**ФГАОУ ВО «МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Лабораторная работа №6**

Файлы

**Вариант № 28**

По дисциплине:

Основы программирования

Выполнил студент 1-го курса группы 243-323

Онищенко А. А.

Проверил

\_\_\_\_\_\_\_\_\_\_\_Никишина И. Н.

**Москва, 2025**

**Задание 1 (корректировка лабораторной работы №1)**

**Постановка задачи**

Написать программу для расчета по формулам, входные данные берутся из файла, выходные записываются в файл. Предварительно подготовить тестовые кейсы в таблице Excel

Чтение и запись данных должна происходить из соответствующих файлов данных.

**Теоретическая часть**

**Работа с файлами**

Программа использует файловый ввод-вывод:

1. **Входные данные**:
   * Читаются из текстового файла
   * Каждая строка содержит одно значение угла в градусах
   * Поддерживается обработка ошибок:
     + Некорректные числовые значения
     + Пустые строки
2. **Выходные данные**:
   * Записываются в текстовый файл
   * Для каждого угла выводится:
     + Значение z₁ с точностью до 5 знаков после запятой
     + Значение z₂ с точностью до 5 знаков после запятой
   * Сообщения об ошибках для некорректных углов

**Особенности реализации**

1. **Используемые модули Python**:
   * math - для тригонометрических функций
   * Встроенные функции для работы с файлами
2. **Обработка ошибок**:
   * Проверка корректности числового ввода
   * Проверка допустимости углов
   * Запись ошибок в выходной файл
3. **Форматирование вывода**:
   * Использование символа градуса (°)
   * Выравнивание чисел по десятичной точке
   * Ограничение количества знаков после запятой

**Описание программы**

Программа написана на алгоритмическом языке Python 3.6, реализована в среде ОС Windows 10 и состоит из частей, отвечающих за ввод данных, вычисление и представление данных на экране монитора.

**Описание алгоритма**

1. **Инициализация математических функций**
2. **Подготовка файловых операций**
   * Определить функцию process\_angles(input\_file, output\_file)
   * Открыть файлы:
     + Входной файл (input\_file) в режиме чтения ('r')
     + Выходной файл (output\_file) в режиме записи ('w')
3. **Обработка входных данных**
   * Для каждой строки во входном файле:
     + Удалить пробелы/переносы (strip())
     + Попытаться преобразовать строку в число (float())
     + При ошибке преобразования:
       - Записать в выходной файл сообщение об ошибке формата
       - Перейти к следующей строке
4. **Тригонометрические вычисления**
5. **Завершение работы**
   * Закрыть файлы (автоматически при использовании with)
   * Вывести сообщение о завершении обработки

**Описание входных и выходных данных**

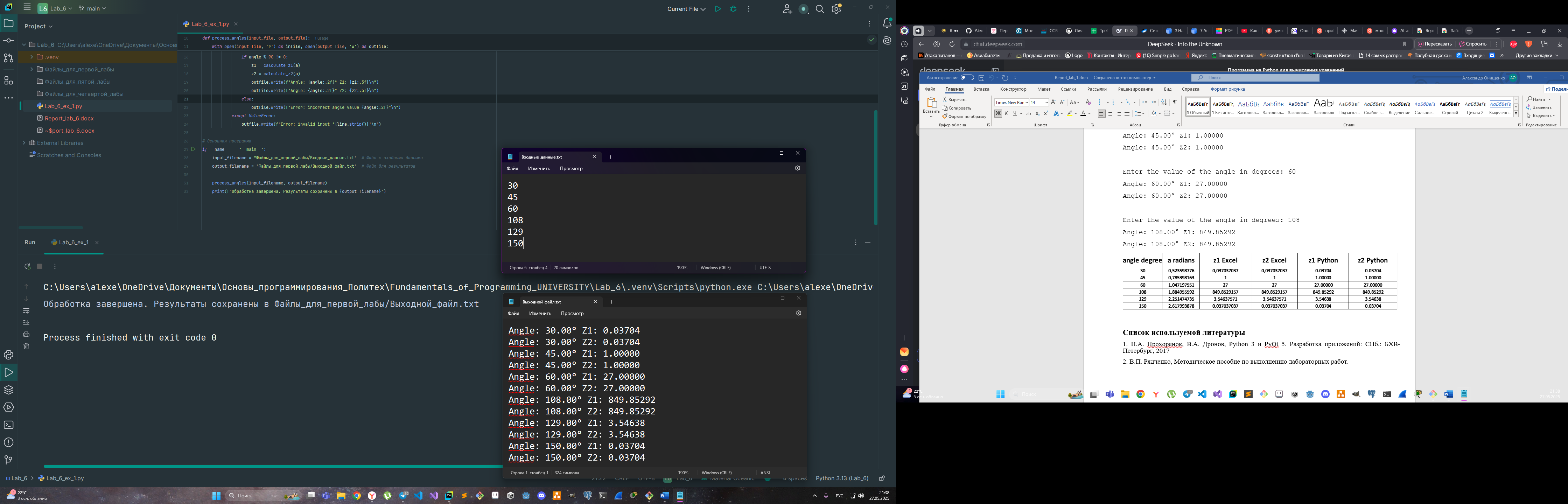
Входные данные поступают из файла «Входные\_данные.txt» в виде строк и преобразуются в вещественные числа. Далее выходные данные записываются в «Выходной\_файл.txt».

