Задание 4 (семинар):

Функция получает на вход текст вида: «2-й четверг ноября», «3-я среда мая» и т.п.   
Преобразуйте его в дату в текущем году

Логируйте ошибки если текст не соответствует формату

import logging

from datetime import datetime, timedelta

import calendar

import re

# Настройка логирования

logging.basicConfig(filename='date\_conversion\_errors.log', level=logging.ERROR,

                    format='%(asctime)s:%(levelname)s:%(message)s')

# Сопоставление русских и английских названий дней недели и месяцев

weekday\_translation = {

    'понедельник': 'Monday',

    'вторник': 'Tuesday',

    'среда': 'Wednesday',

    'четверг': 'Thursday',

    'пятница': 'Friday',

    'суббота': 'Saturday',

    'воскресенье': 'Sunday'

}

month\_translation = {

    'января': 'January',

    'февраля': 'February',

    'марта': 'March',

    'апреля': 'April',

    'мая': 'May',

    'июня': 'June',

    'июля': 'July',

    'августа': 'August',

    'сентября': 'September',

    'октября': 'October',

    'ноября': 'November',

    'декабря': 'December'

}

def convert\_to\_date(text):

    # Регулярное выражение для парсинга входного текста

    pattern = re.compile(r"(\d+)-(й|я) (\w+) (\w+)")

    match = pattern.match(text)

    # Если формат не соответствует, логируем ошибку

    if not match:

        logging.error(f"Некорректный формат: {text}")

        return None

    # Извлечение данных из входного текста

    week\_number, week\_suffix, weekday\_name, month\_name = match.groups()

    try:

        # Конвертация номера недели в целое число

        week\_number = int(week\_number)

        # Получение текущего года

        current\_year = datetime.now().year

        # Перевод названий на английский для работы с модулем datetime

        month\_name\_eng = month\_translation[month\_name.lower()]

        weekday\_name\_eng = weekday\_translation[weekday\_name.lower()]

        # Получение номера месяца

        month\_number = list(calendar.month\_name).index(month\_name\_eng)

        # Получение номера дня недели

        weekday\_number = list(calendar.day\_name).index(weekday\_name\_eng)

    except (ValueError, KeyError) as e:

        # Логирование ошибки при конвертации данных

        logging.error(f"Ошибка при конвертации: {text} - {e}")

        return None

    # Проверка корректности номера недели

    if week\_number < 1 or week\_number > 5:

        logging.error(f"Неверный номер недели: {week\_number}")

        return None

    # Находим первый день месяца

    first\_day\_of\_month = datetime(current\_year, month\_number, 1)

    first\_weekday\_of\_month = first\_day\_of\_month.weekday()

    # Рассчитываем смещение до нужного дня недели

    days\_to\_add = (weekday\_number - first\_weekday\_of\_month + 7) % 7

    first\_occurrence = first\_day\_of\_month + timedelta(days=days\_to\_add)

    # Находим нужный день

    target\_date = first\_occurrence + timedelta(weeks=week\_number-1)

    # Проверка, что дата находится в правильном месяце

    if target\_date.month != month\_number:

        logging.error(f"Недопустимая дата: {text}")

        return None

    return target\_date

# Пример использования функции с выводом успешного результата

date\_texts = ["2-й четверг ноября", "3-я среда мая", "5-й понедельник января", "$$$", "среда ноября", "третья среда мая", "6-й понедельник января", "2-й четвёрг ноябрь", "3-я среда июля"]

for text in date\_texts:

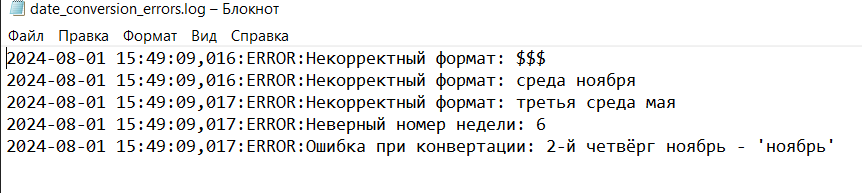
    date = convert\_to\_date(text)

    if date:

        print(f"Дата для '{text}': {date.strftime('%Y-%m-%d')}")

    else:

        print(f"Не удалось конвертировать дату для '{text}'")



