

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 ПРАКТИЧЕСКОЙ РАБОТЕ №5**  
**дисциплины «Анализ данных»**

Выполнил:  
Степанов Леонид Викторович  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение  
средств вычислительной  
техники и автоматизирование  
систем», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., канд. техн. наук,  
доцент, доцент кафедры  
инфокоммуникаций

---

(подпись)

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

Ставрополь, 2024 г.

Тема: Работа с файловой системой в Python3 с использованием модуля pathlib

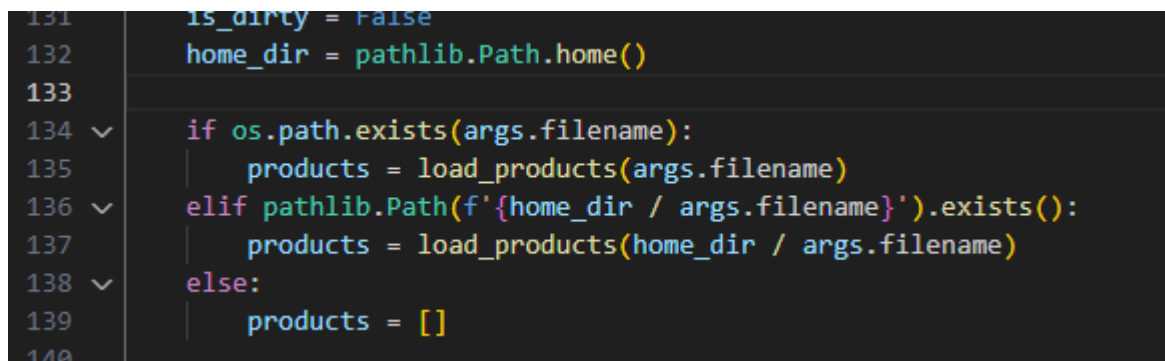
Цель работы: приобретение навыков по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.

Ход работы:

Индивидуальное 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.

В файле ind.py, была реализована возможность хранения файла данных в домашнем каталоге пользователя, фрагмент кода, в котором реализована эта возможность изображен на рис.1



```
131 is_dirty = False
132 home_dir = pathlib.Path.home()
133
134 if os.path.exists(args.filename):
135     products = load_products(args.filename)
136 elif pathlib.Path(f'{home_dir / args.filename}').exists():
137     products = load_products(home_dir / args.filename)
138 else:
139     products = []
140
```

Рисунок 1 – Фрагмент кода в ind.py

Индивидуальное 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.

В файле ind2.py был разработан аналог утилиты tree в Linux на рис.2 представлена справка по разработанной утилите

```
(poetry) D:\Python-labs\lab5\prog>poetry run ind2.py -h
usage: ind2.py [-h] [-l LEVEL] [-d DIR] [-s SHOWSIZE] [-t TIME] [path]

Утилита для отображения дерева каталогов и файлов

positional arguments:
  path                  Путь к каталогу

options:
  -h, --help            show this help message and exit
  -l LEVEL, --level LEVEL
                        Уровень вложенности
  -d DIR, --dir DIR     Показывать только директории
  -s SHOWSIZE, --showsize SHOWSIZE
                        Показать размер файлов
  -t TIME, --time TIME  Показать время изменения

(poetry) D:\Python-labs\lab5\prog>
```

Рисунок 2 – Справка

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

Контрольные вопросы:

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

До версии Python 3.4 основным средством для работы с файловой системой были модули `os` и `os.path`. Они предоставляли функции для работы с путями файловой системы, создания, удаления и переименования файлов и каталогов, а также для работы с атрибутами файлов. Однако, с появлением Python 3.4, был введен модуль `pathlib`, который предоставил более удобный и объектно-ориентированный подход к работе с путями файловой системы.

