

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 №5**  
**дисциплины «анализ данных»**

Выполнил:  
Середа Кирилл Витальевич  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Роман Александрович

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

Тема: Работа с файловой системой в Python 3 с использованием модуля pathlib.

Цель: приобрести навыки работы с файловой системой в Python 3.x с использованием модуля pathlib.

### Ход выполнения заданий

#### 1) Выполнил первое 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 .

```
(DA_5) MelancholySeal@Kiras-MacBook-Air prog % python ind1.py add -f people.json -n "Иван Иванов" -b 1990-01-01 -p 1234567890
Файл people.json не существует.
(DA_5) MelancholySeal@Kiras-MacBook-Air prog % python ind1.py find -f people.json -p 1234567890
Ф.И.О.: Иван Иванов
Дата рождения: 1990-01-01
Телефон: 1234567890
(DA_5) MelancholySeal@Kiras-MacBook-Air prog % python ind1.py display -f people.json
+-----+-----+-----+
| Ф.И.О. | Дата рождения | Телефон |
+-----+-----+-----+
| Иван Иванов | 1990-01-01 | 1234567890 |
+-----+-----+-----+
(DA_5) MelancholySeal@Kiras-MacBook-Air prog %
```

Рисунок 1 – Вывод первого задания

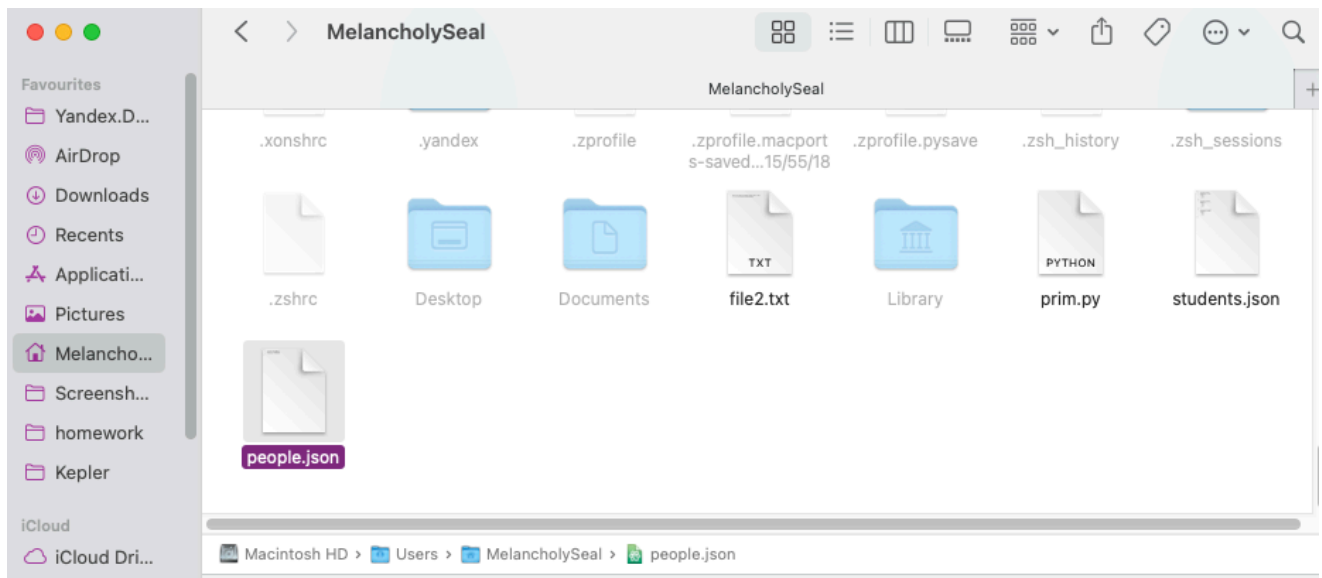


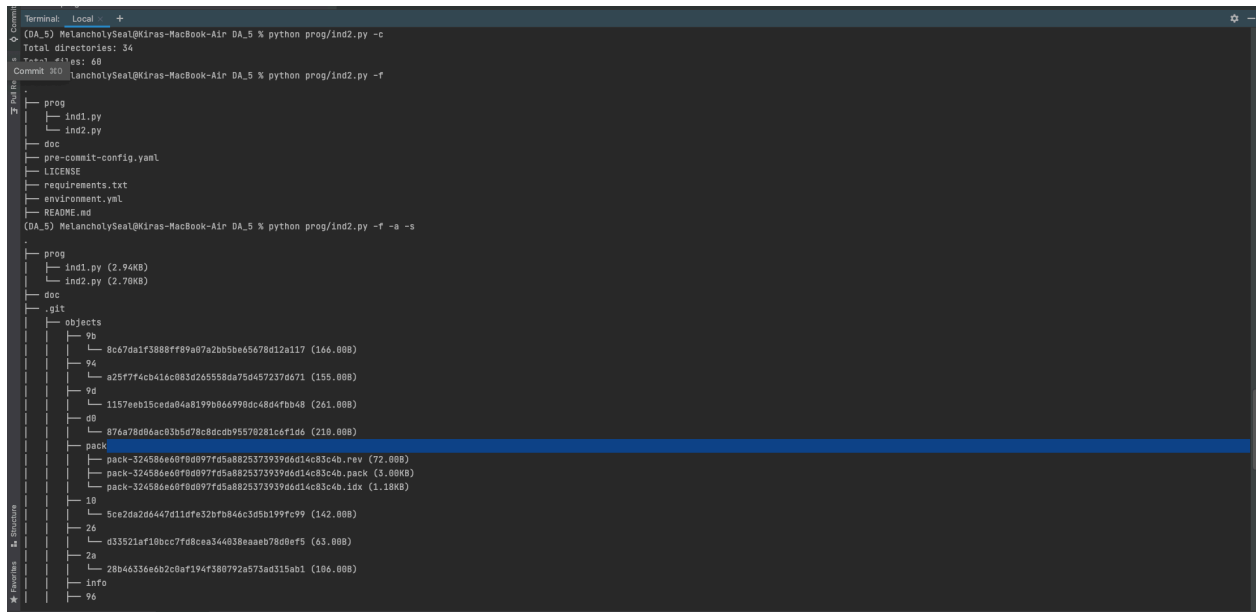
Рисунок 2 – файл в домашнем каталоге

#### 2) Выполнил второе задание

##### 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.



```
Terminal: Local +
(DA_5) MelancholySeal@Kiras-MacBook-Air DA_5 % python prog/ind2.py -c
Total directories: 34
Total files: 60
Commit 900 MelancholySeal@Kiras-MacBook-Air DA_5 % python prog/ind2.py -f
prog
├── ind1.py
├── ind2.py
├── doc
├── pre-commit-config.yaml
├── LICENSE
├── requirements.txt