**Листинг программы**

*from* math *import* \*  
  
*def* calculate\_z1(a):  
 *return* (sin(a) \*\* 2 - tan(a) \*\* 2) / (cos(a) \*\* 2 - (1 / tan(a)) \*\* 2)  
  
*def* calculate\_z2(a):  
 *return* tan(a) \*\* 6  
  
  
*def* process\_angles(input\_file, output\_file):  
 *with open*(input\_file, 'r') *as* infile, *open*(output\_file, 'w') *as* outfile:  
 *for* line *in* infile:  
 *try*:  
 angle = *float*(line.strip())  
 a = radians(angle)  
 *if* angle % 90 != 0:  
 z1 = calculate\_z1(a)  
 z2 = calculate\_z2(a)  
 outfile.write(f"Angle: {angle:.2f}° Z1: {z1:.5f}\n")  
 outfile.write(f"Angle: {angle:.2f}° Z2: {z2:.5f}\n")  
 *else*:  
 outfile.write(f"Error: incorrect angle value {angle:.2f}°\n")  
 *except ValueError*:  
 outfile.write(f"Error: invalid input '{line.strip()}'\n")  
  
*# Основная программа  
if* \_\_name\_\_ == "\_\_main\_\_":  
 input\_filename = "Файлы\_для\_первой\_лабы/Входные\_данные.txt" *# Файл с входными данными* output\_filename = "Файлы\_для\_первой\_лабы/Выходной\_файл.txt" *# Файл для результатов* process\_angles(input\_filename, output\_filename)  
 *print*(f"Обработка завершена. Результаты сохранены в {output\_filename}")

**Результаты и тестовые кейсы**

****



**Задание 2 (корректировка лабораторной работы №4)**

**Постановка задачи**

С использованием модуля Random сформировать одномерный массив, состоящий из n вещественных элементов в котором элементы случайным образом принимают положительный или отрицательный знак и значение от -10 до 10. Для заданного числа amin < y < amax, вычислить:

1. Произведение элементов массива, значения модуля которого больше y.
2. Сумму модулей остальных элементов.

Чтение и запись данных должна происходить из соответствующих файлов данных.

**Теоретическая часть**

Данная программа выполняет обработку одномерного массива с использованием генерации случайных чисел, условных операторов, математических вычислений и файловых операций ввода и вывода.

**Описание программы**

Программа написана на алгоритмическом языке Python 3.6, реализована в среде ОС Windows 10 и состоит из частей, отвечающих за ввод данных, вычисление и представление данных на экране монитора.

