Министерство науки и высшего образования Российской Федерации

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №22
дисциплины «Основы программной инженерии»

Выполнил:
Плугатырев Владислав Алексеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил Воронкин Роман Александрович
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Работа с файловой системе в Python3 с использованием модуля pathlib.

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

Ход работы.

1. Создание репозитория.

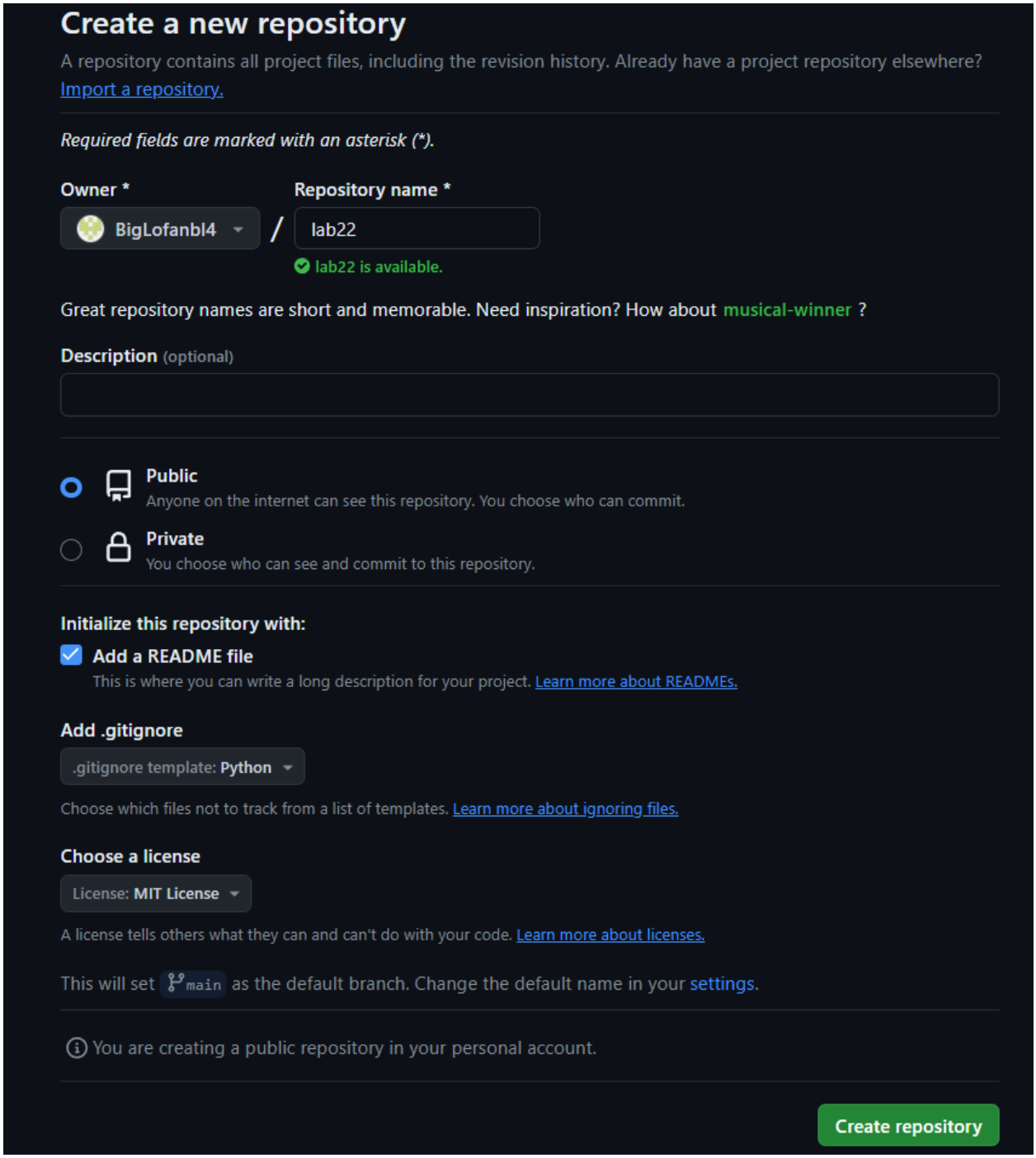

Рисунок 1.1 – Создание репозитория

2. Выполнение примеров из лабораторной работы.