Без кода к 4 заданию не хотел писать решение пятого задания  
  
  
5 задание:

Добавьте возможность запуска из командной строки. При этом значение любого параметра можно опустить. В этом случае берётся первый в месяце день недели, текущий день недели и/или текущий месяц.

import logging

from datetime import datetime, timedelta

import calendar

import re

import argparse

# Настройка логирования

logging.basicConfig(filename='date\_conversion\_errors.log', level=logging.ERROR,

                    format='%(asctime)s:%(levelname)s:%(message)s')

# Сопоставление русских и английских названий дней недели и месяцев

weekday\_translation = {

    'понедельник': 'Monday',

    'вторник': 'Tuesday',

    'среда': 'Wednesday',

    'четверг': 'Thursday',

    'пятница': 'Friday',

    'суббота': 'Saturday',

    'воскресенье': 'Sunday'

}

month\_translation = {

    'января': 'January',

    'февраля': 'February',

    'марта': 'March',

    'апреля': 'April',

    'мая': 'May',

    'июня': 'June',

    'июля': 'July',

    'августа': 'August',

    'сентября': 'September',

    'октября': 'October',

    'ноября': 'November',

    'декабря': 'December'

}

def convert\_to\_date(text):

    # Регулярное выражение для парсинга входного текста

    pattern = re.compile(r"(\d\*)-?(й|я)? ?(\w+)? ?(\w+)?")

    match = pattern.match(text)

    # Если формат не соответствует, логируем ошибку

    if not match:

        logging.error(f"Некорректный формат: {text}")

        return None

    # Извлечение данных из входного текста

    week\_number, week\_suffix, weekday\_name, month\_name = match.groups()

    # Получение текущих значений

    current\_date = datetime.now()

    current\_weekday\_name = current\_date.strftime('%A').lower()

    current\_month\_name = current\_date.strftime('%B').lower()

    # Установка значений по умолчанию

    week\_number = int(week\_number) if week\_number else 1

    weekday\_name = weekday\_name if weekday\_name else current\_weekday\_name

    month\_name = month\_name if month\_name else current\_month\_name

    try:

        # Перевод названий на английский для работы с модулем datetime

        month\_name\_eng = month\_translation[month\_name.lower()]

        weekday\_name\_eng = weekday\_translation[weekday\_name.lower()]

        # Получение номера месяца

        month\_number = list(calendar.month\_name).index(month\_name\_eng)

        # Получение номера дня недели

        weekday\_number = list(calendar.day\_name).index(weekday\_name\_eng)

    except (ValueError, KeyError) as e:

        # Логирование ошибки при конвертации данных

        logging.error(f"Ошибка при конвертации: {text} - {e}")

        return None

    # Проверка корректности номера недели

    if week\_number < 1 or week\_number > 5:

        logging.error(f"Неверный номер недели: {week\_number}")

        return None

    # Находим первый день месяца

    first\_day\_of\_month = datetime(current\_date.year, month\_number, 1)

    first\_weekday\_of\_month = first\_day\_of\_month.weekday()

    # Рассчитываем смещение до нужного дня недели

    days\_to\_add = (weekday\_number - first\_weekday\_of\_month + 7) % 7

    first\_occurrence = first\_day\_of\_month + timedelta(days=days\_to\_add)

    # Находим нужный день

    target\_date = first\_occurrence + timedelta(weeks=week\_number-1)

    # Проверка, что дата находится в правильном месяце

    if target\_date.month != month\_number:

        logging.error(f"Недопустимая дата: {text}")

        return None

    return target\_date

def main():

    parser = argparse.ArgumentParser(description="Конвертация текстового описания даты в дату текущего года.")

