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 №2.19
дисциплины «Анализ данных»
Вариант 13

Выполнил:
Иващенко Олег Андреевич
2 курс, группа ИВТ-б-о-22-1,
09.03.02 «Информационные и вычислительные машины»,
направленность (профиль)
«Программное обеспечение средств вычислительной техники и автоматизированных систем»

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович,
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: Приобретение навыков по работе с файловой системой с помощью библиотеки pathlib языка программирования Python версии 3.x.

Порядок выполнения работы

Индивидуальное задание 1. Для своего варианта лабораторной работы 2.17 добавить возможность хранения файла данных в домашнем каталоге пользователя. Для выполнения операций с файлом необходимо использовать модуль pathlib.

Листинг – Код программы individual_1.py

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


import json
import sys
from datetime import datetime
import argparse
import os.path
import pathlib


def print_help():
    """
    Функция вывода доступных пользователю команд
    """

    print("list - вывод всех добавленных записей")
    print("add - добавление новых записей")
    print("find - найти запись по фамилии")
    print("exit - завершение работы программы")



def add_worker(workers, surname, name, phone, date):
    """
    Функция добавления новой записи, возвращает запись
    """

    workers.append(
        {
            "surname": surname,
            'name': name,
            'phone': phone,
            'date': date
        }
    )

    return workers
```

```
def print_list(list):
    """
    Функция выводит на экран список всех существующих записей
    """

    for member in list:
        print(f"{member['surname']} {member['name']} | "
                f"{member['phone']} | {member['date']}")



def find_member(workers, period):
    """
    Функция для вывода на экран всех записей, чьи фамилии совпадают
    с введённой (не возвращает никаких значений)
    """

    count = 0
    members = []


    for member in workers:
        year = datetime.strptime(member['date'], "%d.%m.%Y").year
        if datetime.now().year - period >= year:
            members.append(member)
            count += 1

    if count == 0:
        print("Записи не найдены")
    else:
        return members



def get_home_path(filename):
    home_dir =  pathlib.Path.home()
    return home_dir / filename



def save_file(filename, data):
    """
    Сохранение списка сотрудников в файл формата JSON
    """


    with open(get_home_path(filename), "w", encoding="utf-8") as file:
        json.dump(data, file, ensure_ascii=False, indent=4)



def load_file(filename):
    """
    Загрузка данных о сотрудниках из указанного JSON-файла
    """


    with open(get_home_path(filename), "r", encoding="utf-8") as file:
        return json.load(file)
```

```
def parse_datetime(value):
    try:
        return datetime.strptime(value, "%d.%m.%Y")
    except ValueError:
        print("Error")



def main(command_line=None):
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="The data file name"
    )


    parser = argparse.ArgumentParser("workers")
    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="Add a new worker"
    )
    add.add_argument(
        "-s",
        "--surname",
        action="store",
        required=True,
        help="The worker's surname"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="The worker's name"
    )
    add.add_argument(
        "-p",
        "--phone",
        action="store",
        help="The worker's phone"
    )
    add.add_argument(
        "-d",
        "--date",
        action="store",
        required=True,
        help="The date of hiring"
```

```
    )


    _ = subparsers.add_parser(
        "display",
        parents=[file_parser],
        help="Display all workers"
    )


    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Select the workers"
    )
    select.add_argument(
        "-p",
        "--period",
        action="store",
        type=int,
        required=True,
        help="The required period"
    )


    args = parser.parse_args(command_line)


    is_dirty = False
    if os.path.exists(args.filename):
        workers = load_file(args.filename)
    else:
        workers = []


    if args.command == "add":
        workers = add_worker(
            workers,
            args.surname,
            args.name,
            args.phone,
            args.date
        )
        is_dirty = True

    elif args.command == "display":
        print_list(workers)


    elif args.command == "select":
        selected = find_member(workers, args.period)
        print_list(selected)


    if is_dirty:
        save_file(args.filename, workers)



if __name__ == "__main__":
    """
    Основная программа
```

```
    """
    main()
```

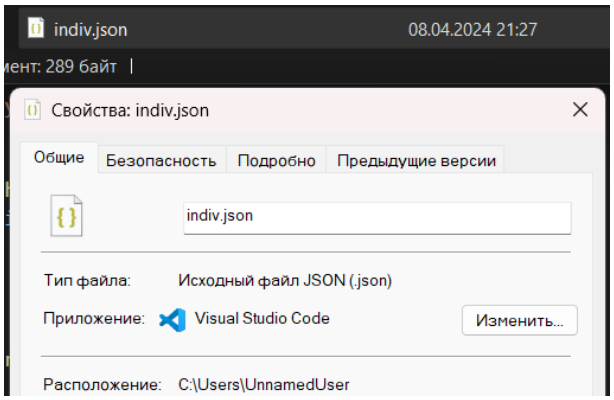

Рисунок 1.1 – Местоположение файла

Рисунок 1.2 – Первоначальное содержимое файла indiv.json

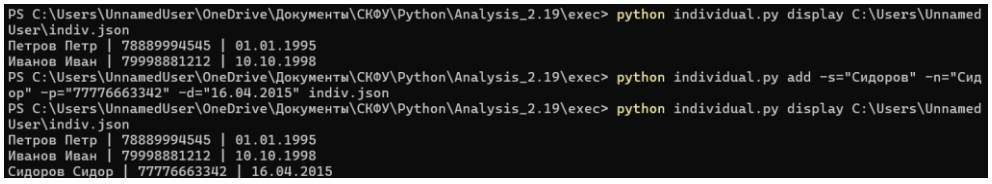

Рисунок 1.3 – Выполнение программы

Рисунок 1.4 – Новая запись в JSON-файле

Индивидуальное задание 2. Разработать аналог утилиты tree в Linux. Использовать возможности модуля argparse для управления отображаемым дерева каталогов файловой системы. Добавить уникальные возможности в данный программный продукт.

Листинг – Код программы individual_2.py

