

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»
Вариант 29

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

(подпись)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____
Ставрополь, 2024 г.

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

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

Порядок выполнения работы:

Вариант 29

Выполнены задания:

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 .

Код программы:

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

# 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 .

import argparse
import json
import os.path
import pathlib

# Вариант 29
def add_route(staff, start, end, number):
    """
    Добавить маршрут
    """
    staff.append({
        'name_start': start,
        'name_end': end,
        'number': number
    })
    return staff

def list(routes):
    """
    Вывести список маршрутов
    """
    if routes:
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
```

```

        '_' * 4,
        '_' * 30,
        '_' * 30,
        '_' * 8
    )
    print(line)

    print("| {:^4} | {:^30} | {:^30} | {:^8} |".format(
        "№",
        "Начальный пункт",
        "Конечный пункт",
        "Номер"
    )
    )
    print(line)

    for idx, route in enumerate(routes, 1):
        print("| {:>4} | {:<30} | {:<30} | {:>8} |".format(
            idx,
            route.get('name_start', ""),
            route.get('name_end', ""),
            route.get('number', 0)
        )
        )
        print(line)
    else:
        print("Список маршрутов пуст.")

def save_routes(file_name, staff):
    """
    Сохранить все маршруты в файл JSON.
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_routes(file_name):
    """
    Загрузить все маршруты из файла JSON.
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def select_routes(routes, command):
    """
    Вывести выбранные маршруты
    """
    station = command
    count = 0

    for route in routes:

```

```

        if (station.lower() == route["name_start"].lower() or
            station == route["name_end"].lower()):

            count += 1
            print('{:>4}: {}-{}, номер маршрута: {}'.format(count,
                route["name_start"], route["name_end"], route["number"]))

    if count == 0:
        print("Маршрут не найден.")

def main(command_line=None):
    """
    Основная функция
    """
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )
    file_parser.add_argument(
        "-g",
        "--general",
        action="store_true",
        help="Save file in general directory"
    )
    # Создать основной парсер командной строки.
    parser = argparse.ArgumentParser("routes")
    parser.add_argument(
        "--version",
        action="version",
        version="% (prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")
    # Создать субпарсер для добавления маршрута.
    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Add a new route"
    )
    add.add_argument(
        "-s",
        "--start",
        action="store",
        required=True,
        help="Start position on route"
    )
    add.add_argument(
        "-e",
        "--end",
        action="store",
        help="End position on route"
    )

```

```

)
add.add_argument(
    "-n",
    "--number",
    action="store",
    type=int,
    required=True,
    help="Number of route"
)
# Создать субпарсер для отображения всех маршрутов.
_ = subparsers.add_parser(
    "display",
    parents=[file_parser],
    help="Display all routes"
)
# Создать субпарсер для выбора маршрутов.
select = subparsers.add_parser(
    "select",
    parents=[file_parser],
    help="Select the routes"
)
select.add_argument(
    "-t",
    "--station",
    action="store",
    type=str,
    required=True,
    help="Routes with this station"
)
# Выполнить разбор аргументов командной строки.
args = parser.parse_args(command_line)
# Загрузить все маршруты из файла, если файл существует.
is_dirty = False

filepath = pathlib.Path.cwd() / args.filename
if args.general:
    filepath = pathlib.Path.home() / args.filename
if os.path.exists(filepath):
    routes = load_routes(filepath)
else:
    routes = []

# Добавить маршрут.
if args.command == "add":
    if(routes is None):
        routes = []
    routes = add_route(
        routes,
        args.start,
        args.end,
        args.number
    )

```

```

is_dirty = True
# Отобразить все маршруты.
elif args.command == "display":
    list(routes)

# Выбрать требуемые маршруты.
elif args.command == "select":
    select_routes(routes, args.station)
    print(args.station)
# Сохранить данные в файл, если список маршрутов был изменен.
if is_dirty:
    save_routes(filepath, routes)

if __name__ == '__main__':
    main()

```

Результат работы программы:

```

(Data_Analysis) C:\Users\HAIER\Desktop\Задания\Анализ данных\Data_lr_5\code>python ind_1.py display routes.json --general


| № | Начальный пункт | Конечный пункт | Номер |
|---|-----------------|----------------|-------|
| 1 | Москва          | Воронеж        | 1     |
| 2 | Норильск        | Воронеж        | 2     |


(Data_Analysis) C:\Users\HAIER\Desktop\Задания\Анализ данных\Data_lr_5\code>python ind_1.py display routes.json


| № | Начальный пункт | Конечный пункт | Номер |
|---|-----------------|----------------|-------|
| 1 | Москва          | Воронеж        | 1     |


