Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий и систем связи

**Лабораторная работа №1**

**Вариант №1**

Выполнил(и:)

Алексеев Т.Ю.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc115913159)

[1. ЗАДАНИЕ 2 4](#_Toc115913160)

[1.1 Описание функции транспонирования матрицы 4](#_Toc115913161)

[1.1 Описание функции для умножения матриц 5](#_Toc115913162)

[1.2 Описание функции определения ранга матрицы 6](#_Toc115913163)

[2. ЗАДАНИЕ 3 7](#_Toc115913164)

[2.1 Описание функции транспонирования матриц при помощи NumPy 7](#_Toc115913165)

[2.2 Описание функции умножения матриц при помощи NumPy 8](#_Toc115913166)

[2.3 Описание функции определения ранга матрицы при помощи Numpy 9](#_Toc115913167)

[2.4 Достоинства и недостатки библиотеки NumPy 10](#_Toc115913168)

[3. ЗАДАНИЕ 4 11](#_Toc115913169)

[3.1 Описание функции возведение матрицы в -1 степень 11](#_Toc115913170)

[3.2 Описание функции возведения матрицы в -1 степень при помощи NumPy 13](#_Toc115913171)

[3.3 Сравнение времени работы самостоятельно написанного алгоритма возведения матрицы в -1 степень и алгоритма из библиотеки NumPy 14](#_Toc115913172)

[СПИСОК ЛИТЕРАТУРЫ 15](#_Toc115913173)

# ВВЕДЕНИЕ

Целью данной работы являлось знакомство с языком программирования Python, обучение работы с матрицами и знакомство с библиотеками NumPy и timeit.

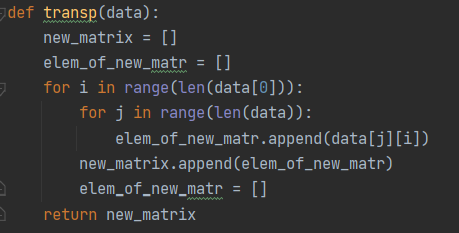
Для достижения данной цели необходимо было выполнить следующие задания:

1. Задание 2: создание программы на языке Python, обладающей возможностями транспонирования матрицы, умножения матриц и определения ранга матриц.
2. Задание 3: выполнить Задание 2 с помощью NumPy и оценить достоинства и недостатки данной библиотеки.
3. Задание 4: написать программу для возведения матрицы в -1 степень и сравнить время её работы с аналогом из библиотеки NumPy.

# 1. ЗАДАНИЕ 2

# 1.1 Описание функции транспонирования матрицы

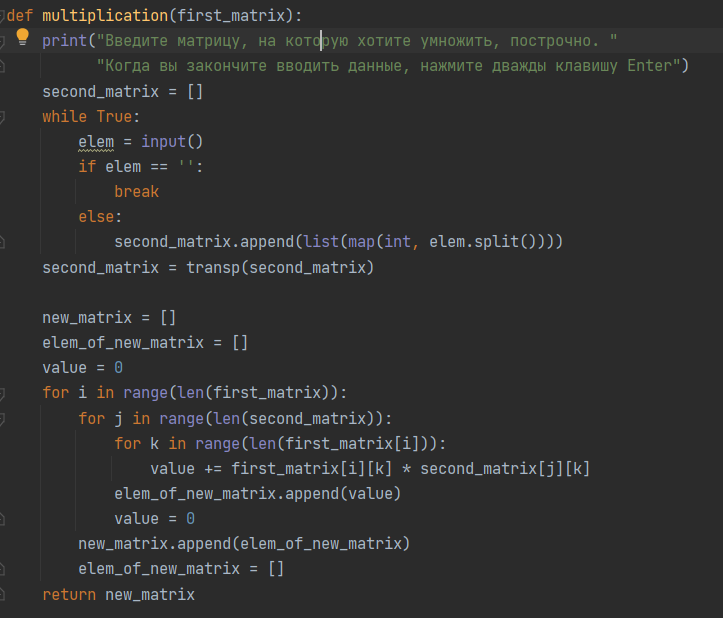
Алгоритм транспонирования матрицы мною был реализован через создание новой матрицы. Во вложенных циклах for в список elem\_of\_new\_matr вносились значения столбцов матрицы. После завершения внесения в него очередного столбца список добавлялся в новую матрицу как строка. Таким образом, за счёт перевода столбцов старой матрицы в строки новой выполняется транспонирование матрицы.