    args = parser.parse\_args()

    while True:

        user\_input = input("Введите желаемую дату для анализа (или нажмите Enter для выхода): ").strip()

        if not user\_input:

            break

        date = convert\_to\_date(user\_input)

        if date:

            print(f"Дата для '{user\_input}': {date.strftime('%Y-%m-%d')}")

        else:

            print(f"Не удалось конвертировать дату для '{user\_input}'")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Для проверки использую следующие фразы:

среда ноября

2-й четверг в ноябре

6-й понедельник января

0-й вторник февраля

2-й четвёрг ноябрь

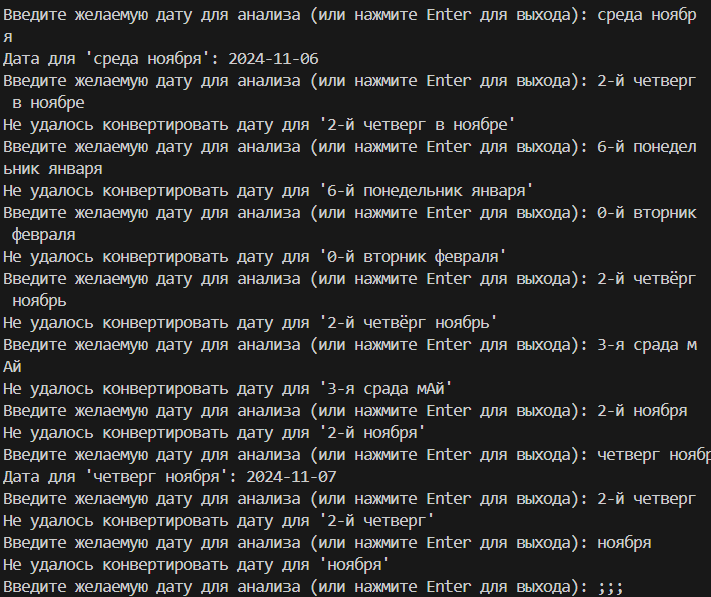
3-я срада мАй

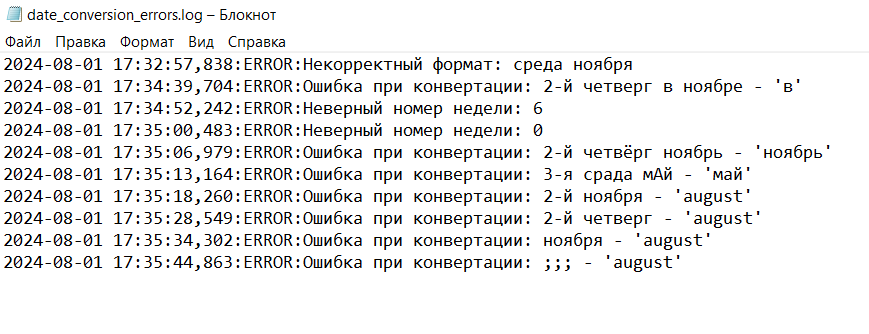
2-й ноября

четверг ноября

2-й четверг

ноября

;;;



Задание №6 Напишите код, который запускается из командной строки и получает на вход путь до директории на ПК. Соберите информацию о содержимом в виде объектов namedtuple. Каждый объект хранит: ○ имя файла без расширения или название каталога, ○ расширение, если это файл, ○ флаг каталога, ○ название родительского каталога. В процессе сбора сохраните данные в текстовый файл используя логирование.

import os

import logging

from collections import namedtuple

# Настройка логирования

logging.basicConfig(filename='directory\_contents.log', level=logging.DEBUG,

                    format='%(asctime)s:%(levelname)s:%(message)s')

# Определение структуры данных

FileInfo = namedtuple('FileInfo', ['name', 'extension', 'is\_directory', 'parent\_directory'])

def collect\_directory\_info(directory\_path):

    """Собирает информацию о содержимом директории."""