├── environment.yml
├── README.ad
└── (DA_5) MelancholySeal@Kiras-MacBook-Air DA_5 % python prog/ind2.py -f -a -s
prog
├── ind1.py (2.94KB)
├── ind2.py (2.70KB)
├── doc
├── .git
├── objects
├── 9b
├── 8c67da1f3888ff89a97a2bb5be65678d12a117 (166.00B)
├── 94
├── a25f774cb41c083d265558da75d457237d671 (155.00B)
├── 9d
├── 1157eeb15ceda94a8199b866998dc48d4fbb48 (261.00B)
├── 0b
├── 876e78d06ac83b5d78cdcd995578281c6f1d6 (210.00B)
├── pack
├── pack-32458ae60f8d977fda8825373939d6d14c83c4b.rev (72.00B)
├── pack-32458ae60f8d977fda8825373939d6d14c83c4b.pack (3.00KB)
├── pack-32458ae60f8d977fda8825373939d6d14c83c4b.idx (1.18KB)
├── 1b
├── 5ce2da2d6447d11dfe32bf8b84ec3d9b199fc99 (142.00B)
├── 26
├── d13521af18bcc7fd8cea344038aeeb78d8ef5 (63.00B)
├── 2a
├── 28b4633ae6b2c0af194f380792a573ad315ab1 (186.00B)
├── info
├── 96
```

Рисунок 3 – Вывод задания

## Ответы на вопросы

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

До появления **pathlib** в Python 3.4, работа с файловой системой осуществлялась с помощью следующих модулей:

- **os**: Предоставлял функции для работы с путями (**os.path**), создания и удаления директорий и файлов, получения информации о файлах и директориях.
- **os.path**: Содержал функции для манипулирования путями (например, **os.path.join**, **os.path.basename**, **os.path.dirname**).
- **shutil**: Использовался для высокоуровневых операций с файлами и директориями, таких как копирование, перемещение и удаление (**shutil.copy**, **shutil.move**, **shutil.rmtree**).
- **glob**: Для поиска файлов и директорий по шаблону.
- **fnmatch**: Для проверки соответствия имени файла шаблону.

- **stat**: Для получения информации о состоянии файлов (например, прав доступа).