**Описание алгоритма**

1. **Получение входных параметров**:
   * n (размер массива)
   * y (пороговое значение)
2. **Генерация массива**:
3. **Вычислительный блок**:
   * Разделение элементов на две группы
   * Вычисление произведения для |x| > y
   * Вычисление суммы модулей для |x| ≤ y
4. **Формат вывода**:
   * Оригинальный массив
   * Результаты вычислений
   * Сообщения об ошибках

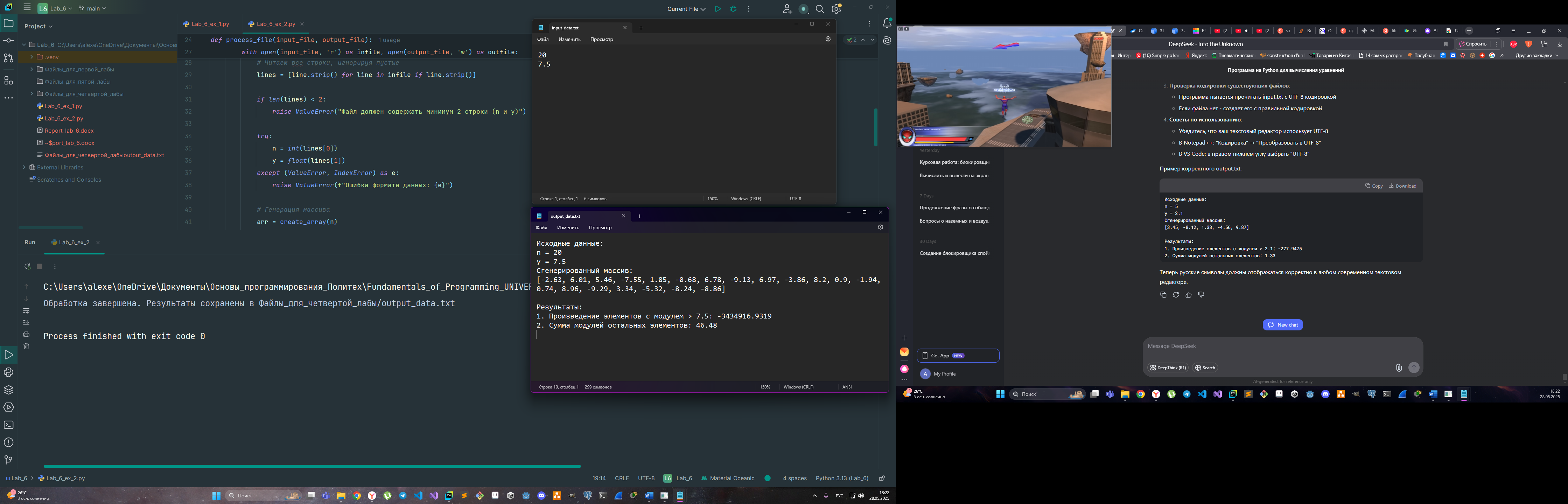
**Описание входных и выходных данных**

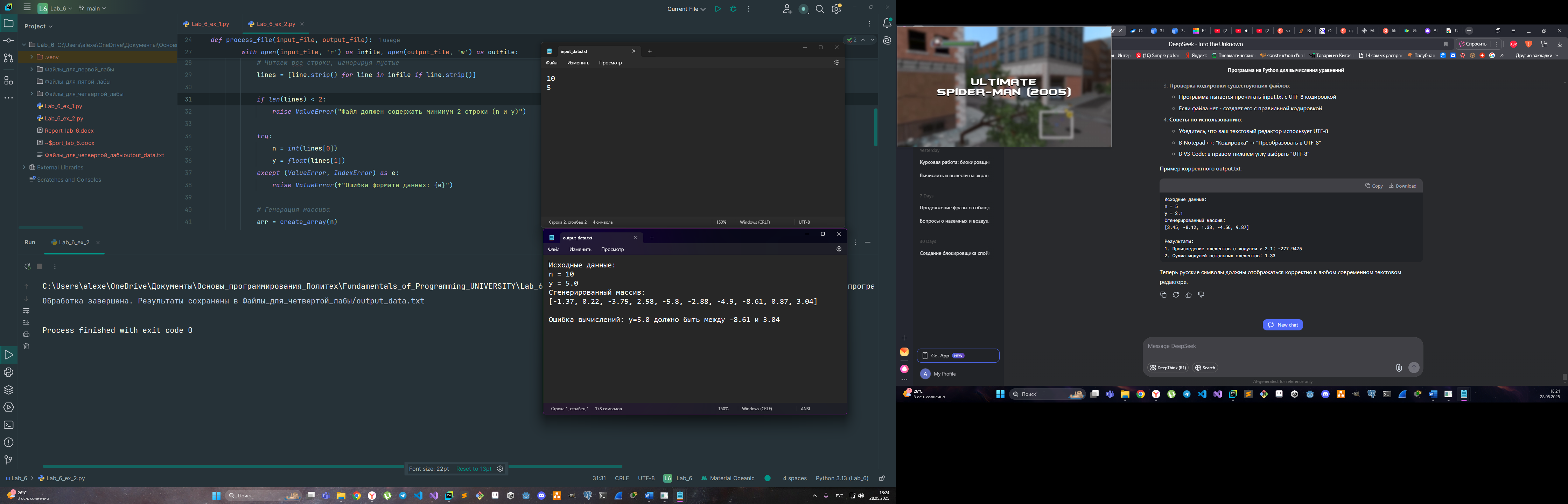
Входные данные поступают из файла «input\_data.txt» в виде строк и преобразуются в вещественные числа. Далее выходные данные записываются в «output\_data.txt».

