Санкт-Петербургский Национальный Исследовательский Университет
Информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №1
Вариант №1

Выполнили:
Смирнов И.И.
Касьяненко В.М.
Проверил:
Мусаев А.А.

Санкт-Петербург
2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Для становления хорошим специалистом в области программирования на языке Python необходимо знать основной функционал языка, а также библиотеки, которые помогут при решения поставленных задач.

Цель данной работы – ознакомление с языком программирования Python.

В ходе выполнения лабораторной работы были решены следующие задачи:

- создание программ на языке Python с функциями транспонирования, умножения и определения ранга матриц, а также нахождения обратных матриц
- ознакомление с библиотеками Numpy и Timeit
- анализ библиотеки Numpy
- сравнение быстродействия функций с использованием библиотеки Numpy и без нее

1 ЗАДАНИЕ 1

1.1 Ввод матриц

Для начала выполнения задания необходимо создать матрицы и ввести в них значения. Чтобы это сделать пользователю предлагается внести данные о количестве строк и столбцов в матрице. После чего будет использован поэлементный ввод в матрицу (рисунок 1).

```
row1 = int(input('Введите количество строк в матрице 1: '))
col1 = int(input('Введите количество столбцов в матрице 1: '))

matrix = [[0 for j in range(col1)] for i in range(row1)]
for i in range(row1):
    for j in range(col1):
        matrix[i][j] = int(input('Введите число в '+str(i+1)+' строке '+str(j+1)+' столбце: '))
```

Рисунок 1 – Ввод матрицы

1.2 Транспонирование матрицы

Транспонированная матрица — матрица, полученная путем замены строк на столбцы и наоборот исходной матрицы. Для реализации задачи считываются все элементы исходной матрицы при помощи вложенного цикла с последующей их перестановкой и выводом (рисунок 2).

Для более эстетичного вывода полученной матрицы используется метод ljust(), который возвращает выровненную по левому краю строку заданной минимальной ширины (в данном случае 2 символа).

```
print('Транспонированная матрица:')
for i in range(col1):
    for j in range(row1):
        print(str(matrix[j][i]).ljust(2), end=' ')
    print()
```

Рисунок 2 – Транспонирование матрицы

1.3 Умножение матриц

Для выполнения следующей операции потребуется ввести проверку возможности умножения введенных матриц. Чтобы это осуществить сравнивается число колонок первой матрицы с числом строк во второй. Если они не равны, то выводится ошибка, в другом случае выполняется умножение. Для умножения используется сложение произведений элементов строк первой матрицы на элементы столбцов второй матрицы с последующей записью значений в новую матрицу, а также производится вывод этой матрицы пользователю (рисунок 3).

```
cmatrix = [[0 for j in range(col2)] for i in range(row1)]
if col1 == row2:
    for i in range(row1):
        for j in range(col2):
            cmatrix[i][j] = sum(matrix[i][kk] * bmatrix[kk][j] for kk in range(row2))

    print('Умножение матриц:')
    for i in range(row1):
        for j in range(col2):
            print(str(cmatrix[i][j]).ljust(2), end=' ')
        print()
else:
    print('Ошибка! Такие матрицы нельзя перемножить')
```

Рисунок 3 – Умножение матриц

1.4 Определение ранга матрицы

Рангом матрицы называется максимальное число линейно независимых строк (ни одна из них не выражается линейно через другие).

В начале необходимо удалить нулевые строки в исходной матрице, а также найти первый ненулевой элемент в строке. Для нахождения такого элемента программа проходится по строке и ищет первый элемент не равный 0, на который впоследствии делит всю строку, чтобы привести этот элемент к 1. Далее вычисляется коэффициент для дальнейших операций над ним путем деления первого ненулевого элемента следующей строки на элемент, стоящий над ним по столбцу. После чего из элементов следующей строки вычитается произведение данного коэффициента и соответствующего элемента строки выше (рисунок 4). Таким образом, в следующей строке первый ненулевой элемент становится нулевым.

```
matrix_copy = matrix.copy()
for r_copy in matrix_copy:
    if r_copy == [0] * len(r_copy):
        matrix.remove(r_copy)

for r_num in range(len(matrix)):
    for i_del in range(len(matrix[r_num])):
        if matrix[r_num][i_del] != 0:
            delitel = matrix[r_num][i_del]
            break
    matrix[r_num] = [matrix[r_num][i] / delitel for i in range(len(matrix[r_num]))]

    for r_next in range(r_num + 1, len(matrix)):
        k = matrix[r_next][i_del] / matrix[r_num][i_del]
        matrix[r_next] = [matrix[r_next][i] - k * matrix[r_num][i] for i in range(len(matrix[r_next]))]
```