### 2. PEP 428

PEP 428 регламентирует включение модуля `pathlib` в стандартную библиотеку Python. Этот модуль предоставляет различные классы путей, представляющих пути файловой системы с семантикой, соответствующей различным операционным системам. Целью этого модуля является предоставление простой иерархии классов для обработки путей файловой системы и обычных операций, выполняемых над ними.

### 3. Создание путей средствами модуля `pathlib`

Создание путей средствами модуля `pathlib` осуществляется путем создания экземпляров соответствующих классов, таких как `Path`. Например, для создания пути к файлу можно использовать следующий синтаксис:

```
python
from pathlib import Path
path = Path('каталог/файл.txt')
```

4. Получение пути дочернего элемента файловой системы с помощью модуля `pathlib`

Для получения пути дочернего элемента файловой системы с помощью модуля `pathlib` можно использовать метод `joinpath()`. Например:

```
python
from pathlib import Path
path = Path('родительский_путь')
child_path = path.joinpath('дочерний_элемент')
```

5. Получение пути к родительским элементам файловой системы с помощью модуля `pathlib`

Для получения пути к родительским элементам файловой системы с помощью модуля `pathlib` можно использовать метод `parent`. Например:

```
python
from pathlib import Path
path = Path('путь/к/файлу')
parent_path = path.parent
```

6. Операции с файлами с помощью модуля `pathlib`

Модуль `pathlib` предоставляет удобные методы для выполнения операций с файлами, такие как проверка существования, чтение, запись, удаление и многое другое. Например, для проверки существования файла можно использовать метод `exists()`, а для удаления файла – метод `unlink()`.

7. Выделение компонентов пути файловой системы с помощью модуля `pathlib`

С помощью модуля `pathlib` можно выделить различные компоненты пути, такие как имя файла, расширение файла, имя каталога и другие. Например, для получения имени файла можно использовать метод `name`, а для получения расширения файла – метод `suffix`.

8. Отличия в использовании модуля `pathlib` для различных операционных систем

Модуль `pathlib` предоставляет абстракцию от различий между операционными системами, что позволяет писать переносимый код для работы с файловой системой. Например, он автоматически учитывает различия в разделителях пути между Windows, Unix и другими операционными системами, что делает его использование удобным и унифицированным. Как выполнить перемещение и удаление файлов с помощью модуля `pathlib`?

9. Подсчет файлов в файловой системе

Для выполнения подсчета файлов в файловой системе с помощью модуля `pathlib` можно использовать метод `glob()`, который позволяет получить список файлов, соответствующих заданному шаблону. Например:

```
from pathlib import Path
path = Path('путь/к/каталогу')
file_count = sum(1 for _ in path.glob('**/*') if _.is_file())
```

10. Отображение дерева каталогов файловой системы

Для отображения дерева каталогов файловой системы с помощью модуля `pathlib` можно использовать метод `iterdir()`, который возвращает итератор по содержимому каталога. Например:

```
from pathlib import Path

def print_tree(path, indent=""):
    print(f'{indent}{path.name}/')
    for child in path.iterdir():
        if child.is_dir():
            print_tree(child, indent + ' ')
        else:
```

```
print(f'{indent} {child.name}')
```

```
print_tree(Path('путь/к/каталогу'))
```

### 11. Создание уникального имени файла

Для создания уникального имени файла с помощью модуля `pathlib` можно использовать метод `Path.with_name()`, который возвращает новый путь с указанным именем файла. Например:

```
from pathlib import Path
```

```
path = Path('путь/к/каталогу/файл.txt')
```

```
unique_name = path.with_name('уникальное_имя.txt')
```

### 12. Отличия в использовании модуля `pathlib` для различных операционных систем

Модуль `pathlib` предоставляет абстракцию от различий между операционными системами. Например, он автоматически учитывает различия в разделителях пути между Windows и Unix. Кроме того, он предоставляет классы, такие как `WindowsPath` и `PosixPath`, которые представляют пути в зависимости от операционной системы, что делает его использование удобным и универсальным.