Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий и систем связи

**Лабораторная работа № 1**

Выполнили

Кремпольская Е.А.

Петрова Н.Г.

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

**Введение**

В этой работе мы напишем несколько функций, позволяющих производить действия над матрицами, изучим и проанализирую рациональность использования библиотеки NumPy в Python для реализации этих же функций, а также с помощью библиотеки timeit сравнили быстродействие нашей функции с ее аналогом из библиотеки NumPy.

**Задание 1**

Создадим программу на языке Python, которая содержит следующие функции: умножение матриц, транспонирование матрицы, определение ранга матрицы.

Начнём с ввода произвольной матрицы. Считываем количество её строк и столбцов, далее создаём 2д-список (далее – матрица) и считываем в неё элементы построчно, используя цикл for (Рисунок 1).

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 1 – Ввод матриц

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Результат ввода матриц

**Транспонирование**

Создадим функцию transp (Рисунок 3), которая будет принимать на вход нашу матрицу, а возвращать транспонированную матрицу.

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 3 – Функция transp

В качестве аргументов наша функция принимает матрицу и её размеры. Внутри самой функции создаём новую матрицу, у которой высота равна ширине нашей введённой матрицы, а ширина соответственно равна высоте нашей матрицы. Далее с помощью вложенного цикла проходимся по нашему списку matrix\_1 и присваиваем в список transp\_matrix элементы с противоположными индексами (столбец становится строчкой, строчка – столбцом). Таким образом у нас получается транспонированная матрица, которую мы благополучно возвращаем из функции.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Результат фунции transp

**Умножение**

Создадим функцию умножения матриц (Рисунок 5), принимающую 2 матрицы в качестве аргументов.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Функция multi

Внутри функции создаём матрицу, состоящую из нулей, шириной второй матрицы, высотой первой матрицы. Далее проходимся по строкам первой матрицы, по столбцам второй и по элементам в них (элементов одинаковое количество, т. к. высота первой матрицы должна быть равна ширине второй, иначе их нельзя было бы умножить). К элементу multi\_matrix, имеющему координаты (строка первой матрицы; столбец второй матрицы) прибавляем соответственные произведения элементов этих строки и столбца (Рисунок 5).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Результат функции multi

**Ранг матрицы**

Создадим функцию ranks (Рисунок 9), которая будет принимать на вход нашу матрицу и выводить её ранг.

Для нахождения ранга матрицы воспользуемся функцией определителя.

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 7 – Функция ranks

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Результат функции ranks

**Задание 2**

Реализовали программу, созданную нами в задании 1, при помощи библиотеки NumPy. Импортируем библиотеку NumPy. Ввод данных аналогичен первому заданию.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Функции: транспортирование, умножение и определение ранга через NumPy

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Результат функций: транспортирование, умножение и определение ранга через NumPy

Очевидным преимуществом библиотеки NumPy по сравнению с созданием функций вручную является упрощение кода в несколько десятков раз с сохранением скорости. Но чтобы уверенно работать с библиотекой NumPy нужно отдельно изучать команды и их взаимодействие.

**Задание 3**

Создадим функцию для возведения матрицы A размерности 3x3 в степень -1 и сравним быстродействие нашей функции с её аналогом из NumPy.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – Обратная матрица 3\*3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Результат функции obr

**Сравнение времени нашего кода с NumPy**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 – Код для сравнения времени выполнения нашего кода и с помощью NumPy

**Сравнение быстродействия**

Импортируем библиотеку timeit для измерения быстродействия кода. Сначала замерим скорость нашего кода, а затем NumPy (Рисунок 9).

Результат кода с использованием NumPy примерно в 1,7 раза медленнее написанного кода без библиотеки:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14 – Время выполнения нашего кода и с помощью NumPy

**Вывод**

В результате выполненной работы были получены знания о действиях над матрицами, библиотеке NumPy, функции timeit. Были проведены эксперименты, позволяющие сравнить работоспособность библиотеки NumPy и сделать оценку удобства и нужности применения данной библиотеки.