Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 2.19. Работа с файловой системе в Python3 с использованием модуля pathlib»

ОТЧЕТ
по лабораторной работе №22
дисциплины
«Основы программной инженерии»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Лабораторная работа 2.19. 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:

```python
#!/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}')


tree(pathlib.Path.cwd())

```

Рисунок 1 – Пример 1

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

import pathlib
from datetime import datetime

time, file_path = max((f.stat().st_mtime, f) for f in pathlib.Path.cwd().iterdir())
print(datetime.fromtimestamp(time), file_path)

```

Рисунок 2 – Пример 2

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

import pathlib
from datetime import datetime

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')
```

Рисунок 3 – Пример 3

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`.

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

import argparse
import sys
import pathlib
from pathlib import Path
import json
import jsonschema
from jsonschema import validate
from prod_schema import schema


def add_product(products, prod, shop, cost):
    """
    Ввод информации о товарах.
    """
    products.append(
        {
            "product": prod,
            "shop": shop,
            "cost": cost
```

```
        }
    )
    return products


def save_products(path, products):
    """
    Сохранить список всех товаров в формате JSON
    """
    with open(path, "w", encoding="utf-8") as fout:
        json.dump(products, fout, ensure_ascii=False, indent=4)


def load_products(path):
    """
    Загрузить список всех товаров из файла JSON
    """
    with open(path, "r", encoding="utf-8") as fin:
        return json.load(fin)


def product_list(products):
    """
    Вывод списка товаров
    """
    if products:
        line = '+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20
        )
        print(line)
        print(
            '| {:^25} | {:^15} | {:^14} |'.format(
                "Товар",
                "Магазин",
                "Стоимость"
            )
        )
        print(line)

        for product in products:
            print(
                '| {:^25} | {:^15} | {:^14} |'.format(
                    product.get('product', ''),
                    product.get('shop', ''),
                    product.get('cost', 0)
                )
            )
            print(line)


def select(products, shop):
    """
    Выбрать товары из конкретного магазина.
    """
    result = []
    for product in products:
        if product.get('shop', '') == shop:
            result.append(product)
    return result


def validation(json_data):
```

```
    """
    Валидация данных
    """
    try:
        validate(instance=json_data, schema=schema)
    except:
        raise jsonschema.exceptions.ValidationError("Данные недействительны")

    msg = "Данные успешно загружены"
    return True, msg


def main(command_line=None):
    """
    Главная функция программы.
    """
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="Путь к файлу"
    )

    parser = argparse.ArgumentParser("products")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")

    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Добавить новый продукт"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="Название продуктов"
    )
    add.add_argument(
        "-s",
        "--shop",
        action="store",
        help="Название магазина"
    )
    add.add_argument(
        "-c",
        "--cost",
        action="store",
        type=float,
        required=True,
        help="Стоимость товара"
    )

    list = subparsers.add_parser(
        "list",
        parents=[file_parser],
        help="Отобразить список товаров"
    )
```

```
    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Выбрать товары из магазина"
    )
    select.add_argument(
        "-s",
        "--shop",
        action="store",
        type=str,
        required=True,
        help="Название магазина"
    )

    args = parser.parse_args(command_line)
    path = pathlib.Path.home() / args.filename

    is_dirty = False
    if path.exists:
        products = load_products(path)
    else:
        products = []

    match args.command:
        case 'add':
            products = add_product(
                products,
                args.name,
                args.shop,
                args.cost
            )
            is_dirty = True
        case 'list':
            product_list(products)
        case 'select':
            selected = select(products, args.shop)
            product_list(selected)

    if is_dirty:
        save_products(path, products)


if __name__ == '__main__':
    main()
```

Листинг 1 – Индивидуальное задание 1

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 argparse
import pathlib
from pathlib import Path
import sys
```

```
def dir_tree(path, prefix=''):
    print(f'{prefix}├── {Path(path).name}')
    for item in Path(path).iterdir():
        if item.is_dir():
            dir_tree(item, prefix + '│   ')
        else:
            print(f'{prefix}│   ├──{item.name}')


def main(command_line=None):
    """
    Главная функция программы.
    """
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "Path",
        action="store",
        help="Путь к директории"
    )
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")

    tree = subparsers.add_parser(
        "tree",
        parents=[file_parser],
        help="Отобразить дерево каталога"
    )

    args = parser.parse_args(command_line)

    if args.command == 'tree':
        path = pathlib.Path(args.Path)
        dir_tree(path)


if __name__ == '__main__':
    main()
```

Листинг 2 – Индивидуальное задание 2

Вывод: приобретение навыки по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.