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


import collections
import pathlib


if __name__ == "__main__":
    print(
        collections.Counter(
            p.suffix for p in pathlib.Path.cwd().parent.iterdir()
        )
    )
```

Рисунок 2.1 – Код первого примера

```
#!/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().parent)
```

Рисунок 2.2 – Код второго примера

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


import pathlib
from datetime import datetime


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

Рисунок 2.3 – Код третьего примера

```python
#!/usr/bin/env python
# -*- 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().parent, "test_{:03d}.txt")
    print(path)
```

Рисунок 2.4 – Код четвертого примера

3. Выполнение первого задания: 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.

```python
# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)

# Домашний каталог
home_path = Path.home()/args.filename

is_dirty = False
if home_path.exists():
    people = load_people(args.filename)
else:
    people = []

match args.command:
    case "add":
        people = add_person(
            people, args.surname, args.name, args.zodiac, args.birthday
        )
        people.sort(
            key=lambda x: datetime.strptime(
                ".".join(x["birthday"]), "%d.%m.%Y"
            )
        )
        is_dirty = True

    case "select":
        selected = select_people(args.surname, people)
        display_people(selected)

    case "display":
        display_people(people)

if is_dirty:
    save_people(home_path, people)
```

Рисунок 3.1 – Код программы

```
lab22) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\22\lab22>python ind1.py add data.json -s="Plugatyrev" -n="Vladislav" -z="Capricorn" -b="12.01.2005"
lab22) C:\Users\vladi\OneDrive\Рабочий стол\Основы программной инженерии\22\lab22>
```

Рисунок 3.2 – Запуск программы

Рисунок 3.3 – Полученный файл

4. Выполнение второго задания: 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.

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

import argparse
from pathlib import Path


def tree(directory, spacer="", level=1, directory_only=False):
    if level == 0:
        return

    files = list(Path(directory).iterdir())

    for index, file in enumerate(files):
        connector = "└── " if index == len(files) - 1 else "├──"

        if file.is_dir():
            print(spacer + connector + file.name)
            tree(file, spacer + "|    ", level - 1, directory_only)
        elif file.is_file() and not directory_only:
            print(spacer + connector + file.name)


if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Display directory tree structure"
    )
    parser.add_argument(
        "directory",
        nargs="?",
        type=str,
        default=".",
        help="The root directory path",
    )
    parser.add_argument(
        "-l",
        "--level",
        type=int,
        default=-1,
        help="Max display depth of the directory tree",
    )
    parser.add_argument(
        "-d",
        "--directory-only",
        action="store_true",
        help="Show directories only",
    )

    args = parser.parse_args()
    tree(args.directory, level=args.level, directory_only=args.directory_only)
```