**Листинг программы**

*from* random *import* uniform  
*import* sys  
  
*def* create\_array(n):  
 *"""Создает массив из n случайных чисел от -10 до 10"""  
 return* [*round*(uniform(-10, 10), 2) *for* \_ *in range*(n)] *# Округляем до 2 знаков для читаемости  
  
def* calculate(arr, y):  
 *"""Вычисляет произведение и сумму модулей с проверкой корректности y"""  
 if not* (*min*(arr) < y < *max*(arr)):  
 *raise ValueError*(f"y={y} должно быть между {*min*(arr)} и {*max*(arr)}")  
  
 product = 1  
 sum\_modules = 0  
  
 *for* num *in* arr:  
 *if abs*(num) > y:  
 product \*= num  
 *else*:  
 sum\_modules += *abs*(num)  
  
 *return round*(product, 4), *round*(sum\_modules, 4) *# Округляем результаты  
  
def* process\_file(input\_file, output\_file):  
 *"""Обрабатывает данные из файла и записывает результаты"""  
 try*:  
 *with open*(input\_file, 'r') *as* infile, *open*(output\_file, 'w') *as* outfile:  
 *# Читаем все строки, игнорируя пустые* lines = [line.strip() *for* line *in* infile *if* line.strip()]  
  
 *if len*(lines) < 2:  
 *raise ValueError*("Файл должен содержать минимум 2 строки (n и y)")  
  
 *try*:  
 n = *int*(lines[0])  
 y = *float*(lines[1])  
 *except* (*ValueError*, *IndexError*) *as* e:  
 *raise ValueError*(f"Ошибка формата данных: {e}")  
  
 *# Генерация массива* arr = create\_array(n)  
  
 *# Запись исходных данных* outfile.write(f"Исходные данные:\n")  
 outfile.write(f"n = {n}\n")  
 outfile.write(f"y = {y}\n")  
 outfile.write(f"Сгенерированный массив:\n{arr}\n\n")  
  
 *try*:  
 *# Вычисления* product, sum\_modules = calculate(arr, y)  
  
 *# Запись результатов* outfile.write("Результаты:\n")  
 outfile.write(f"1. Произведение элементов с модулем > {y}: {product}\n")  
 outfile.write(f"2. Сумма модулей остальных элементов: {sum\_modules}\n")  
 *except ValueError as* e:  
 outfile.write(f"Ошибка вычислений: {e}\n")  
  
 *except FileNotFoundError*:  
 *print*(f"Ошибка: файл {input\_file} не найден!", file=sys.stderr)  
 *except Exception as* e:  
 *print*(f"Неожиданная ошибка: {e}", file=sys.stderr)  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 input\_filename = 'Файлы\_для\_четвертой\_лабы/input\_data.txt'  
 output\_filename = 'Файлы\_для\_четвертой\_лабы/output\_data.txt'  
  
 process\_file(input\_filename, output\_filename)  
 *print*(f"Обработка завершена. Результаты сохранены в {output\_filename}")

**Результаты работы программы**





**Задание 3 (корректировка лабораторной работы №5)**

**Постановка задачи**

Произведением двух матриц Amn на Bnl называется такая матрица Cml, для которой:

То есть элемент матрицы С равен сумме произведений элементов i-й строки матрицы A на соответствующие элементы k-го столбца матрицы B.

