**Лабораторная работа №4**

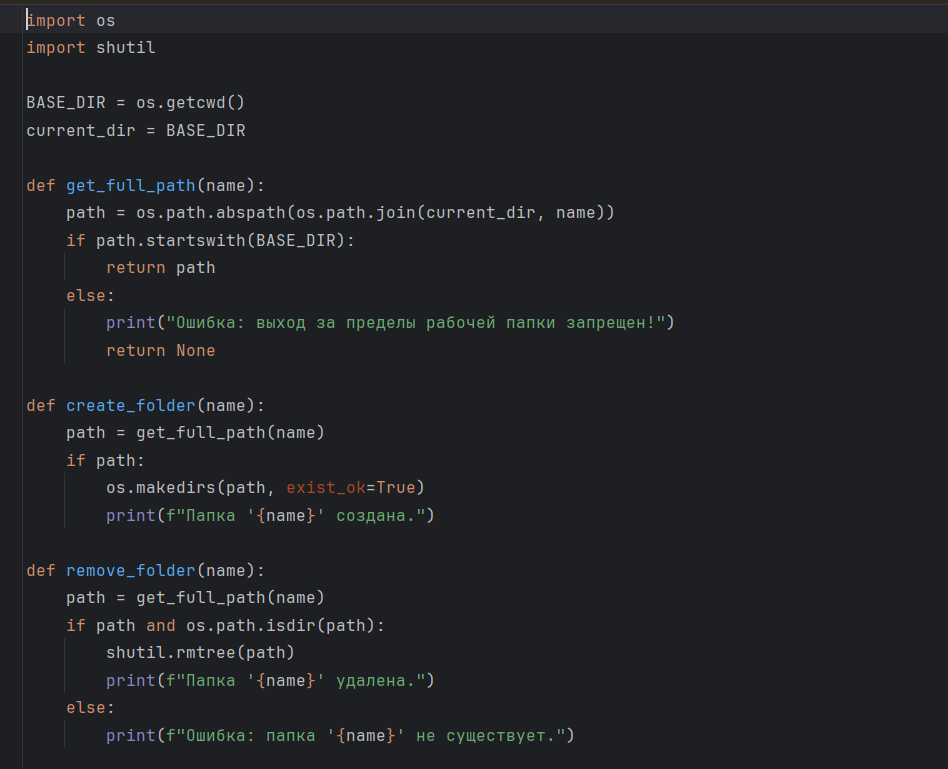
Файловый менеджер

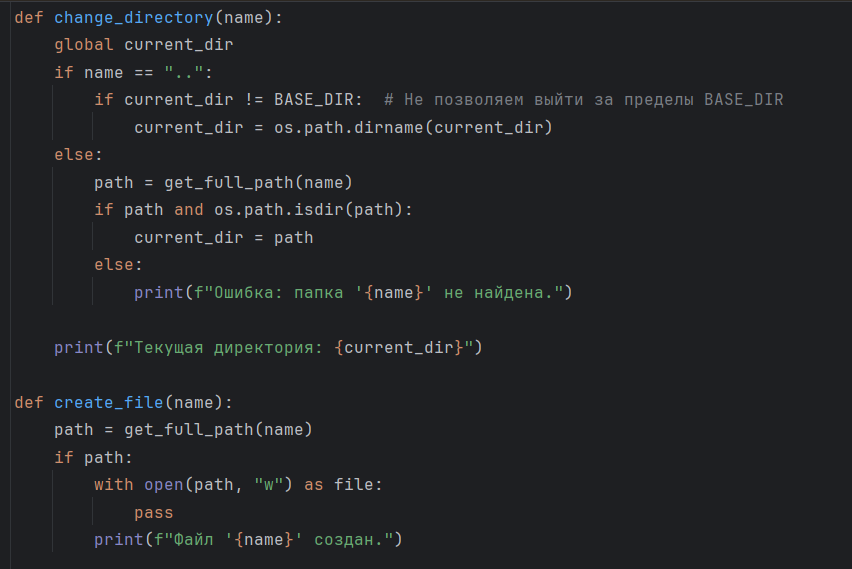
**Задания для выполнения**

Необходимо создать примитивный файловый менеджер. Программа должна работать в определенной папке (рабочей папки менеджера) и позволять пользователю выполнять следующие простые действия в пределах рабочей папки:

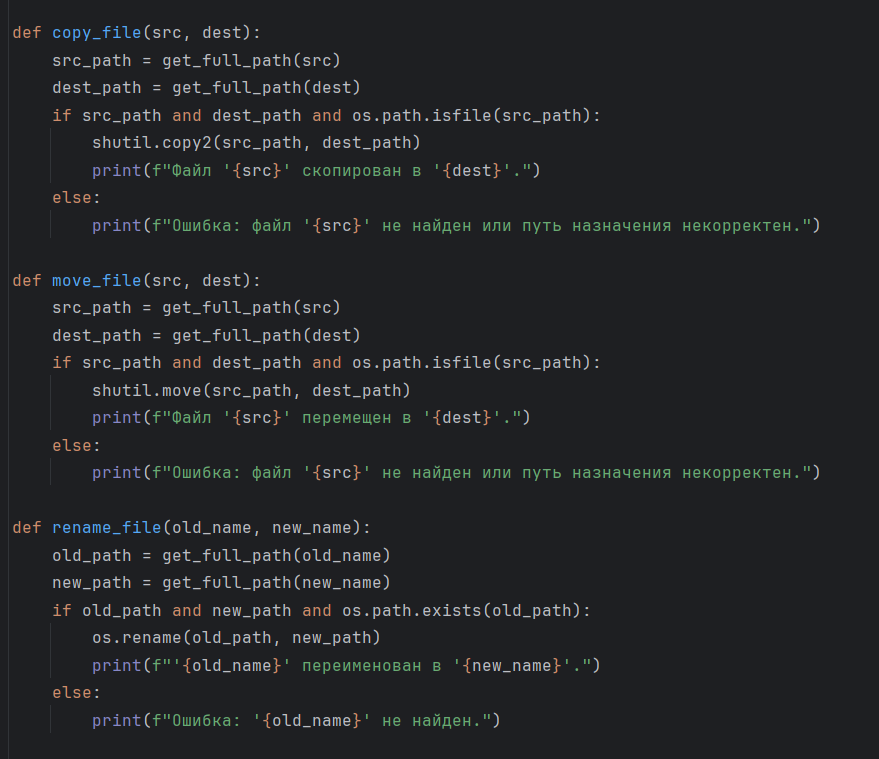
1. Создание папки (с указанием имени);
2. Удаление папки по имени;
3. Перемещение между папками (в пределах рабочей папки) - заход в папку по имени, выход на уровень вверх;
4. Создание пустых файлов с указанием имени;
5. Запись текста в файл;
6. Просмотр содержимого текстового файла;
7. Удаление файлов по имени;
8. Копирование файлов из одной папки в другую;
9. Перемещение файлов;
10. Переименование файлов.