```
import os
import argparse



def tree(path, level, max_levels, show_hidden):
    """
    Вывод списка каталогов и файлов по указанному пути,
    аналогично утилите tree в ОС Linux
    """
    if level > max_levels:
        return

    for element in os.listdir(path):
        if not show_hidden and element.startswith('.'):
            continue


        dir = os.path.join(path, element)
        if os.path.isdir(dir):
            print('  ' * level + f'/{element}')
            tree(dir, level + 1, max_levels, show_hidden)
```

```
        else:
            print('  ' * level + element)



def main():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'directory',
        nargs='?',
        default='.',
        help="Директория"
    )


    parser.add_argument(
        '-l',
        '--level',
        type=int,
        default=float('inf')
    )


    parser.add_argument(
        '-a',
        '--all',
        action='store_true',
        help="Вывод скрытых файлов"
    )


    parser.add_argument(
        'author',
        nargs='?',
        const=True,
        help="Вывод автора программы"
    )


    args = parser.parse_args()


    if args.author is None:
        print(f'> Автор работы: Иващенко О.А.\n')
        return


    path = os.path.abspath(args.directory)
    if not os.path.exists(path):
        print("Указанного каталога не существует")
        return

    if not os.path.isdir(path):
        print(f"Ошибка: {path} - не каталог")
        return
```

```
    print(f'Список файлов в каталоге {path}')
    tree(path, 0, args.level, args.all)



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

Рисунок 2.1 – Вывод содержимого без скрытых файлов

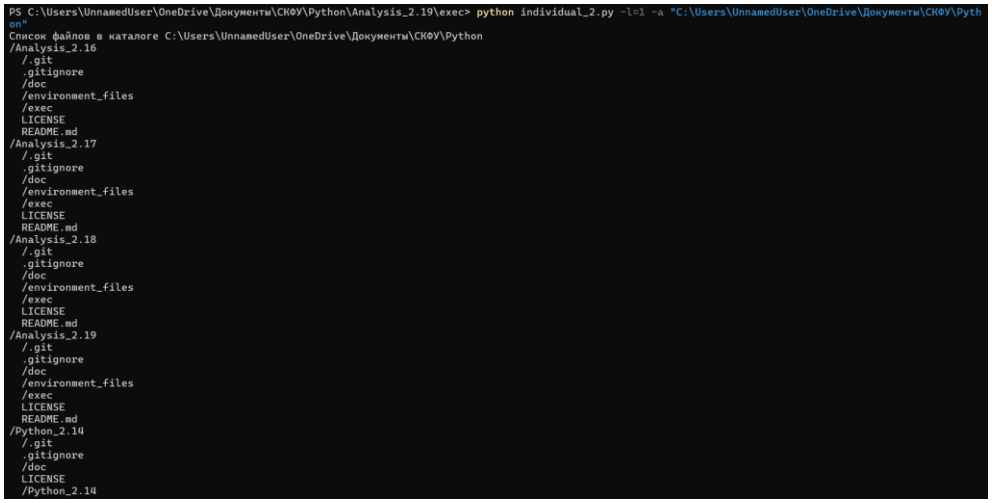

Рисунок 2.2 – Вывод содержимого с скрытыми файлами

Рисунок 2.3 – Указание автора работы

Контрольные вопросы

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

До Python 3.4 существовали различные средства для работы с файловой системой, такие как модули os, os.path, shutil.

2. Что регламентирует PEP 428?

PEP 428 регламентирует модуль pathlib, который предоставляет обектно-ориентированный интерфейс для работы с путями к файлами и каталогам.

3. Как осуществляется создание путей средствами модуля pathlib?

Создание путей средствами модуля pathlib осуществляется с помощью метода Path() и указания пути к файлу или каталогу в виде строки.

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

Для получения пути дочернего элемента файловой системы с помощью модуля pathlib используется метод joinpath().

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

Получение пути к родительским элементам файловой системы с помощью модуля pathlib выполняется с использованием атрибута parent.

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

Операции с файлами с помощью модуля pathlib выполняются с использованием методов этого модуля, таких как open() для открытия файла и unlink() для удаления файла.

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

Компоненты пути файловой системы могут быть выделены с помощью атрибутов объектов Path, таких как name, stem, suffix, suffixes.

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

Перемещение и удаление файлов с помощью модуля pathlib выполняется с использованием методов replace() для перемещения и unlink() для удаления.

9. Как выполнить подсчет файлов в файловой системе?

Для подсчёта файлов в файловой системе можно использовать рекурсивный обход каталогов и подсчёт файлов.

10. Как отобразить дерево каталогов файловой системы?

Для отображения дерева каталогов файловой системы можно использовать рекурсивную функцию, которая будет проходить по всем файлам и подкаталогам, выводя их иерархический список.

11. Как создать уникальное имя файла?

Уникальное имя файла можно создать, добавив к имени файла уникальный идентификатор, например, текущее время или случайное число.

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

Отличия в использовании модуля pathlib для различных ОС заключается в различиях в путях к файлам и каталогам, таких как использование обратного слеша в Windows и прямого слеша в UNIX-подобных системах. Модуль pathlib автоматически учитывает эти различия при работе с путями.

Выводы: В процессе выполнения лабораторной работы были приобретены навыки по работе с файловой системой при помощи библиотеки pathlib языка программирования Python. Были выполнены две индивидуальные задачи.