Рисунок 4 – Преобразование исходной матрицы

После преобразований количество строк берется за ранг матрицы, а также проводится проверка на наличие нулевых строк. В случае их обнаружения ранг матрицы уменьшается на 1 за каждую такую строку, после чего производится окончательный вывод значения ранга (рисунок 5).

```
rank_matrix = len(matrix)

for r_fin in matrix:
    null = True
    for i in r_fin:
        if i != 0:
            null = False
            break
    if null:
        rank_matrix -= 1
print()
print('Ранг матрицы:')
print(rank_matrix)
```

Рисунок 5 – Корректировка значения ранга матрицы

2 ЗАДАНИЕ 2

2.1 Подготовка к дальнейшим операциям с библиотекой NumPy

При выполнении данного задания будет использован аналогичный ввод из первого задания. Также, необходимо отметить, что для работы с библиотекой NumPy необходимо произвести подключение с помощью ввода специальной строки: «import numpy».

2.2 Транспонирование матрицы с использованием NumPy

Для выполнения операции транспонирования код с использованием библиотеки NumPy можно записать в одну строку: «numpy.transpose(matrix)» (рисунок 6). Далее можно произвести вывод готовой матрицы способом, рассмотренным в первом разделе.

```
print()
arr_matrix = numpy.transpose(matrix)
print('Транспонированная матрица:')
for i in range(col1):
    for j in range(row1):
        print(str(arr_matrix[i][j]).ljust(2), end=' ')
    print()
```

Рисунок 6 – Транспонирование матрицы с использованием NumPy

2.3 Умножение матриц с использованием NumPy

Как в подразделе 1.3, так и здесь будет использована проверка на возможность умножения введенных матриц. Далее, используя библиотеку NumPy, появляется возможность произвести умножение матриц в одну строку: «numpy.matmul(matrix, bmatrix)» (рисунок 7).

```
if col1 == row2:
    cmatrix = numpy.matmul(matrix, bmatrix)
    print('Результат умножения матриц:')
    for i in range(row1):
        for j in range(col2):
            print(str(cmatrix[i][j]).ljust(2), end=' ')
        print()
else:
    print('Ошибка! Такие матрицы нельзя перемножить')
```

Рисунок 7 – умножение матриц с использованием NumPy

2.4 Определение ранга матрицы с использованием NumPy

С помощью библиотеки NumPy можно посчитать ранг матрицы в одну строку: «numpy.linalg.matrix_rank(matrix)» (рисунок 8).

```
rank = numpy.linalg.matrix_rank(matrix)
print('Ранг матрицы:')
print(rank)
```

Рисунок 8 – Определение ранга матрицы с использованием NumPy

2.5 Преимущества и недостатки библиотеки NumPy

Явным преимуществом библиотеки NumPy является краткая запись кода, что значительно уменьшает время написания программы. Также, необходимо отметить, что в библиотеке существует большое методов, которые позволяют выполнять различные операции с большим количеством данных.

Из недостатков можно заметить, что массивы NumPy имеют фиксированный размер и в отличие от списков Python не могут динамически расти.