Написать программу вычисления произведения двух матриц.

Программа должна по заданным размерностям матриц сообщать о возможности получения такого произведения.

Чтение и запись данных должна происходить из соответствующих файлов данных.

**Теоретическая часть**

Двумерный массив (матрица) — это структура данных, представляющая собой массив массивов, где каждый элемент имеет два индекса: номер строки и номер столбца. В Python двумерные массивы чаще всего реализуются как:

* Списки списков (list of lists)
* Массивы из библиотеки NumPy
* Матрицы из специализированных библиотек (SciPy, Pandas)

**Описание программы**

Программа написана на алгоритмическом языке Python 3.6, реализована в среде ОС Windows 10 и состоит из частей, отвечающих за ввод данных, вычисление и представление данных на экране монитора.

**Описание алгоритма**

1. Инициализация и подготовка
2. Чтение данных из файла
3. Проверка возможности умножения
4. Умножение матриц
5. Запись результатов
6. Обработка ошибок
7. Завершение работы

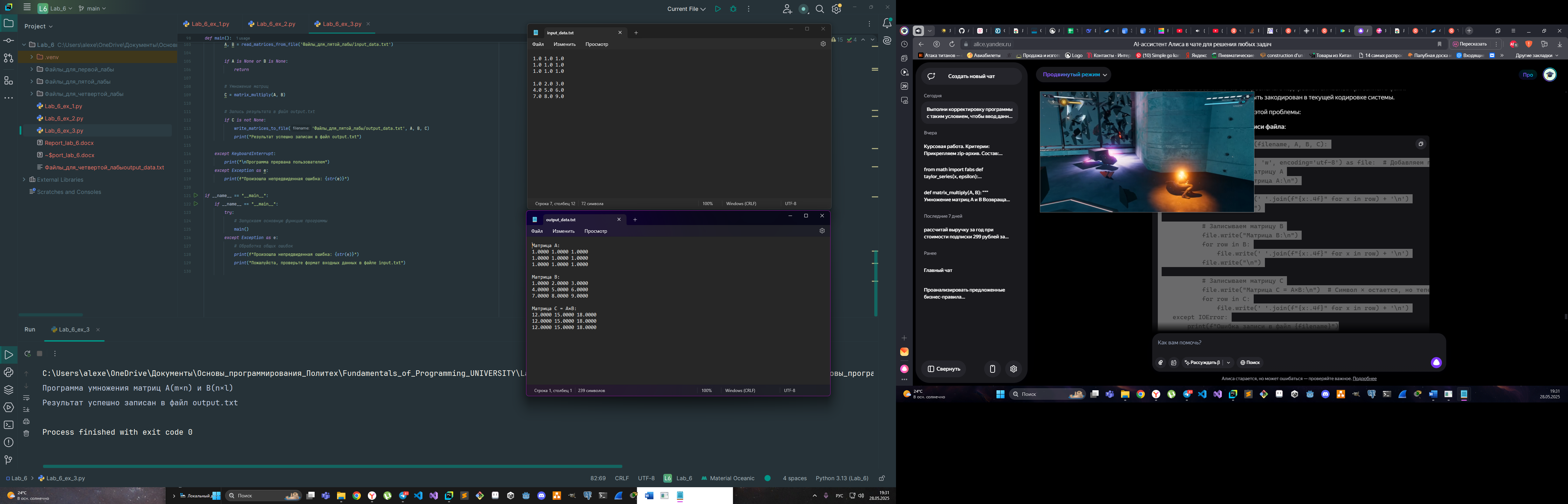
**Описание входных и выходных данных**

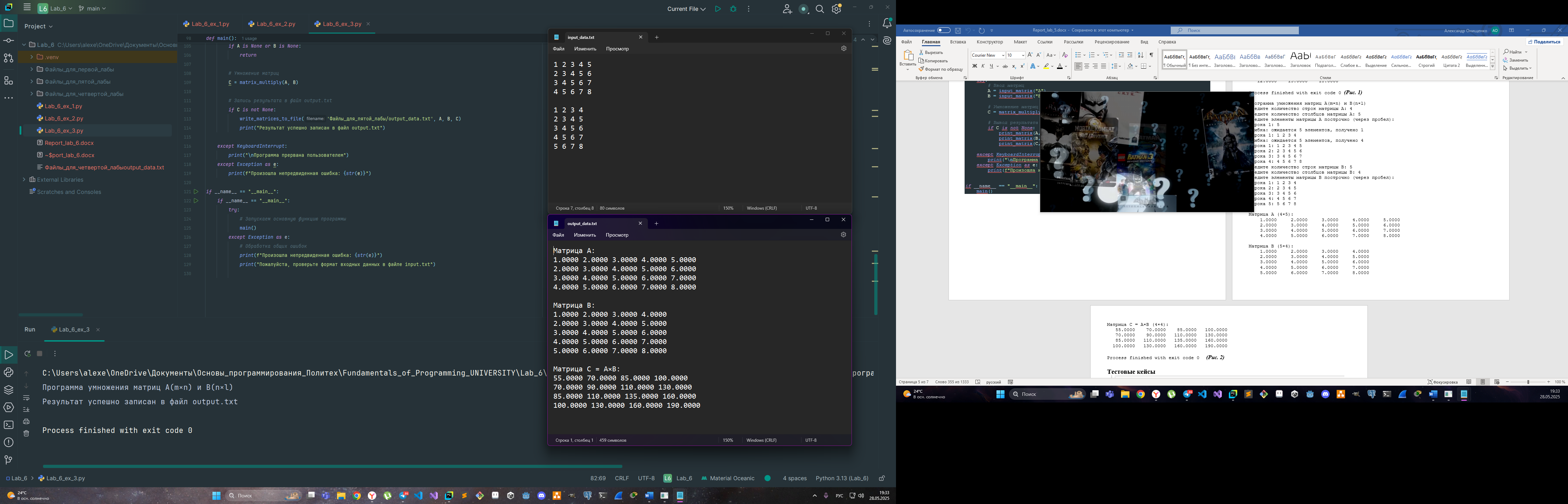
Входные данные поступают из файла «input\_data.txt» в виде строк и преобразуются в вещественные числа. Далее выходные данные записываются в «output\_data.txt

**Листинг программы**

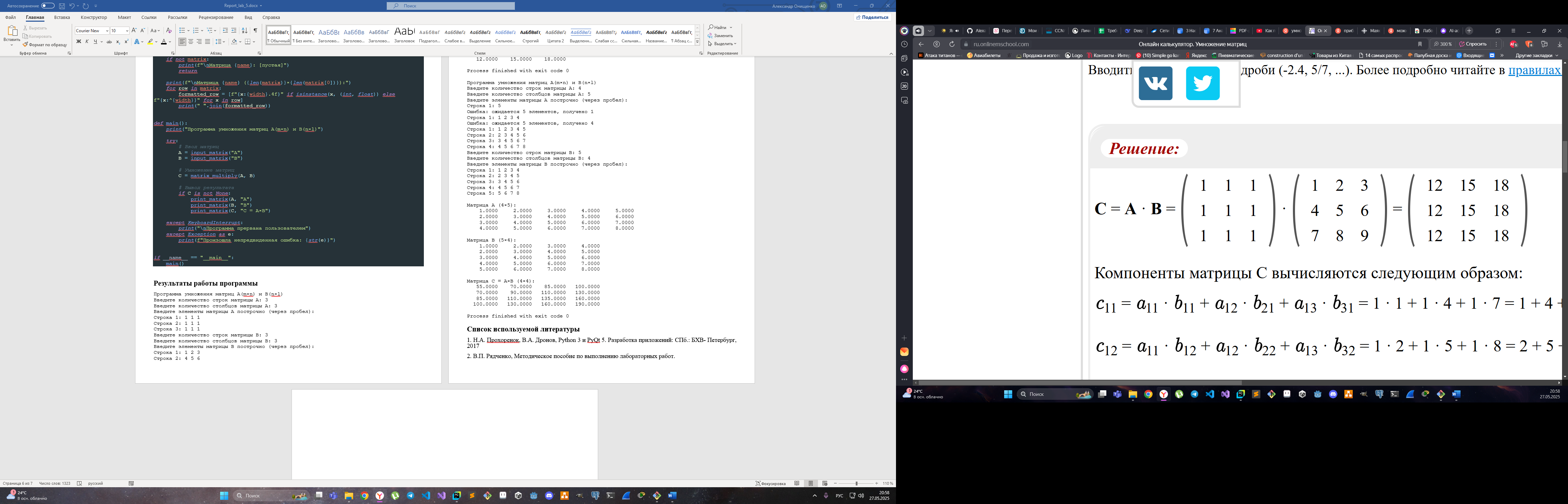
*def* matrix\_multiply(A, B):  
 *"""  
 Умножение матриц A и B  
 Возвращает матрицу-произведение или None, если умножение невозможно  
 """  
 # Проверка на пустые матрицы  
 if not* A *or not* B:  
 *print*("Ошибка: одна из матриц пустая")  
 *return None  
  