    if not os.path.isdir(directory\_path):

        logging.error(f"Указанный путь не является директорией: {directory\_path}")

        print(f"Ошибка: '{directory\_path}' не является директорией.")

        return

    logging.info(f"Начинаем обработку директории: {directory\_path}")

    try:

        # Проход по всем каталогам и файлам в указанной директории

        for root, dirs, files in os.walk(directory\_path):

            logging.info(f"Обрабатываем каталог: {root}")

            for name in dirs:

                # Для каталогов

                dir\_path = os.path.join(root, name)

                parent\_dir = os.path.basename(root)

                file\_info = FileInfo(

                    name=name,

                    extension=None,

                    is\_directory=True,

                    parent\_directory=parent\_dir

                )

                logging.info(f"Каталог: {file\_info}")

            for name in files:

                # Для файлов

                file\_path = os.path.join(root, name)

                parent\_dir = os.path.basename(root)

                name\_without\_ext, ext = os.path.splitext(name)

                file\_info = FileInfo(

                    name=name\_without\_ext,

                    extension=ext[1:],  # убираем точку из расширения

                    is\_directory=False,

                    parent\_directory=parent\_dir

                )

                logging.info(f"Файл: {file\_info}")

    except PermissionError as e:

        logging.error(f"Ошибка доступа к директории '{directory\_path}': {e}")

    except FileNotFoundError as e:

        logging.error(f"Файл или директория не найдены при обработке '{directory\_path}': {e}")

    except OSError as e:

        logging.error(f"OS ошибка при обработке '{directory\_path}': {e}")

    except Exception as e:

        logging.error(f"Неизвестная ошибка при сборе информации из '{directory\_path}': {e}")

def main():

    while True:

        # Запрос директорий у пользователя

        directories\_input = input("Введите пути до директорий, разделенные пробелом (или нажмите Enter для выхода): ")

        # Проверка на пустую строку

        if not directories\_input.strip():

            print("Завершение программы.")

            break

        directories = directories\_input.split()

        # Обработка каждой директории

        for directory in directories:

            if os.path.isdir(directory):

                print(f"Обрабатывается директория: {directory}")

                collect\_directory\_info(directory)

            else:

                # Если директория не существует, записываем ошибку в лог

                logging.error(f"Указанный путь не является директорией: {directory}")

                print(f"Ошибка: '{directory}' не является директорией.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()



1\*

Урок 5. Генераторы и генераторные выражения

Напишите функцию get\_file\_info, которая принимает на вход строку - абсолютный путь до файла.  
Функция возвращает кортеж из трёх элементов: путь, имя файла, расширение файла.

**Пример использования**.  
На входе:

file\_path = "C:/Users/User/Documents/example.txt"

На выходе:

('C:/Users/User/Documents/', 'example', '.txt')

Задание: Добавить логирование ошибок и полезной информации. Также реализуйте возможность запуска из командной строки с передачей параметров

import os

import logging

import time

from datetime import datetime

# Настройка логирования

logging.basicConfig(filename='file\_info.log', level=logging.DEBUG,

                    format='%(asctime)s:%(levelname)s:%(message)s')

# Настройка логирования пользовательских действий

user\_log = logging.getLogger('user\_log')

user\_log.setLevel(logging.DEBUG)

fh = logging.FileHandler('user\_activity.log')

formatter = logging.Formatter('%(asctime)s:%(message)s')

fh.setFormatter(formatter)

user\_log.addHandler(fh)

def get\_file\_info(file\_path):

    """Получает информацию о файле: путь, имя и расширение."""

    try:

        # Проверка существования пути

        if not os.path.exists(file\_path):

            raise FileNotFoundError(f"Путь не существует: {file\_path}")

        # Получаем имя файла с расширением

        file\_name\_with\_ext = os.path.basename(file\_path)

        # Разделяем имя файла на имя и расширение

        file\_name, file\_extension = os.path.splitext(file\_name\_with\_ext)

        # Получаем путь к директории

        directory\_path = os.path.dirname(file\_path)

