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

```
(DA_5) MelancholySeal@Kiras-MacBook-Air prog % python ind1.py add -f students.json -n='Анатолий' -g='131' --grades="5 5 4 5"
(DA_5) MelancholySeal@Kiras-MacBook-Air prog % python ind1.py display -f students.json
+-------------------------------+----------------+----------------------+
|            Ф.И.О.             |  Номер группы  |     Успеваемость     |
+-------------------------------+----------------+----------------------+
| =Гена                         | =111           | 5.0, 5.0, 5.0, 5.0   |
| =Гена                         | =111           | 5.0, 5.0, 5.0, 5.0   |
| =Анатолий                     | =131           | 5.0, 5.0, 4.0, 5.0   |
+-------------------------------+----------------+----------------------+
(DA_5) MelancholySeal@Kiras-MacBook-Air prog %
```

Рисунок 1 – Вывод первого задания

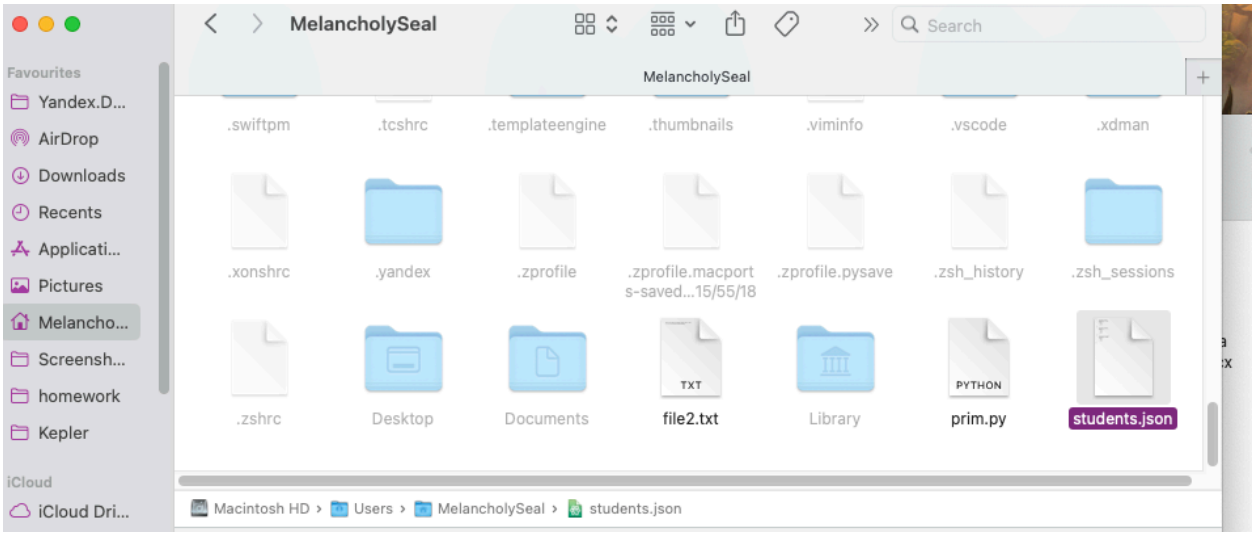

Рисунок 2 – файл в домашнем каталоге

2) Выполнил второе задание

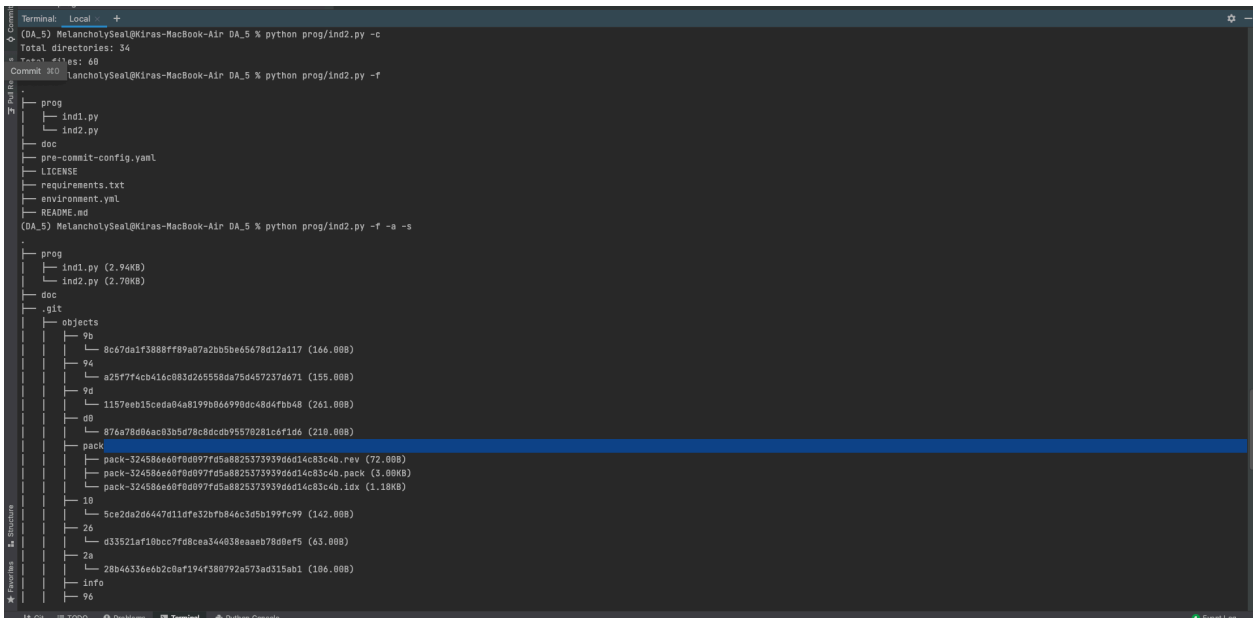

Рисунок 3 – Вывод задания

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 i
mport 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.