 # Проверка возможности умножения* cols\_A = *len*(A[0])  
 rows\_B = *len*(B)  
  
 *if* cols\_A != rows\_B:  
 *print*(f"Ошибка: невозможно умножить ({*len*(A)}×{cols\_A}) на ({rows\_B}×{*len*(B[0])})")  
 *return None  
  
 # Проверка согласованности размеров  
 for* row *in* A:  
 *if len*(row) != cols\_A:  
 *print*("Ошибка: несоответствие размеров в матрице A")  
 *return None  
  
 for* row *in* B:  
 *if len*(row) != *len*(B[0]):  
 *print*("Ошибка: несоответствие размеров в матрице B")  
 *return None  
  
 # Создание результирующей матрицы* m = *len*(A)  
 n = *len*(B[0])  
 C = [[0.0 *for* \_ *in range*(n)] *for* \_ *in range*(m)]  
  
 *# Транспонируем B для более эффективного доступа по столбцам* B\_transposed = *list*(*zip*(\*B))  
  
 *# Вычисление произведения с оптимизированным доступом  
 for* i *in range*(m):  
 *for* j *in range*(n):  
 *# Используем sum и генератор для вычисления скалярного произведения* C[i][j] = *sum*(A[i][k] \* B\_transposed[j][k] *for* k *in range*(cols\_A))  
  
 *return* C  
  
*def* read\_matrices\_from\_file(filename):  
 *"""Чтение двух матриц из файла input.txt"""  
 try*:  
 *with open*(filename, 'r') *as* file:  
 *# Читаем первую матрицу* matrix\_A = []  
 *while True*:  
 line = file.readline().strip()  
 *if not* line: *# Пустая строка между матрицами  
 break* row = [*float*(x) *for* x *in* line.split()]  
 matrix\_A.append(row)  
  
 *# Читаем вторую матрицу* matrix\_B = []  
 *for* line *in* file:  
 row = [*float*(x) *for* x *in* line.strip().split()]  
 matrix\_B.append(row)  
  
 *return* matrix\_A, matrix\_B  
  
 *except FileNotFoundError*:  
 *print*(f"Ошибка: файл {filename} не найден")  
 *return None*, *None  
 except ValueError*:  
 *print*("Ошибка: все элементы должны быть числами!")  
 *return None*, *None  
  
def* write\_matrices\_to\_file(filename, A, B, C):  
 *try*:  
 *with open*(filename, 'w', encoding='utf-8') *as* file: *# Добавляем параметр encoding  
 # Записываем матрицу A* file.write("Матрица A:\n")  
 *for* row *in* A:  
 file.write(' '.join(f"{x:.4f}" *for* x *in* row) + '\n')  
 file.write("\n")  
  
 *# Записываем матрицу B* file.write("Матрица B:\n")  
 *for* row *in* B:  
 file.write(' '.join(f"{x:.4f}" *for* x *in* row) + '\n')  
 file.write("\n")  
  
 *# Записываем матрицу C* file.write("Матрица C = A×B:\n")   
 *for* row *in* C:  
 file.write(' '.join(f"{x:.4f}" *for* x *in* row) + '\n')  
 *except IOError*:  
 *print*(f"Ошибка записи в файл {filename}")  
  