Приложение 1 – Содержание функции transp

# Описание функции для умножения матриц

Так как операция умножения матриц бинарная, то в начале функции представлено считывание второй матрицы. Также, для более простого перебора была использована функция transp, описанная ранее. В переменную value записывается сумма произведений соответствующих элементов в i-ой строке из первой матрицы и в j-ой строке из второй матрицы. После этого переменная добавляется в elem\_of\_new\_matrix – i-ю строку в результирующей матрице. Таким образом, функция возвращает результат умножения двух матриц.



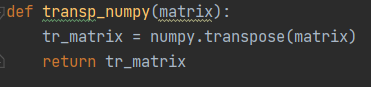
Приложение 2 – Содержание функции multiplication

# Описание функции определения ранга матрицы

# 2. ЗАДАНИЕ 3

# 2.1 Описание функции транспонирования матриц при помощи NumPy

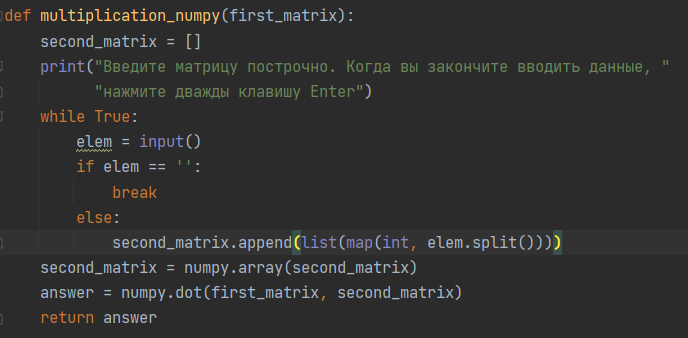
Библиотека NumPy помогает выполнять сложные операции с матрицами буквально в одну строку. Но переменная matrix перед этим должна быть преобразована с помощью numpy.array(matrix) в массив типа numpy. А само транспонирование выполняется с помощью функции numpy.transpose(matrix).



Приложение 4 – Содержание функции transp\_numpy

# 2.2 Описание функции умножения матриц при помощи NumPy

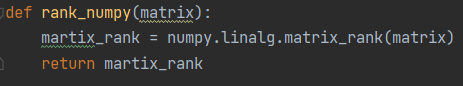
Так как операция умножения матриц бинарная, то опять начинаем со считывания второй матрицы. Её также необходимо привести к массиву вида numpy с помощью numpy.array(second\_matrix). Умножение выполняется с помощью функции numpy.dot(first\_matrix, second\_matrix).



Приложение 5 – Содержание функции multiplication\_numpy

# 2.3 Описание функции определения ранга матрицы при помощи Numpy

На вход функции подаётся переменная matrix – массив типа numpy. Сама операция определения ранга матрицы выполняется с помощью функции numpy.linalg.matrix\_rank(matrix).



Приложение 6 – Содержание функции rank\_numpy

# 2.4 Достоинства и недостатки библиотеки NumPy

Достоинства:

1. Возможность выполнить сложные математические операции с матрицами буквально одной строчкой кода.
2. В библиотеке представлены максимально оптимизированные алгоритмы, прошедшие множество проверок и правок сообщества программистов.

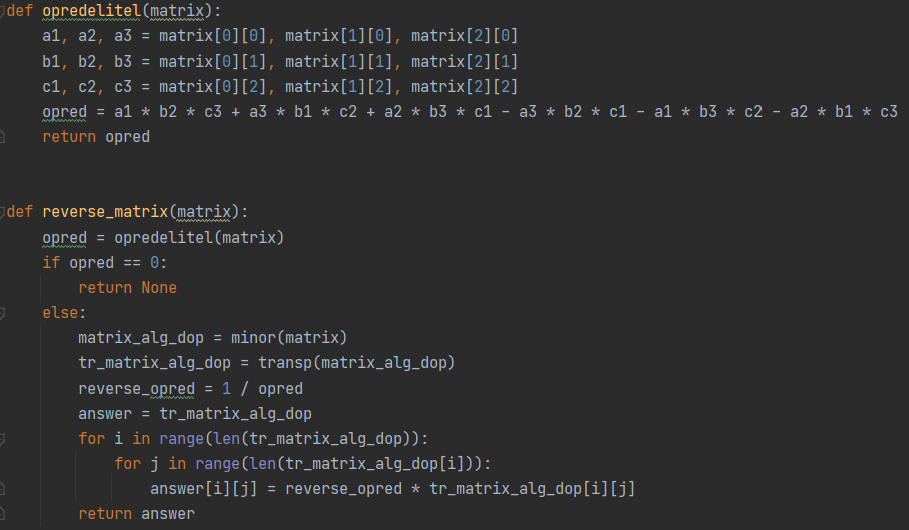
Недостатки:

1. Нет общего понимания алгоритма и нет возможности вносить свои правки в алгоритм выполнения операции, так как он закрыта и представляет собой лишь работающую функцию.
2. NumPy работает с матрицами, которые преобразованы в массивы типа numpy. Это может создать проблемы при дальнейшей самостоятельной работе с ними.