        # Логируем успешную обработку

        logging.info(f"Успешно обработан файл: {file\_path}")

        # Возвращаем кортеж с результатами

        return (directory\_path, file\_name, file\_extension)

    except Exception as e:

        logging.error(f"Ошибка при обработке пути '{file\_path}': {e}")

        return (None, None, None)

def main():

    file\_paths = []

    start\_time = time.time()

    user\_log.info("Начало работы программы")

    while True:

        path = input("Введите путь к файлу (или 'stop' для завершения ввода): ")

        user\_log.info(f"Ввод пользователя: {path}")

        if path.lower() == 'stop':

            break

        file\_paths.append(path)

    total\_files = len(file\_paths)

    user\_log.info(f"Количество введенных путей: {total\_files}")

    for path in file\_paths:

        start\_request\_time = time.time()

        print(f"Обрабатывается путь: {path}")

        file\_info = get\_file\_info(path)

        end\_request\_time = time.time()

        if file\_info[0] is not None:

            print(f"Путь: {file\_info[0]}")

            print(f"Имя файла: {file\_info[1]}")

            print(f"Расширение: {file\_info[2]}")

        else:

            print(f"Не удалось получить информацию о файле: {path}")

        user\_log.info(f"Ответ программы для {path}: {file\_info}")

        user\_log.info(f"Время обработки запроса: {end\_request\_time - start\_request\_time:.2f} секунд")

    end\_time = time.time()

    total\_time = end\_time - start\_time

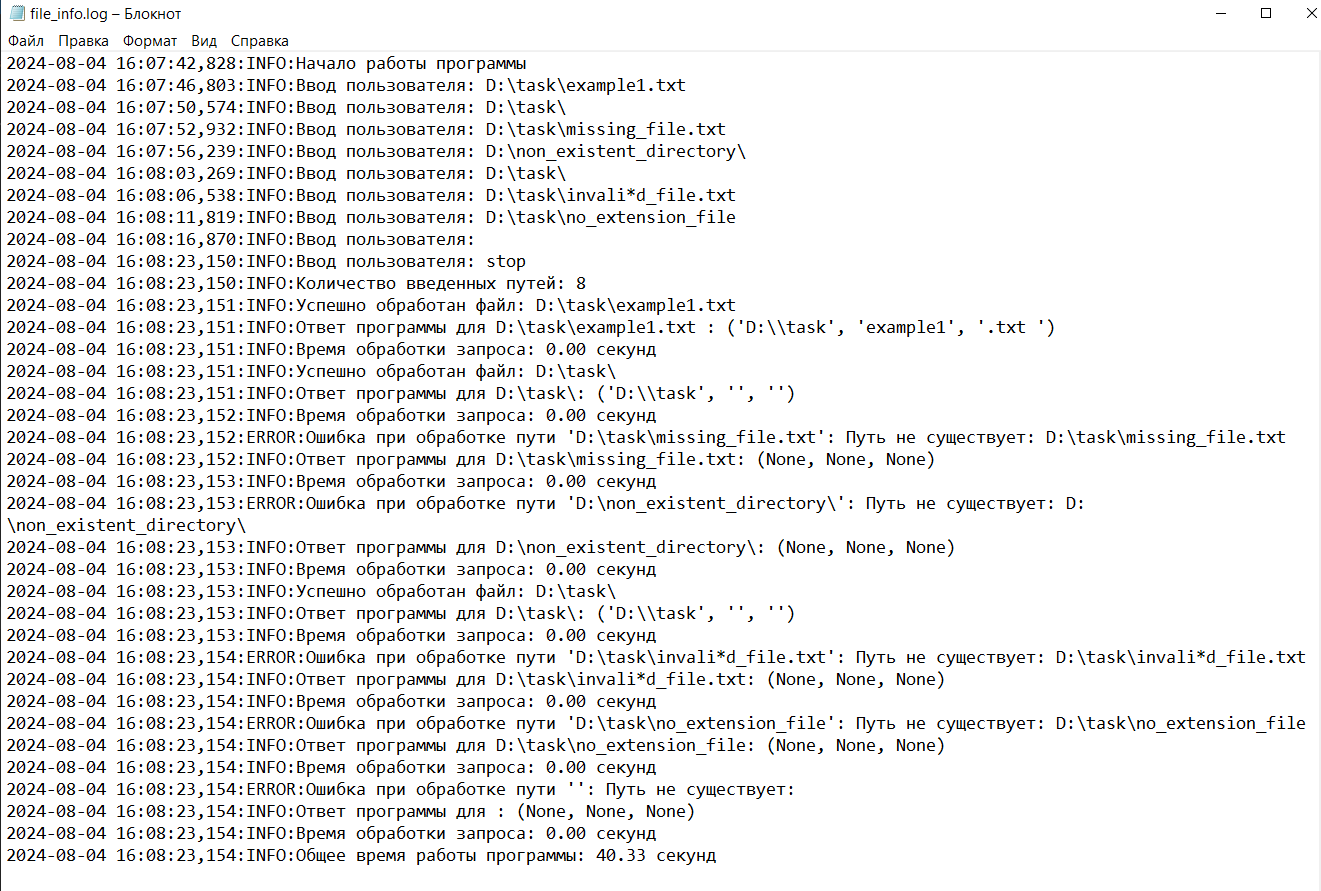
    user\_log.info(f"Общее время работы программы: {total\_time:.2f} секунд")