*def* main():  
 *print*("Программа умножения матриц A(m×n) и B(n×l)")  
  
 *try*:  
 *# Чтение матриц из файла input.txt* A, B = read\_matrices\_from\_file('Файлы\_для\_пятой\_лабы/input\_data.txt')  
  
 *if* A *is None or* B *is None*:  
 *return  
  
 # Умножение матриц* C = matrix\_multiply(A, B)  
  
 *# Запись результата в файл output.txt  
 if* C *is not None*:  
 write\_matrices\_to\_file('Файлы\_для\_пятой\_лабы/output\_data.txt', A, B, C)  
 *print*("Результат успешно записан в файл output.txt")  
  
 *except KeyboardInterrupt*:  
 *print*("\nПрограмма прервана пользователем")  
 *except Exception as* e:  
 *print*(f"Произошла непредвиденная ошибка: {*str*(e)}")  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 *if* \_\_name\_\_ == "\_\_main\_\_":  
 *try*:  
 *# Запускаем основную функцию программы* main()  
 *except Exception as* e:  
 *# Обработка общих ошибок  
 print*(f"Произошла непредвиденная ошибка: {*str*(e)}")  
 *print*("Пожалуйста, проверьте формат входных данных в файле input.txt")

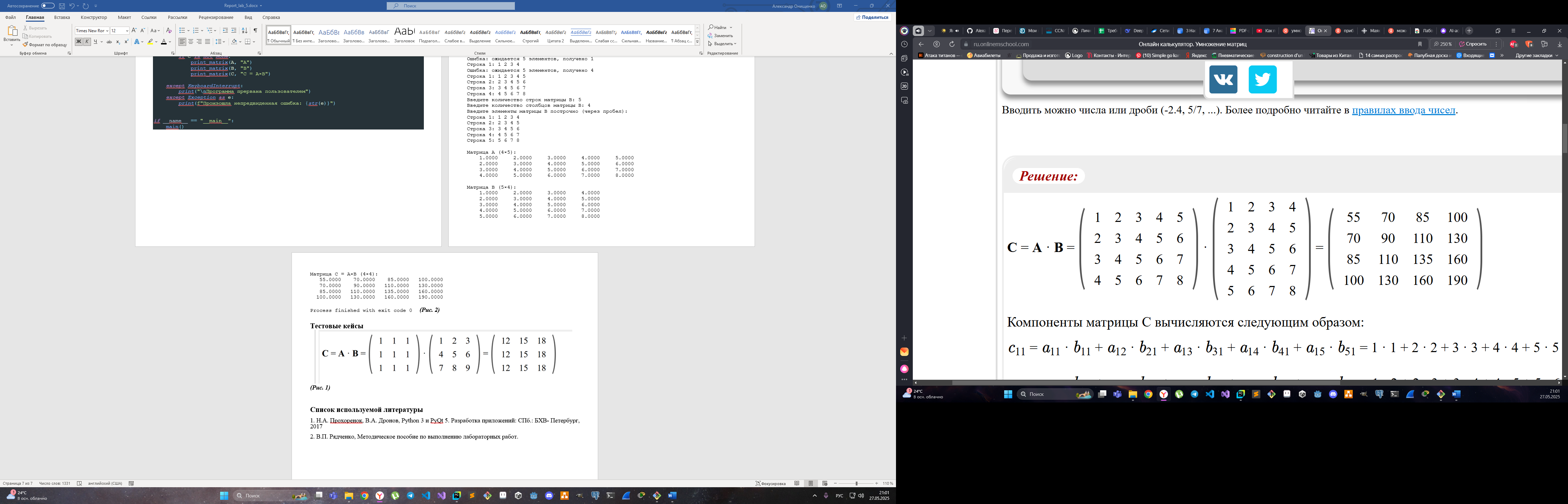
**Результаты работы программы**





**Тестовые кейсы**

***(Рис. 1)***



***(Рис. 2)***

**Список используемой литературы**

1. Н.А. Прохоренок, В.А. Дронов, Python 3 и PyQt 5. Разработка приложений: СПб.: БХВ- Петербург, 2017

2. В.П. Рядченко, Методическое пособие по выполнению лабораторных работ.