Рисунок 4.1 – Код программы

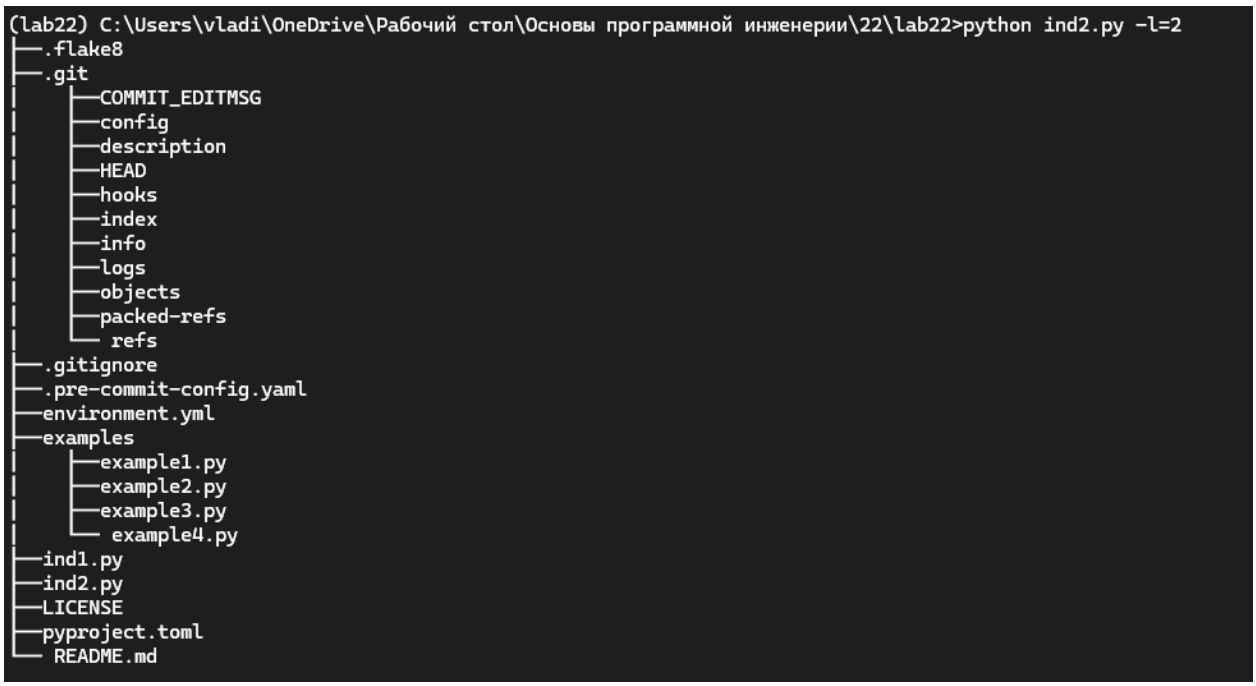

Рисунок 4.2 – Работа программы

Ответы на контрольные вопросы

1. Средства для работы с файловой системой до Python 3.4:

До введения модуля «pathlib» в стандартную библиотеку Python 3.4, за работу с файлами и путями файловой системы отвечали в основном модули «os», «os.path» и «shutil». Была также библиотека «glob» для шаблонов поиска файлов, а для более высокоуровневых операций файла и директорий использовались «io» или встроенные функции, такие как «open».

2. PEP 428:

PEP 428, также известный как "The pathlib module – object-oriented filesystem paths", регламентирует добавление модуля «pathlib» в стандартную библиотеку Python. Этот PEP описывает объектно-ориентированный подход к работе с путями файловой системы.

3. Создание путей с помощью модуля «pathlib»:

Для создания путей с помощью «pathlib» вы создаете экземпляры классов «Path» или его подклассов. Например: «p = Path('/usr/local/bin')».

4. Получение пути дочернего элемента:

Путь к дочернему элементу можно получить, используя оператор деления («/»), предусмотренный «pathlib»: «child_path = parent_path / 'child'«.

5. Получение путей родительских элементов:

Чтобы получить родительский путь, вы можете использовать атрибут «.parent» или метод «.parents[]» объекта Path: «parent_path = my_path.parent» или «grandparent_path = my_path.parents[1]».

6. Операции с файлами в «pathlib»:

Модуль «pathlib» позволяет выполнять различные операции с файлами, такие как открытие файла («open»), проверку существования («exists»), удаление файла («unlink»), создание директорий («mkdir»), переименование или перемещение («rename» или «replace»).

7. Выделение компонентов пути:

Используя методы «parts», «name», «suffix», «stem» объекта Path, вы можете извлечь различные компоненты пути.

8. Перемещение и удаление файлов:

Для перемещения файлов можно использовать метод «rename()», а для удаления файлов — метод «unlink()». Для каталогов есть метод «rmdir()», который удаляет пустые директории.

9. Подсчет файлов в каталоге:

Для подсчёта файлов можно использовать, например, конструкцию «len(list(Path('path/to/dir').iterdir()))».

10. Отображение дерева каталогов:

Чтобы отобразить дерево каталогов, можно использовать рекурсивный обход в комбинации с печатью имен файлов и директорий, сформированной в виде структуры дерева.

11. Создание уникального имени файла:

Вы можете использовать модуль «tempfile», чтобы создать уникальное имя файла, например, «tempfile.NamedTemporaryFile(delete=False)».

12. Отличия в использовании модуля

«pathlib» для различных операционных систем: «pathlib» имеет различные реализации для разных операционных систем (WindowsPath для Windows и PosixPath для Unix), что позволяет ему обрабатывать системно-специфические различия в путях. Однако интерфейс класса Path большей частью единообразен и позволяет писать кросс-платформенный код для работы с файловой системой.