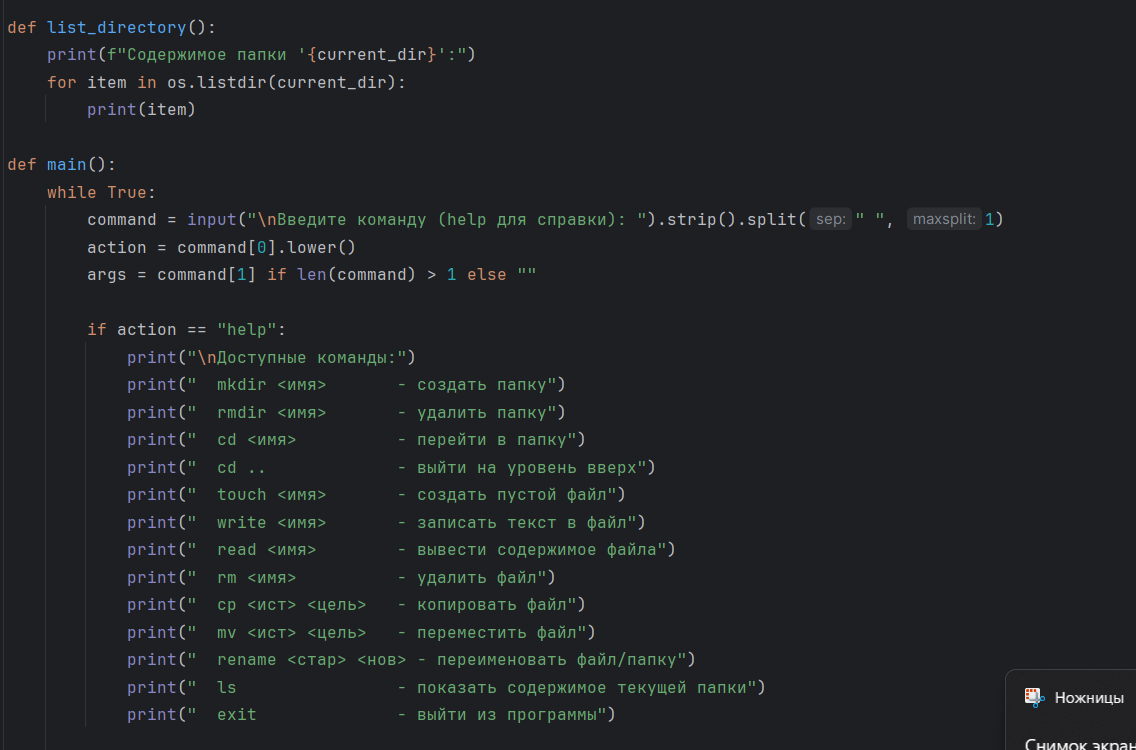
Сначала создадим файл в python: Lab4.py, а далее запишем в него код, который реализует примитивный файловый менеджер.



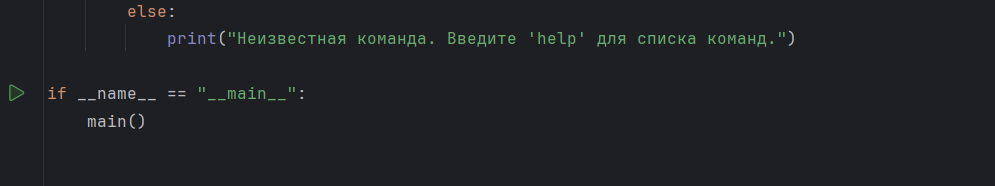












Разберем код построчно и по функциям, чтобы понять, как он работает.

1. Импорт библиотек

import os

import shutil

• os – библиотека для работы с файлами и папками (создание, удаление, перемещение и т. д.).

• shutil – библиотека для копирования и перемещения файлов и папок.

2. Определение рабочей папки

BASE\_DIR = os.getcwd()

current\_dir = BASE\_DIR

• os.getcwd() – возвращает путь к текущей папке, где запущен скрипт.

• BASE\_DIR – определяет основную рабочую папку, за пределы которой нельзя выходить.

• current\_dir – хранит текущую папку, в которой мы находимся.

3. Функция get\_full\_path(name)

def get\_full\_path(name):

    path = os.path.abspath(os.path.join(current\_dir, name))

    if path.startswith(BASE\_DIR):

        return path

    else:

        print("Ошибка: выход за пределы рабочей папки запрещен!")

        return None

Что делает:

• Получает полный путь к файлу или папке.

• Проверяет, чтобы путь не выходил за пределы BASE\_DIR.

• Если путь безопасный – возвращает его, иначе выводит ошибку.

4. Функции работы с папками

Создание папки

def create\_folder(name):

    path = get\_full\_path(name)

    if path:

        os.makedirs(path, exist\_ok=True)

        print(f"Папка '{name}' создана.")

• os.makedirs(path, exist\_ok=True) – создает папку (если она уже существует, ошибка не будет вызвана).

• Выводит сообщение о создании.