    print(f"Общее количество введенных путей: {total\_files}")

    print(f"Общее время работы программы: {total\_time:.2f} секунд")

if \_\_name\_\_ == "\_\_main\_\_":

    main()



2\*  
Урок 6 модули. Задание 1.

Проверка корректности даты

Вы работаете над разработкой программы для проверки корректности даты, введенной пользователем. На вход будет подаваться дата в формате "день.месяц.год". Ваша задача - создать программу, которая проверяет, является ли введенная дата корректной или нет.

Ваша программа должна предоставить ответ "True" (дата корректна) или "False" (дата некорректна) в зависимости от результата проверки.

import logging

from datetime import datetime

import time

# Настройка логирования

logging.basicConfig(filename='date\_validation.log', level=logging.DEBUG,

                    format='%(asctime)s:%(levelname)s:%(message)s')

def is\_valid\_date(date\_str):

    if not date\_str:

        logging.warning("Пустая строка введена.")

        return False

    if len(date\_str) != 10:

        logging.error(f"Некорректная длина строки для даты: {date\_str}")

        return False

    if date\_str.count('.') != 2:

        logging.error(f"Некорректный формат даты (ожидалось 'день.месяц.год'): {date\_str}")

        return False

    day, month, year = date\_str.split('.')

    if not (day.isdigit() and month.isdigit() and year.isdigit()):

        logging.error(f"Дата содержит недопустимые символы: {date\_str}")

        return False

    try:

        # Попытка преобразовать строку в объект datetime

        datetime.strptime(date\_str, "%d.%m.%Y")

        logging.info(f"Дата {date\_str} корректна.")

        return True

    except ValueError as e:

        logging.error(f"Ошибка проверки даты {date\_str}: {e}")

        return False

def main():

    logging.info("Программа запущена.")

    start\_program\_time = time.time()  # Время начала работы программы

    count = 0

    while True:

        date\_input = input("Введите дату в формате 'день.месяц.год' (или оставьте пустым для выхода): ")

        if date\_input.strip() == '':

            logging.info("Пользователь завершил программу.")

            print("Выход из программы.")

            break

        logging.info("///////////////////")

        start\_time = time.time()  # Время начала анализа даты

        logging.info(f"Начало анализа даты: {date\_input}")

        result = is\_valid\_date(date\_input)

        end\_time = time.time()  # Время окончания анализа даты

        elapsed\_time = end\_time - start\_time

        logging.info(f"Окончание анализа даты: {date\_input}. Время анализа: {elapsed\_time:.4f} секунд.")