(Data_Analysis) C:\Users\HAIER\Desktop\Задания\Анализ данных\Data_lr_5\code>

```

Рисунок 1 – Если нет параметра –general, то используется путь к текущему исполняемому файлу, в противном случае указывается путь к домашнему каталогу пользователя

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.

Код программы:

```

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

# 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.

```

```

import argparse
from pathlib import Path
import os
import datetime

def tree(directory, indent="", max_depth=0, current_depth=0,
        only_dirs=False, only_files=False, show_time=False,
        show_extension=True):
    if current_depth > max_depth:
        return

    for item in sorted(directory.iterdir(),
                        key=lambda item: (item.is_file(), item.name.lower())):
        if item.is_dir():
            if only_files:
                continue
            print(f'{indent}{item.name}/')
            tree(item, indent + ' ', max_depth, current_depth + 1,
                  only_dirs, only_files, show_time, show_extension)

        elif not only_dirs:
            file_info = (item.name if show_extension
                          else os.path.splitext(item.name)[0])

            if show_time:
                creation_time = datetime.datetime.fromtimestamp(
                    item.stat().st_ctime)
                file_info += (f" (Created: "
                              f"{creation_time.strftime('%Y-%m-%d %H:%M:%S')})")

            print(indent + file_info)

def main():
    """
    Основная функция
    """
    parser = argparse.ArgumentParser(description="Tree")
    parser.add_argument('directory', nargs='?', default='.', type=Path,
                        help='Start directory (default: current directory)')
    #Добавление параметра, задающего уровень вложенности
    parser.add_argument('--depth', type=int, default=0,
                        help='Maximum display depth of the directory tree')
    #Добавление параметра, отвечающего за отображение только директорий
    parser.add_argument('--dirs', action='store_true',
                        help='Display only directories')
    #Добавление параметра, отвечающего за отображение только файлов
    parser.add_argument('--files', action='store_true',
                        help='Display only files')
    #Добавление параметра, отвечающего за отображение времени создания файла
    parser.add_argument('--time', action='store_true',
                        help='Display file creation times')
    #Добавление параметра, отвечающего за отображение расширений файлов

```

```

parser.add_argument('--extension', action='store_true',
                    help='Hide file extensions')

args = parser.parse_args()

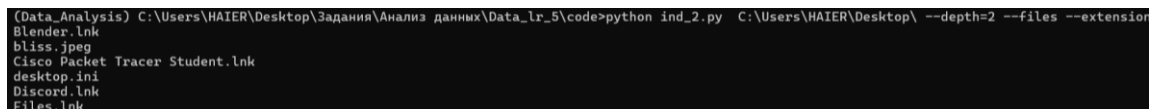
if not args.directory.is_dir():
    print('Error: The specified directory does not exist.')
    return

tree(args.directory, max_depth=args.depth, only_dirs=args.dirs,
     only_files=args.files, show_time=args.time,
     show_extension=args.extension)

if __name__ == '__main__':
    main()

```

Результат работы программы:



```

(Data_Analysis) C:\Users\HAIER\Desktop\Задания\Анализ данных\Data_lr_5\code>python ind_2.py C:\Users\HAIER\Desktop\ --depth=2 --files --extension
Blender.lnk
bliss.jpeg
Cisco Packet Tracer Student.lnk
desktop.ini
Discord.lnk
Files.lnk

```

Рисунок 2 – Вывод содержимого каталога с определёнными условиями (показывать расширение файлов, отображать только файлы и т.д.)

Ответы на контрольные вопросы:

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

Существовали следующие средства: os, os.path.

2. Что регламентирует PEP 428?

PEP 428 регламентирует представление дерева каталогов в стандартной библиотеке Python.

3. Как осуществляется создание путей средствами модуля pathlib ?

Используется метод Path.

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

Необходимо использовать метод resolve() или оператор /.

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

Для этого используется атрибут parent.

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

Создаются объекты `Path`, которые можно использовать для различных целей.

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

Необходимо использовать атрибуты объектов `Path`, такие как: `name`, `suffix`, `parent`.

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

Для перемещения файлов необходимо использовать метод `rename()` или `replace()`, а для удаления – метод `unlink()`.

9. Как выполнить подсчет файлов в файловой системе?

Для подсчета файлов можно использовать рекурсивную функцию.

10. Как отобразить дерево каталогов файловой системы?

Для отображения дерева каталогов файловой системы можно использовать рекурсивную функцию, которая будет выводить дерево каталогов.

11. Как создать уникальное имя файла?

Для создания уникального имени файла можно модуль `tempfile`.

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

Основное отличие заключается в используемых разделителях каталогов, есть различия и в поддерживаемых атрибутах файловых систем.

Выводы: в ходе выполнения работы получены навыки написания программ с интерфейсом командной строки, осуществляющих определённый набор действий с файловой системой.