Удаление папки

def remove\_folder(name):

    path = get\_full\_path(name)

    if path and os.path.isdir(path):

        shutil.rmtree(path)

        print(f"Папка '{name}' удалена.")

    else:

        print(f"Ошибка: папка '{name}' не существует.")

• Проверяет, существует ли папка.

• shutil.rmtree(path) – удаляет папку со всем содержимым.

Смена текущей папки

def change\_directory(name):

    """Перемещается в указанную папку или выходит на уровень вверх."""

    global current\_dir

    if name == "..":

        if current\_dir != BASE\_DIR:

            current\_dir = os.path.dirname(current\_dir)

    else:

        path = get\_full\_path(name)

        if path and os.path.isdir(path):

            current\_dir = path

        else:

            print(f"Ошибка: папка '{name}' не найдена.")

    print(f"Текущая директория: {current\_dir}")

• cd <имя папки> – переходит в указанную папку.

• cd .. – поднимается на уровень выше, но не выходит за пределы BASE\_DIR.

5. Функции работы с файлами

Создание пустого файла

def create\_file(name):

    path = get\_full\_path(name)

    if path:

        with open(path, "w") as file:

            pass

        print(f"Файл '{name}' создан.")

• Использует open(path, "w") – создает пустой файл.

Запись в файл

def write\_to\_file(name, text):

    path = get\_full\_path(name)

    if path and os.path.isfile(path):

        with open(path, "w", encoding="utf-8") as file:

            file.write(text)

        print(f"Текст записан в файл '{name}'.")

    else:

        print(f"Ошибка: файл '{name}' не существует.")

• Открывает файл в режиме "w" (перезаписывает его содержимое).

• Записывает переданный text в файл.

Чтение файла

def read\_file(name):

    path = get\_full\_path(name)

    if path and os.path.isfile(path):

        with open(path, "r", encoding="utf-8") as file:

            print(f"Содержимое файла '{name}':\n")

            print(file.read())

    else:

        print(f"Ошибка: файл '{name}' не найден.")

• Открывает файл в режиме "r" и выводит его содержимое.

Удаление файла

def delete\_file(name):

    path = get\_full\_path(name)

    if path and os.path.isfile(path):

        os.remove(path)

        print(f"Файл '{name}' удален.")

    else:

        print(f"Ошибка: файл '{name}' не существует.")

• os.remove(path) – удаляет файл.

6. Копирование, перемещение и переименование файлов

Копирование файла

def copy\_file(src, dest):

    src\_path = get\_full\_path(src)

    dest\_path = get\_full\_path(dest)

    if src\_path and dest\_path and os.path.isfile(src\_path):

        shutil.copy2(src\_path, dest\_path)

        print(f"Файл '{src}' скопирован в '{dest}'.")

    else:

        print(f"Ошибка: файл '{src}' не найден или путь назначения некорректен.")

• shutil.copy2(src, dest) – копирует файл, сохраняя метаданные.

Перемещение файла

def move\_file(src, dest):

    src\_path = get\_full\_path(src)

    dest\_path = get\_full\_path(dest)

    if src\_path and dest\_path and os.path.isfile(src\_path):

        shutil.move(src\_path, dest\_path)

        print(f"Файл '{src}' перемещен в '{dest}'.")

    else:

        print(f"Ошибка: файл '{src}' не найден или путь назначения некорректен.")

• shutil.move(src, dest) – перемещает файл.

Переименование файла

def rename\_file(old\_name, new\_name):

    old\_path = get\_full\_path(old\_name)

    new\_path = get\_full\_path(new\_name)

    if old\_path and new\_path and os.path.exists(old\_path):

        os.rename(old\_path, new\_path)

        print(f"'{old\_name}' переименован в '{new\_name}'.")

    else:

        print(f"Ошибка: '{old\_name}' не найден.")

• os.rename(old\_path, new\_path) – переименовывает файл или папку.