# 3. ЗАДАНИЕ 4

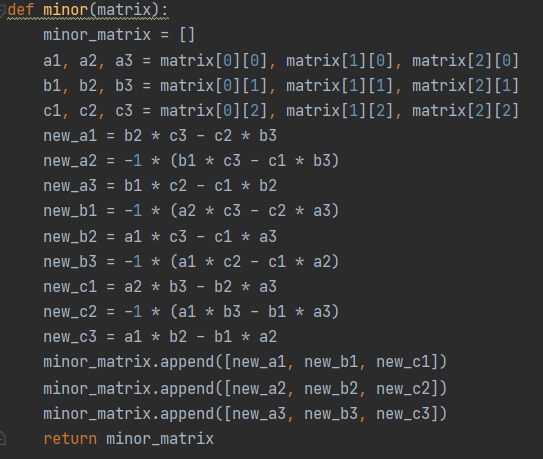
# 3.1 Описание функции возведение матрицы в -1 степень

Основная функция reverse\_matrix начинается с того, что нам нужно узнать определитель исходной матрицы. Он определяется с помощью функции opredelitel(matrix), подставляя значения в формулу определителя для матрицы 3х3. Если определитель равен 0, то для исходной матрицы не существует обратной, возвращаем None и завершаем работу функции.



Приложение 7 – Содержание функций reverse\_matrix и opredelitel

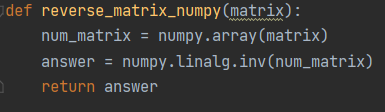
Далее с помощью функции minor(matrix) находим матрицу миноров и матрицу алгебраических дополнений matrix\_alg\_dop. После этого с помощью функции transp(matrix), описанной ранее, получаем транспонированную матрицу алгебраических дополнений tr\_matrix\_alg\_dop. Чтобы получить ответ, нам необходимо умножить каждый элемент матрицы на число, обратное определителю. В итоге мы получаем матрицу в -1 степени.



Приложение 8 – Содержание функции minor

# 3.2 Описание функции возведения матрицы в -1 степень при помощи NumPy

На вход функции подаётся матрица, которая в первой строке функции становится массивом типа numpy. Получение матрицы в -1 степени происходит с помощью функции numpy.linalg.inv(matrix).



Приложение 9 – Содержание функции reverse\_matrix\_numpy

# 3.3 Сравнение времени работы самостоятельно написанного алгоритма возведения матрицы в -1 степень и алгоритма из библиотеки NumPy

Проводить сравнения времени мы будем с помощью библиотеки timeit. Проведём 4 теста и попробуем вычислить закономерность. Ниже представлена таблица с замерами.

|  |  |  |
| --- | --- | --- |
| № теста | Собственный алгоритм | Алгоритм NumPy |
| 1 | 0.0002716999999998748 | 0.0000306000000001028 |
| 2 | 0.0002501999999999782 | 0.0000296999999997993 |
| 3 | 0.00031060000000060484 | 0.0000239000000004097 |
| 4 | 0.0003543999999990888 | 0.0000245999999997082 |

Таблица 1 – Результаты замеров времени выполнения

Выполнив замеры времени выполнения, можно сказать, что в среднем самостоятельно написанный алгоритм в 11, 2 раза медленнее алгоритма из библиотеки NumPy. Это достаточно значительный перевес, который показывает превосходство алгоритма из библиотеки NumPy.

# СПИСОК ЛИТЕРАТУРЫ

1. NumPy: матрицы и операции над ними // URL: <http://cs.mipt.ru/advanced_python/lessons/lab16.html#section-14> (дата обращения: 05.10.2022)
2. Как найти обратную матрицу? // URL: <http://mathprofi.ru/kak_naiti_obratnuyu_matricu.html> (дата обращения: 05.10.2022)
3. Модуль timeit в Python: как работает с примерами // URL: <https://pythonim.ru/moduli/timeit-python> (дата обращения: 05.10.2022)