Ответы на контрольные вопросы

1. Какие существовали средства для работы с файловой системой до Python 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?** Он регламентирует модуль pathlib.

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

Есть несколько разных способов создания пути. Прежде всего, существуют classmethods наподобие .cwd() (текущий рабочий каталог) и .home() (домашний каталог вашего пользователя):

```
>>> import pathlib
>>> pathlib.Path.cwd()
PosixPath('/home/gahjelle/realpython/')
```

Путь также может быть явно создан из его строкового представления:

```
>>> pathlib.Path(r'C:\Users\gahjelle\realpython\file.txt')
WindowsPath('C:/Users/gahjelle/realpython/file.txt')
```

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

5. **Как получить путь к родительским элементам файловой системы с помощью модуля pathlib ?** .parent : каталог, содержащий файл, или родительский каталог, если путь является каталогом

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

Для простого чтения и записи файлов в библиотеке `pathlib` есть несколько удобных методов:

- `.read_text()` : открыть путь в текстовом режиме и вернуть содержимое в виде строки.
- `.read_bytes()` : открыть путь в двоичном/байтовом режиме и вернуть содержимое в виде строки байтов.
- `.write_text()` : открыть путь и записать в него строковые данные.
- `.write_bytes()` : открыть путь в двоичном/байтовом режиме и записать в него данные.

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

- `.name`: имя файла без какого-либо каталога
- `.parent`: каталог, содержащий файл, или родительский каталог, если путь является каталогом
- `.stem`: имя файла без суффикса
- `.suffix`: расширение файла
- `.anchor`: часть пути перед каталогами

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

Через `pathlib` вы также получаете доступ к базовым операциям на уровне файловой системы, таким как перемещение, обновление и даже удаление файлов. По большей части эти методы не выдают предупреждение и не ждут подтверждения, прежде чем информация или файлы будут потеряны. Будьте осторожны при использовании этих методов.

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:

```
if not destination.exists():
    source.replace(destination)
```

Каталоги и файлы могут быть удалены с помощью .rmdir() и .unlink() соответственно.

9. Как выполнить подсчет файлов в файловой системе? Самым простым является метод .iterdir() , который перебирает все файлы в данном каталоге. В следующем примере комбинируется .iterdir() с классом collection.Counter для подсчета количества файлов каждого типа в текущем каталоге:

```
>>> import collections
>>> collections.Counter(p.suffix for p in pathlib.Path.cwd().iterdir())
Counter({'.md': 2, '.txt': 4, '.pdf': 2, '.py': 1})
```

Более гибкие списки файлов могут быть созданы с помощью методов .glob() и .rglob()(рекурсивный глоб). Например, pathlib.Path.cwd().glob('*.txt')

возвращает все файлы с суффиксом .txt в текущем каталоге. Следующее только подсчитывает типы файлов, начинающиеся с p:

```
>>> import collections
>>> collections.Counter(p.suffix for p in pathlib.Path.cwd().glob('*.p*'))
Counter({'.pdf': 2, '.py': 1})
```

10. Как отобразить дерево каталогов файловой системы?

В следующем примере определяется функция `tree()`, которая будет печатать визуальное дерево, представляющее иерархию файлов, с корнем в данном каталоге. Здесь мы также хотим перечислить подкаталоги, поэтому мы используем метод `.rglob()`:

```
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? Pathlib.Path возвращает объект либо WindowsPath , либо PosixPath.