## 2. Что регламентирует PEP 428?

PEP 428 регламентирует создание модуля **pathlib**, который предоставляет объектно-ориентированный интерфейс для работы с путями файловой системы. Он предлагает унифицированный способ представления и манипуляции файловыми путями, независимо от операционной системы. **pathlib** делает код более читаемым и легким для написания, поскольку объекты пути имеют методы, которые интуитивно понятны и хорошо документированы.

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

Создание путей осуществляется с помощью класса **Path** из модуля **pathlib**:

```
from pathlib import Path
# Создание пути к файлу или директории
path = Path('/some/directory/file.txt')
```

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

Для получения пути дочернего элемента используйте оператор **/** или метод **joinpath**:

```
from pathlib import Path
# Создание базового пути
base_path = Path('/some/directory')
# Получение дочернего элемента
child_path = base_path / 'subdirectory' / 'file.txt'
# Или с использованием joinpath
child_path = base_path.joinpath('subdirectory', 'file.txt')
```

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

Для получения пути к родителям используйте атрибут **parent**:

```
from pathlib
import Path

# Создание пути
path = Path('/some/directory/file.txt')

# Получение родительского пути
parent_path = path.parent

# Для получения более высокого уровня родителя
grandparent_path = path.parent.parent
```

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

Операции с файлами включают чтение, запись, удаление и другие действия. Примеры:

```
from pathlib
import Path

# Создание пути к файлу
file_path = Path('/some/directory/file.txt')

# Чтение файла content = file_path.read_text()

# Запись в файл
file_path.write_text('Hello, World!')

# Проверка существования файла
exists = file_path.exists()

# Удаление файла
file_path.unlink()
```

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

Выделение компонентов пути осуществляется с помощью различных методов и атрибутов:

```
from pathlib
import Path
# Создание пути
path = Path('/some/directory/file.txt')
# Получение имени файла
file_name = path.name
# Получение расширения файла
file_extension = path.suffix
# Получение имени файла без расширения
file_stem = path.stem
# Получение всех частей пути
parts = path.parts
```

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

Перемещение и удаление файлов осуществляется с помощью методов **`rename`** и **`unlink`**:

```
from pathlib
import Path
# Создание путей
source = Path('/some/directory/file.txt')
destination = Path('/new/directory/file.txt') #
Перемещение файла
source.rename(destination)
# Удаление файла
destination.unlink()
```

## **9. Как выполнить подсчет файлов в файловой системе?**

Подсчет файлов можно выполнить с помощью метода **`rglob`** для рекурсивного поиска и подсчета файлов:

```
from pathlib
import Path
```

```
# Создание пути к директории
directory = Path('/some/directory')

# Подсчет всех файлов в директории и поддиректориях
file_count = sum(1 for _ in directory.rglob('*')
if _.is_file())
print(f'Total files: {file_count}')
```

## 10. Как отобразить дерево каталогов файловой системы?

Для отображения дерева каталогов используйте рекурсивный обход директорий и файлов с помощью **pathlib**:

```
from pathlib
import Path

def display_tree(directory, prefix=""):
    print(prefix + directory.name)
    prefix += ' | ' for path in sorted(directory.iterdir(), key=lambda p:
(p.is_file(), p.name.lower())):
        if path.is_dir(): display_tree(path, prefix)
        else:
            print(prefix + path.name)

root = Path('/some/directory')
display_tree(root)
```

## 11. Как создать уникальное имя файла?

Создание уникального имени файла можно сделать с использованием модуля **uuid**:

```
from pathlib
import Path import uuid

# Создание уникального имени файла
unique_name = f'{uuid.uuid4()}.txt'
unique_path = Path('/some/directory') / unique_name
```

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

**pathlib** абстрагирует различия между файловыми системами, предоставляя унифицированный интерфейс. Однако, некоторые различия:

- **Пути:** Используются POSIX (Unix-подобные системы) и Windows пути, например, **Path('/some/path')** на Unix и **Path('C:\\some\\path')** на Windows.
- **Разделители:** POSIX использует /, Windows - \\. **pathlib** корректно обрабатывает это.
- **Методы:** Некоторые методы и атрибуты могут вести себя по-разному. Например, символические ссылки (**symlinks**) и их обработка могут различаться.

Пример кроссплатформенного кода:

```
from pathlib
import Path

# Создание пути с использованием
Path path = Path.home() / 'some' / 'directory'

# Обработка символических ссылок
if path.is_symlink(): real_path = path.resolve()
```

Вывод: в ходе выполнения лабораторной работы, приобретены навыки работы с файловой системой в Python 3.x с использованием модуля **pathlib**.