3 ЗАДАНИЕ 3

3.1 Создание функции нахождения обратной матрицы

Для данного задания используется другой ввод, так как задана фиксированная матрица (3x3). Пользователь вводит значения в матрицу построчно (рисунок 9).

```
print("Введите строки матрицы 3x3 (через Enter), указывая все элементы через пробел: ")

matrix = []
for i in range(3):
    val = input()
    matrix.append(list(map(int, val.split())))
```

Рисунок 9 – Ввод матрицы 3x3

Рассмотрим функции, необходимые для выполнения данного алгоритма. Первая из них является определением детерминанта введенной матрицы, который высчитывается методом треугольника. Вторая же отвечает за нахождение минора. Также в данном фрагменте (рисунок 10) можно заметить переменную b, которая впоследствии будет отвечать за неправильный ввод или детерминант равный 0.

```
def inverse(matrix):
    b = "Неправильный ввод или детерминант равен 0"

    def determ(mx):
        diff_diag = mx[0][0] * mx[1][1] * mx[2][2] + mx[0][1] * mx[1][2] * mx[2][0] + mx[1][0] * mx[2][1] * mx[0][2] - \
                    mx[2][0] * mx[1][1] * mx[0][2] - mx[1][0] * mx[0][1] * mx[2][2] - mx[2][1] * mx[1][2] * mx[0][0]
        return diff_diag

    def minor(mi):
        return mi[0][0] * mi[1][1] - mi[0][1] * mi[1][0]
```

Рисунок 10 – Функции для выполнения алгоритма

Для проверки будет сравниваться введенное количество строк и столбцов с заданным (3x3), а также проверяется равен ли детерминант 0. В случае, если программа не прошла проверку, то функция возвращает значение b, иначе обратная матрица существует и будет найдена в дальнейшем.

Создается матрица, которая будет заполняться найденными значениями. Для этого необходимо пройтись по каждому элементу и убрать строку, содержащую этот элемент, а также столбец, после чего определить коэффициент четности, вычислить значение с помощью минора и записать его в подготовленную матрицу. Необходимо отметить, что для корректного вывода, необходимо транспонировать матрицу (рисунок 11).

```
rows_len = len(matrix[0])
columns_len = len(matrix)


mx_det = determ(matrix)


if rows_len == 3 and columns_len == 3 and mx_det != 0:

    result_matrix = [
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]
    ]


    for r_num in range(3):
        for i_row in range(3):
            matrix_copy = [row.copy() for row in matrix]
            matrix_copy.pop(r_num)
            for i_del in range(2):
                matrix_copy[i_del].pop(i_row)


            if (r_num + i_row) % 2 == 0:
                k = 1
            else:
                k = -1

            result_matrix[r_num][i_row] = minor(matrix_copy) * k / mx_det
            matrix_t = [[0 for j in range(3)] for i in range(3)]
            for i in range(3):
                for j in range(3):
                    matrix_t[i][j] = result_matrix[j][i]
    return matrix_t
else:
    return b
```

Рисунок 11 – Нахождение обратной матрицы

3.2 Сравнение быстродействия с помощью библиотеки Timeit

Для сравнения быстродействия собственной функции и ее аналога из библиотеки NumPy будет использована библиотека Timeit. С помощью тестов были определены некоторые результаты, записанные в таблицу 1.

Собственная функция	Библиотека NumPy
0.3126900000497699	0.2275467999279499
0.2742557000601664	0.26804810005705804
0.3037335999542847	0.2427402000175789
0.3363783999811858	0.23354489996563643
0.30570119991898537	0.28481739992275834

Посчитав среднее значение, можно увидеть, что собственная функция работает медленнее в 1,2 раза.

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был получен опыт использования библиотек NumPy и Timeit, а также опыт написания алгоритмов для транспонирования, умножения, определения ранга матриц и нахождения обратных матриц. Также был проведен анализ библиотеки NumPy и изучено быстродействие кода при помощи библиотеки Timeit.

СПИСОК ЛИТЕРАТУРЫ

1. Википедия. Ранг матрицы. [Электронный ресурс] – https://ru.wikipedia.org/wiki/Ранг_матрицы (Дата последнего обращения 05.10.2022);

2. Студворк. Обратная матрица. [Электронный ресурс] – https://studwork.org/spravochnik/matematika/matricy/obratnaya-matrica (Дата последнего обращения 05.10.2022);

3. OnlineMSchool. Минор и алгебраическое дополнение матрицы. [Электронный ресурс] – https://ru.onlinemschool.com/math/library/matrix/minors/ (Дата последнего обращения 05.10.2022).

ПРИЛОЖЕНИЕ

Ссылка на полный код данной лабораторной работы:
https://github.com/VeraKasianenko/Algorithms_1_term_ICT/tree/main/labs/lab_1