        logging.info("///////////////////")

        count += 1

        print(f"Дата {date\_input} корректна: {result}")

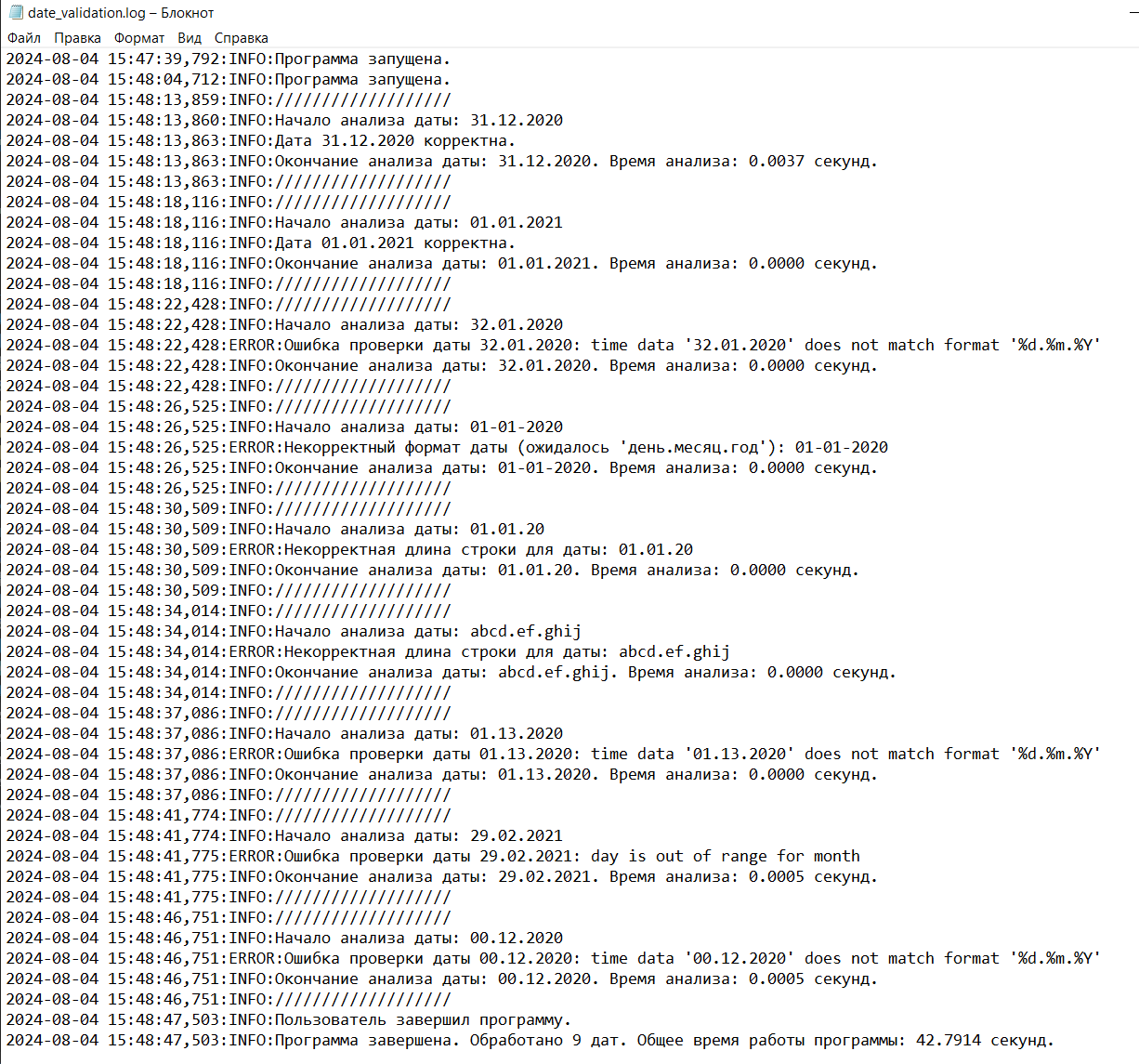
    end\_program\_time = time.time()  # Время окончания работы программы

    total\_elapsed\_time = end\_program\_time - start\_program\_time

    logging.info(f"Программа завершена. Обработано {count} дат. Общее время работы программы: {total\_elapsed\_time:.4f} секунд.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()