7. Основной цикл программы

def main():

    """Основной цикл программы."""

    while True:

        command = input("\nВведите команду: ").strip().split(" ", 1)

        action = command[0].lower()

        args = command[1] if len(command) > 1 else ""

        if action == "mkdir":

            create\_folder(args)

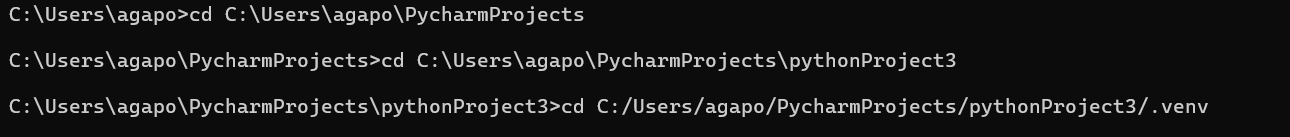
        elif action == "exit":

            break

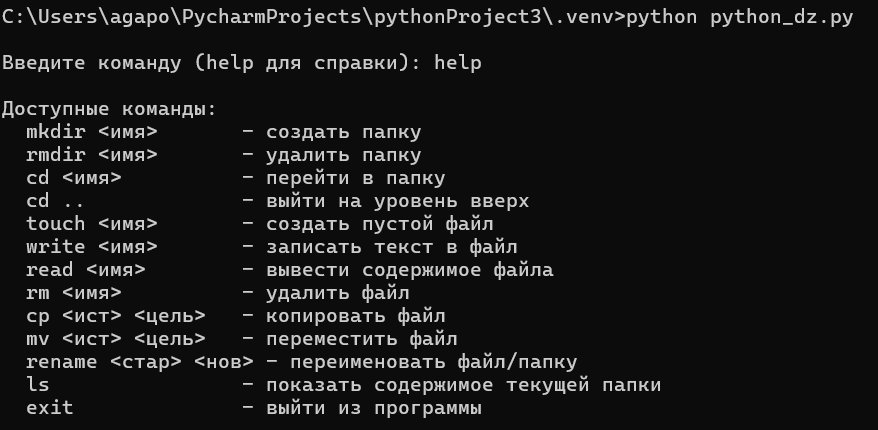
• Запрашивает команду у пользователя.

• В зависимости от введенной команды вызывает соответствующую функцию.

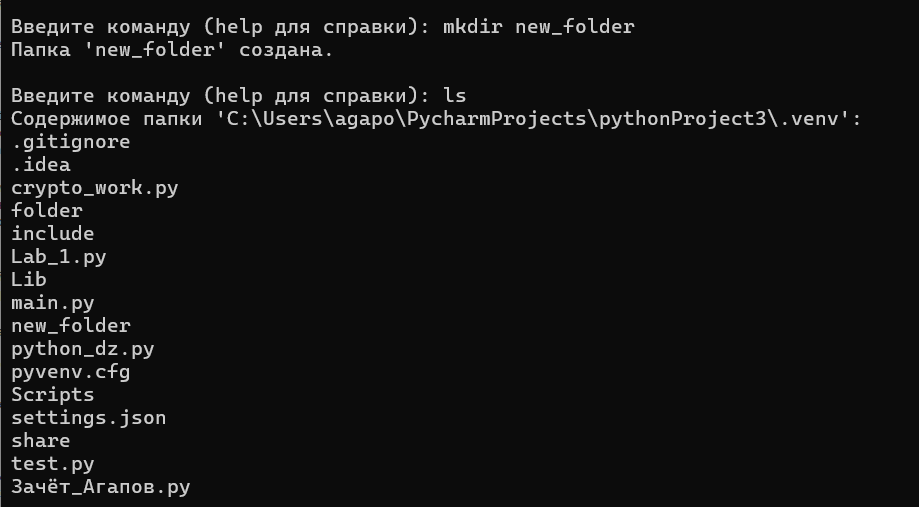
Чтобы запустить менеджер, зайдем в cmd Linux и перейдем в путь, где распорлагается файл:



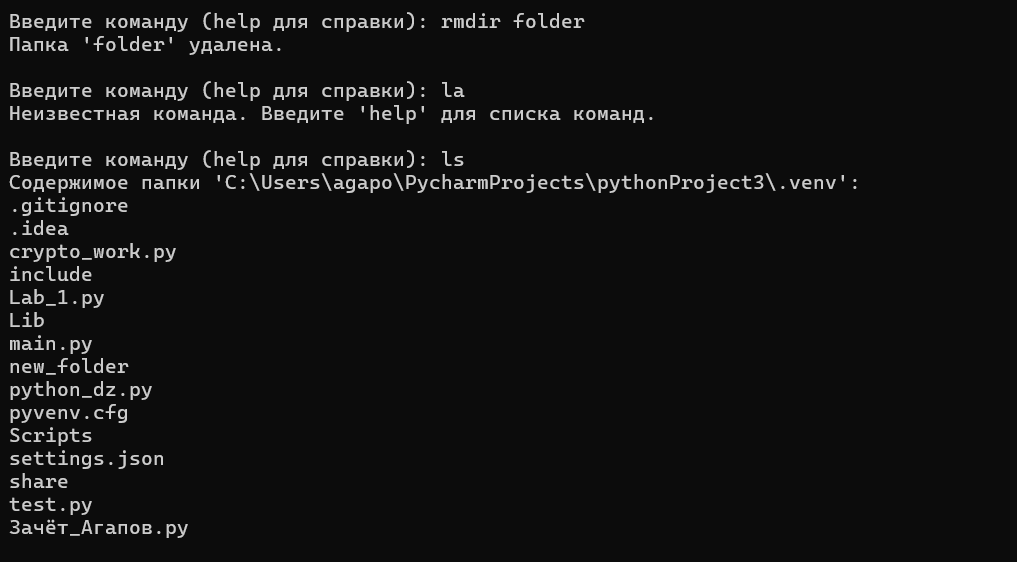
Далее запустим выполнение:



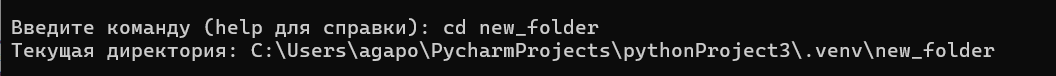
1. Создадим папку



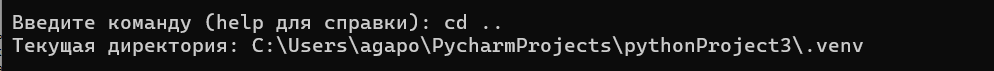
1. Удалим папку



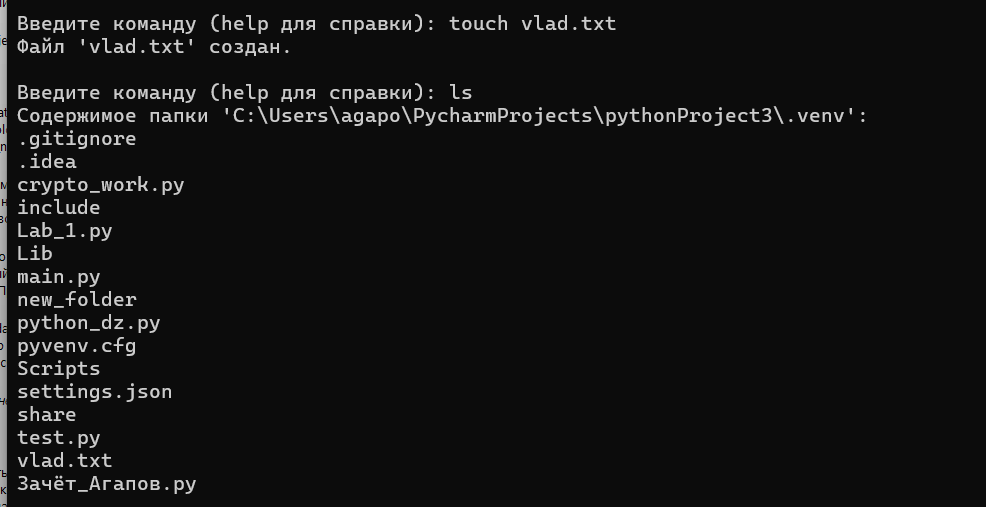
1. Перейдем в папку



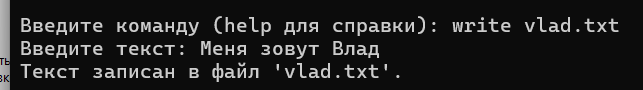
1. Выйдем из папки



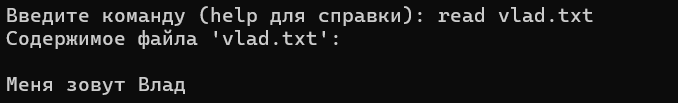
1. Создадим пустой файл



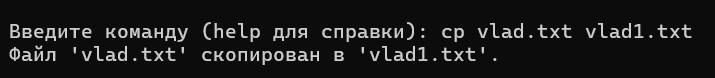
1. Запишем в него текст



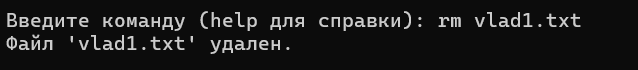
1. Выведем содержимое



1. Копируем файл



1. Удалим файл



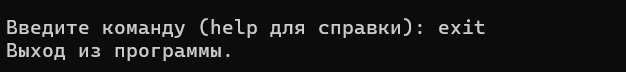
1. Переместим файл



1. Переименуем папку



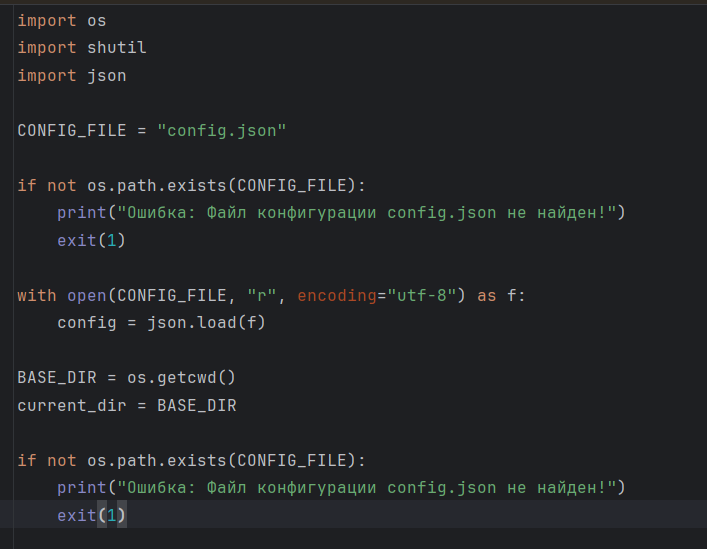
1. Завершим выполнение программы



Теперь запустим этот файловый менеджер на виртуальной машине Линукс Минт. Для этого создадим конфигурационный файл config.json, где укажем рабочую папку Линукс

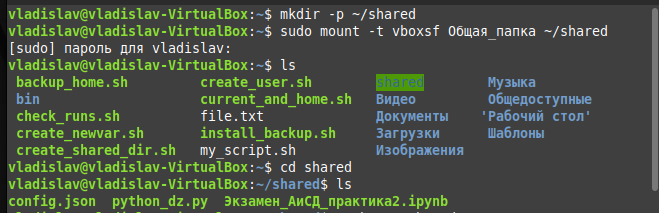


Добавим в наш основной код считывание файла json

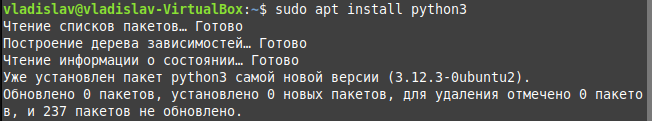


Затем в разделе общие папки добавим наши файлы.

Заходим в Linux и добавляем файлы через общую папку в папку shared



Установим python 3



Перейдем в папку shared и запустим файловый менеджер:

