

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

Факультет инфокоммуникационных технологий

Лабораторная работа №1

Выполнил:

Шишминцев Д. В.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. ЗАДАНИЕ 2	4
1.1 Интерфейс программы	4
1.2 Транспонирование матрицы	4
1.3 Умножение матриц	5
1.4 Ранг матрицы	6
2. ЗАДАНИЕ 3	8
2.1 Реализация второго задания с использованием библиотеки NumPy ..	8
2.2 Достоинства и недостатки использования NumPy	8
2. ЗАДАНИЕ 4	9
3.1 Возведение матрицы в степень -1	9
3.2 Сравнение быстродействия	10
ВЫВОД	11
СПИСОК ЛИТЕРАТУРЫ	12
ПРИЛОЖЕНИЕ	13

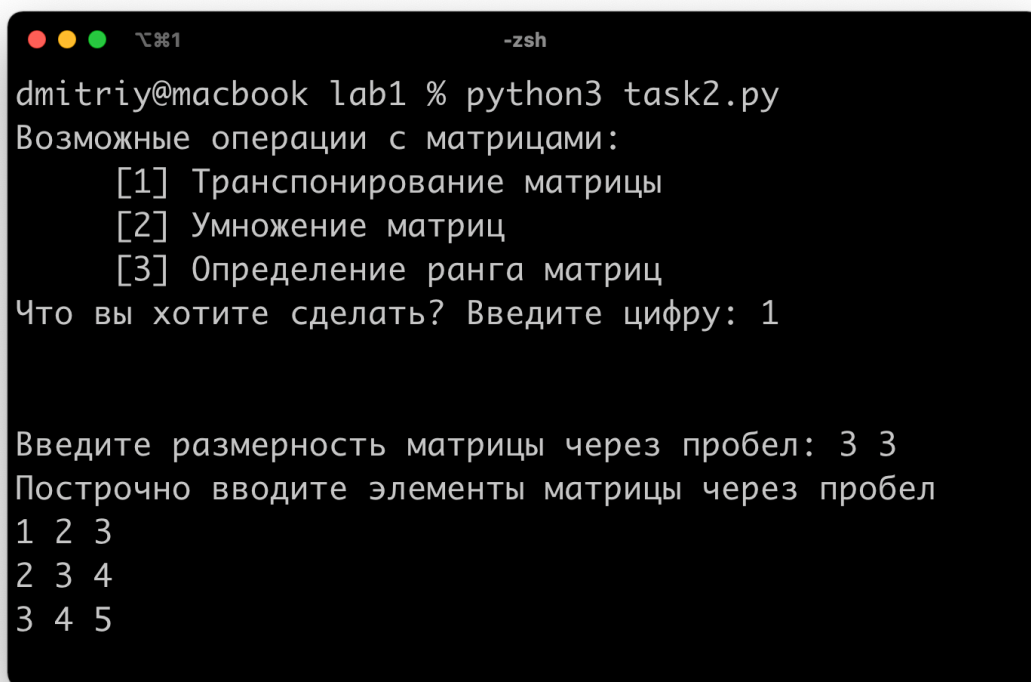
ВВЕДЕНИЕ

Цель данной работы – знакомство с языком программирования Python. В процессе выполнения поставленных задач на операции с матрицами будут задействованы базовые конструкции языка программирования, а также одна из популярных библиотек.

1. ЗАДАНИЕ 2

1.1 Интерфейс программы

В качестве интерфейса используется интерфейс командной строки. Пользователь выбирает операцию над матрицами вводом числа. Задается размерность матрицы, после чего ввод производится построчно. Для хранения и работы с матрицами используются вложенные списки. Интерфейс программы демонстрируется ниже. (Рисунок 1)



```
dmity@macbook lab1 % python3 task2.py
Возможные операции с матрицами:
    [1] Транспонирование матрицы
    [2] Умножение матриц
    [3] Определение ранга матриц
Что вы хотите сделать? Введите цифру: 1

Введите размерность матрицы через пробел: 3 3
Построчно вводите элементы матрицы через пробел
1 2 3
2 3 4
3 4 5
```

Рисунок 1

Данный вид интерфейса удобен для этой программы и прост в реализации.

1.2 Транспонирование матрицы

Первая задача – реализовать функцию для транспонирования матриц.

Транспонирование — в линейной алгебре это операция над матрицами в результате которой матрица поворачивается относительно своей главной диагонали. При этом столбцы исходной матрицы становятся строками

результатирующей. Для реализации использовалось два вложенных цикла. Элементы многомерного списка перебирались и добавлялись в новый список. Функция возвращает многомерный список. Реализация представлена ниже. (Фрагмент кода 1)

```
def transpose(m):  
    new_matrix = []  
    for i in range(0, len(m[0])):  
        new_matrix.append([])  
        for q in range(0, len(m)):  
            new_matrix[i].append(m[q][i])  
    return new_matrix
```

Фрагмент кода 1

1.3 Умножение матриц

Вторая задача – реализовать функцию умножения двух матриц. Умножение матриц – одна из основных операций над матрицами. Матрицы могут быть перемножены если число строк матрицы A равно числу столбцов матрицы B. Принцип умножения демонстрируется на рисунке ниже. (Рисунок 2)

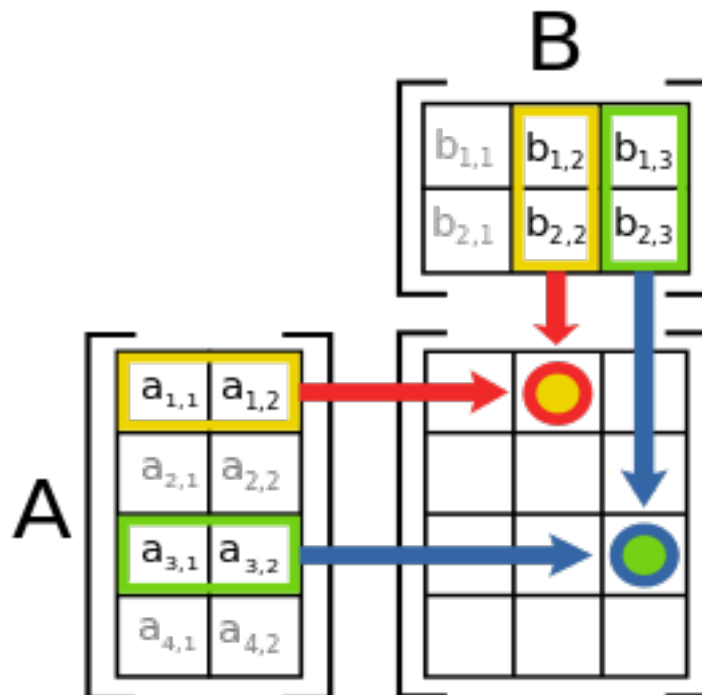


Рисунок 2

Для реализации алгоритма используется 3 вложенных цикла, а так же дополнительная функция `getCollumn` возвращающая столбец матрицы в виде списка. Функция `multiplication` возвращает результирующую матрицу в виде многомерного списка. Реализация представлена ниже. (Фрагмент кода 2)

```
def getCollumn(m, index):
    collumn = []
    for i in range(0, len(m)):
        collumn.append(m[i][index])
    return collumn

def multiplication(a, b):
    if len(a[0]) != len(b):
        return False
    new_matrix = []

    for i in range(0, len(a)):
        new_matrix.append([])

        for q in range(0, len(b[0])):
            row = a[i]
            collumn = getCollumn(b, q)
            element = 0
            for p in range(0, len(row)):
                element += row[p] * collumn[p]
            new_matrix[i].append(element)

    return new_matrix
```

Фрагмент кода 2

1.4 Ранг матрицы

Третьей задачей было написать функцию нахождения ранга матрицы. Рангом матрицы A с m строками и n столбцами называется максимальное число линейно независимых строк (столбцов). Для реализации алгоритма используется метод элементарный преобразований (Метод Гаусса). Написанная функция принимает в качестве аргумента матрицу в виде многомерного списка, а возвращает число – ранг матрицы.

Функция имеет следующий порядок действий:

1. Из многомерного списка (далее - матрицы) удаляются строки содержащие в себе только нули.
2. Перебирается каждая строка матрицы
 - a. Находиться первый ненулевой элемент в строке
 - b. Строка делится на найденный элемент
 - c. Для каждой последующей строки находится коэффициент k отношения элемента строки с индексом p к элементу строки с индексом r .
 - d. Из последующих строк вычитается строка с индексом r умноженная на коэффициент k .
3. В полученной матрице считается количество ненулевых строк, что и является рангом матрицы.

Реализация представлена ниже (Фрагмент кода 3)

```
def rank(matrix):  
    matrix_copy = matrix.copy()  
    for row in matrix_copy:  
        if row.count(0) == len(row):  
            matrix.remove(row)  
    for r in range(len(matrix)):  
        div = 1  
        for i in range(len(matrix[r])):  
            if matrix[r][i] != 0:  
                div = matrix[r][i]  
                break  
        for q in range(len(matrix[r])):  
            matrix[r][q] = matrix[r][q] / div  
        for p in range(r+1, len(matrix)):  
            k = matrix[p][i] / matrix[r][i]  
            for n in range(len(matrix[p])):  
                matrix[p][n] = matrix[p][n] -  
k*matrix[r][n]  
  
    matrix_rank = len(matrix)  
  
    for row in matrix:  
        if row.count(0) == len(row):  
            matrix_rank = matrix_rank - 1  
    return matrix_rank
```

Фрагмент кода 3

2. ЗАДАНИЕ 3

2.1 Реализация второго задания с использованием библиотеки NumPy

Используя популярную Python библиотеку NumPy были реализованы все те же функции (Транспонирование, умножение, определение ранга матрицы).

Код представлен ниже. (Фрагмент кода 4)

```
np.array(ioMatrix.inputMatrix()).T

np.dot(
    np.array(ioMatrix.inputMatrix()),
    np.array(ioMatrix.inputMatrix()))

np.linalg.matrix_rank(ioMatrix.inputMatrix())
```

Фрагмент кода 4

2.2 Достоинства и недостатки использования NumPy

Библиотека NumPy имеет широкий функционал. Доступно множество математических алгоритмов, работа с многомерными массивами. Реализация тех или иных задач с использованием библиотеки NumPy легка и не занимает много времени. Алгоритмы работают быстро, библиотека написана с использованием Си и Фортрана.

Недостатком может быть то, что разработчик зачастую не понимает что скрывается за готовыми алгоритмами NumPy, что может негативно повлиять на процесс изучения программирования.

2 ЗАДАНИЕ 4

3.1 Возведение матрицы в степень -1

Для реализации функции получения обратной матрицы была написана вспомогательная функция `det` принимающая многомерный список на вход и возвращающая число – определитель матрицы. Определитель высчитывается по формуле для матрицы размерности 3x3.

Используя вложенные циклы перебираются все миноры матрицы и вычисляются алгебраические дополнения. Полученный многомерный массив домножается на $(1/d)$, где d – определитель полученный с помощью функции `det`. Реализация алгоритма представлена ниже. (Фрагмент кода 5)

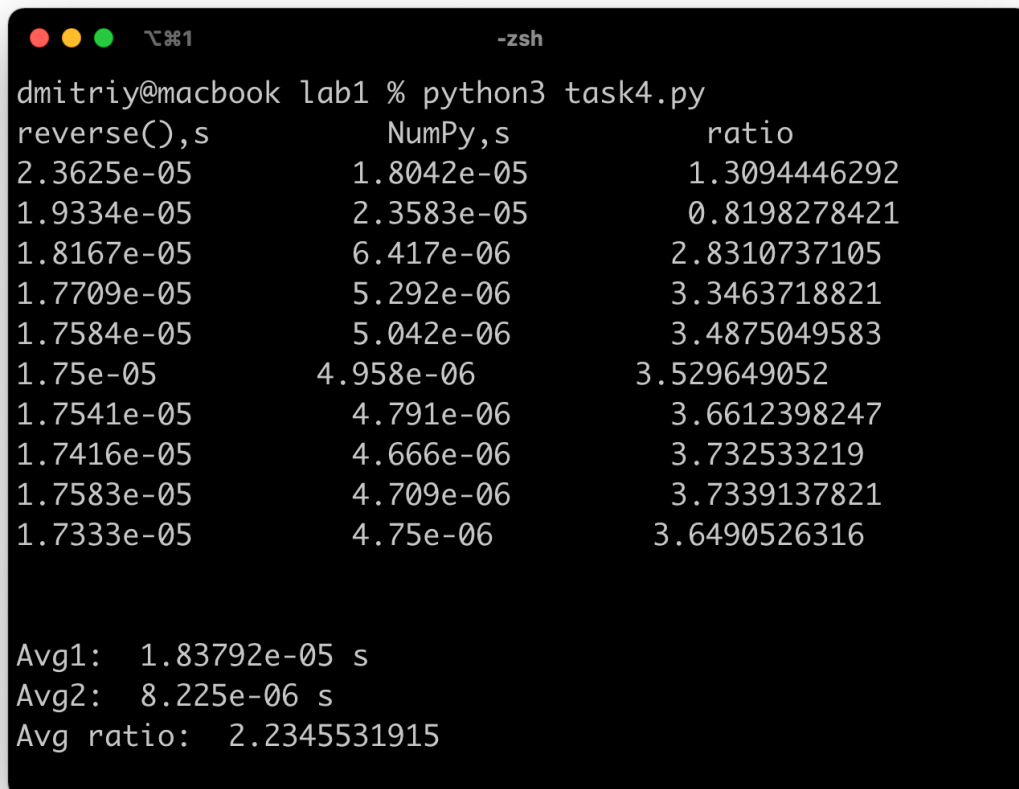
```
def det(m):
    d = m[0][0]*m[1][1]*m[2][2]+\
        m[2][0]*m[0][1]*m[1][2]+\
        m[1][0]*m[2][1]*m[0][2]-\
        m[2][0]*m[1][1]*m[0][2]-\
        m[0][0]*m[2][1]*m[1][2]-\
        m[1][0]*m[0][1]*m[2][2]
    return d

def reverse(matrix):
    d = det(matrix)
    if d == 0:
        print('Обратная матрица невозможна, определитель равен нулю!')
        return
    new_matrix = [[0,0,0],[0,0,0],[0,0,0]]
    k = 1
    for n in range(len(matrix)):
        for m in range(len(matrix[n])):
            a = []
            for q in range(len(matrix)):
                for p in range(len(matrix[q])):
                    if m != p and n != q:
                        a.append(matrix[q][p])
            new_matrix[m][n] = (a[0] * a[3] - a[1] * a[2]) * k
            k = k*(-1)
    for n in range(len(new_matrix)):
        for m in range(len(new_matrix[n])):
            new_matrix[n][m] = round(new_matrix[n][m] * (1/d), 15)
    return new_matrix
```

Фрагмент кода 5

3.2 Сравнение быстродействия

Сравним быстродействия написанной функции и функции, представленной в библиотеке NumPy. Для сравнения был использован модуль timeit. Было произведено 10 замеров быстродействия. Результаты продемонстрированы на изображении ниже (Рисунок 3)



```
dmitriy@macbook lab1 % python3 task4.py
reverse(),s      NumPy,s      ratio
2.3625e-05      1.8042e-05      1.3094446292
1.9334e-05      2.3583e-05      0.8198278421
1.8167e-05      6.417e-06       2.8310737105
1.7709e-05      5.292e-06       3.3463718821
1.7584e-05      5.042e-06       3.4875049583
1.75e-05        4.958e-06       3.529649052
1.7541e-05      4.791e-06       3.6612398247
1.7416e-05      4.666e-06       3.732533219
1.7583e-05      4.709e-06       3.7339137821
1.7333e-05      4.75e-06        3.6490526316

Avg1:  1.83792e-05 s
Avg2:  8.225e-06 s
Avg ratio:  2.2345531915
```

Рисунок 3

Мы видим, что в среднем функция из библиотеки NumPy справляется с поставленной задачей в 2,23 раза быстрее, чем написанная мною функция.

ВЫВОД

Все поставленные задачи были выполнены. В процессе выполнения задач я познакомился с языком программирования Python и библиотекой NumPy, а также закрепил некоторые знания по линейной алгебре. Сравнив быстродействие функций, можно сделать вывод – использование функций из библиотеки NumPy более эффективно, а так же это экономит время разработчика.

СПИСОК ЛИТЕРАТУРЫ

- 1 Transpose – Wikipedia // URL: <https://en.wikipedia.org/wiki/Transpose>
- 2 Matrix Multiplication – Wikipedia // URL: https://en.wikipedia.org/wiki/Matrix_multiplication
3. Rank (Linear algebra) – Wikipedia // URL: [https://en.wikipedia.org/wiki/Rank_\(linear_algebra\)](https://en.wikipedia.org/wiki/Rank_(linear_algebra))
4. Invertible matrix – Wikipedia // URL: https://en.wikipedia.org/wiki/Invertible_matrix
5. NumPy – Wikipedia // URL: <https://en.wikipedia.org/wiki/NumPy>

ПРИЛОЖЕНИЕ

Репозиторий с кодом лабораторной работы на GitHub:

<https://github.com/Evisom/itmo-ads-lab1>