3\*

Урок 7.1 Функция группового переименования файлов

Напишите функцию группового переименования файлов в папке test\_folder под названием rename\_files. Она должна: a. принимать параметр желаемое конечное имя файлов desired\_name. При переименовании в конце имени добавляется порядковый номер.  
b. принимать параметр количество цифр в порядковом номере num\_digits.  
c. принимать параметр расширение исходного файла source\_ext . Переименование должно работать только для этих файлов внутри каталога.  
d. принимать параметр расширение конечного файла target\_ext.  
e. принимать диапазон сохраняемого оригинального имени. Например для диапазона [3, 6] берутся буквы с 3 по 6 из исходного имени файла. К ним прибавляется желаемое конечное имя, если оно передано. Далее счётчик файлов и расширение.  
f. Папка test\_folder доступна из текущей директории.

import os

import logging

from collections import namedtuple

# Настройка логирования

logging.basicConfig(filename='directory\_contents.log', level=logging.DEBUG,

                    format='%(asctime)s:%(levelname)s:%(message)s')

# Определение структуры данных

FileInfo = namedtuple('FileInfo', ['name', 'extension', 'is\_directory', 'parent\_directory'])

def collect\_directory\_info(directory\_path):

    """Собирает информацию о содержимом директории."""

    if not os.path.isdir(directory\_path):

        logging.error(f"Указанный путь не является директорией: {directory\_path}")

        print(f"Ошибка: '{directory\_path}' не является директорией.")

        return

    logging.info(f"Начинаем обработку директории: {directory\_path}")

    try:

        # Проход по всем каталогам и файлам в указанной директории

        for root, dirs, files in os.walk(directory\_path):

            logging.info(f"Обрабатываем каталог: {root}")

            for name in dirs:

                # Для каталогов

                dir\_path = os.path.join(root, name)

                parent\_dir = os.path.basename(root)

                file\_info = FileInfo(

                    name=name,

                    extension=None,

                    is\_directory=True,

                    parent\_directory=parent\_dir

                )

                logging.info(f"Каталог: {file\_info}")

            for name in files:

                # Для файлов

                file\_path = os.path.join(root, name)

                parent\_dir = os.path.basename(root)

                name\_without\_ext, ext = os.path.splitext(name)

                file\_info = FileInfo(

                    name=name\_without\_ext,

                    extension=ext[1:],  # убираем точку из расширения

                    is\_directory=False,

                    parent\_directory=parent\_dir

                )

                logging.info(f"Файл: {file\_info}")

    except PermissionError as e:

        logging.error(f"Ошибка доступа к директории '{directory\_path}': {e}")

    except FileNotFoundError as e:

        logging.error(f"Файл или директория не найдены при обработке '{directory\_path}': {e}")

    except OSError as e:

        logging.error(f"OS ошибка при обработке '{directory\_path}': {e}")

    except Exception as e:

        logging.error(f"Неизвестная ошибка при сборе информации из '{directory\_path}': {e}")

def main():

    while True:

        # Запрос директорий у пользователя

        directories\_input = input("Введите пути до директорий, разделенные пробелом (или нажмите Enter для выхода): ")

        # Проверка на пустую строку

        if not directories\_input.strip():

            print("Завершение программы.")

            break

        directories = directories\_input.split()

        # Обработка каждой директории

        for directory in directories:

            if os.path.isdir(directory):

                print(f"Обрабатывается директория: {directory}")

                collect\_directory\_info(directory)

            else:

                # Если директория не существует, записываем ошибку в лог

                logging.error(f"Указанный путь не является директорией: {directory}")

                print(f"Ошибка: '{directory}' не является директорией.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

