейсами
самолетов')
    parser.add_argument('--file', help='Имя файла JSON для сохранения и чтения данных')
    args = parser.parse_args()

    data_file = args.file if args.file else input("Введите имя файла данных: ")

    # Определение пути к файлу данных в домашнем каталоге пользователя
    home_dir = str(Path.home())
    data_file = os.path.join(home_dir, data_file)

    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "destination": {"type": "string"},
```

```
                "flight_number": {"type": "integer"},
                "type_plane": {"type": "string"}
            },
            "required": ["destination", "flight_number", "type_plane"]
        }
    }

    lst_planes = load_data()

    while True:
        menu(lst_planes)
```

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.

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

import os
import argparse


def tree(directory, max_depth, dir_only, file_ext):
    print_directory(directory, max_depth, dir_only, file_ext, 0)


def print_directory(directory, max_depth, dir_only, file_ext, depth):
    if depth > max_depth:
        return

    prefix = "|   " * depth
    entries = os.scandir(directory)

    for entry in sorted(entries, key=lambda e: e.name):
        if entry.is_dir() and not entry.is_symlink():
            print(prefix + "|-- " + entry.name)
            if not dir_only:
                print_directory(entry.path, max_depth, dir_only, file_ext, depth + 1)
        elif not dir_only and entry.is_file():
            if not file_ext or entry.name.endswith(file_ext):
                print(prefix + "|-- " + entry.name)


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Analog of the 'tree' utility in
Linux.")
    parser.add_argument("directory", nargs="?", default=".", help="Target directory
(default: current directory)")
    parser.add_argument("--level", type=int, default=float("inf"), help="Maximum depth of
tree traversal")
    parser.add_argument("--dir-only", action="store_true", help="Display directories
only")
    parser.add_argument("--file-ext", help="Filter files by extension")

    args = parser.parse_args()

    directory = args.directory
    max_depth = args.level
    dir_only = args.dir_only
    file_ext = args.file_ext

    tree(directory, max_depth, dir_only, file_ext)
```

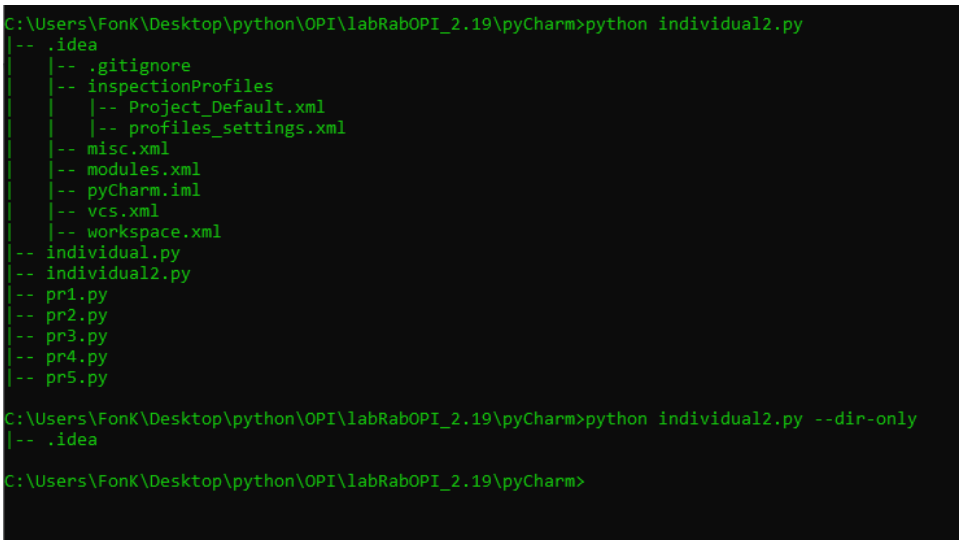

Рисунок 9 – Результат работы программы

10. Зафиксируйте сделанные изменения в репозитории.

Вопросы для защиты работы:

1. Какие существовали средства для работыс файловой системой до

Pytho 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 использование модуля pathlib в Python.

3. Как осуществляется создание путей средствами модуля pathlib?

Создание путей средствами модуля pathlib осуществляется с помощью класса pathlib.PathПрежде всего, существуют classmethods наподобие. cwd() (текущий рабочий каталог) и .home() (домашний каталог вашего пользователя). Путь также может быть явно создан из его строкового представления.

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, у вас 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?

Он позволяет выполнять операции с файлами с помощью объектов Path, которые представляют пути файловой системы.

```
from pathlib import Path

# 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:

```
# запись в файл file_path = Path('/path/to/myfile.txt') with file_path.open(mode='w') as f:     f.write('Hello, world!')
```

Переименование файла:

```
# 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.

```
from pathlib import Path

# 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?

Модуль pathlib также предоставляет методы для перемещения и удаления каталогов, такие как Path.rename() и Path.rmdir(). 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?

Для отображения дерева каталогов файловой системы в Python можно использовать модуль pathlib и рекурсивную функцию. 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.

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.

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