МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Кафедра инфокоммуникаций

Отчет

по лабораторной работе №9

«Основы работы с библиотекой NumPy»

по дисциплине:

«Введение в системы искусственного интеллекта»

Вариант 8

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Михайличенко Руслан Михайлович
_____(подпись)
Проверил:
Воронкин Роман Александрович
_____(подпись)

Ставрополь, 2022 г.

Цель работы: исследовать базовые возможности библиотеки NumPy языка программирования Python.

Задание

8. Соседями элемента A_j в матрице назовем элементы A_k с $i-1<k<i+1$, $j-1<1<j+1$, $(k,1)/(i,j)$. Операция сглаживания матрицы дает новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы размером 7 на 7. В сглаженной матрице найти сумму модулей элементов, расположенных ниже главной диагонали.

Ход работы

```
from math import *
from random import *
import numpy as np


def MakeMatr(n, a, b):
    Matr = (b-a)*np.random.random(size=(n,n)) + a
    return Matr


def CorrectMatr(Matr):
    (nRow, nCol) = Matr.shape
    for Row in range(nRow+1):
        for Col in range(nCol+1):
            if Row == 0:
                if Col == 0:
                    Matr[Row][Col] = (Matr[Row][Col+1]+Matr[Row+1][Col])/2
            if Row == 0:
                if Col > 0 and Col < (nCol-1):
                    Matr[Row][Col] = (Matr[Row][Col+1]+Matr[Row][Col-1]+Matr[Row+1][Col])/3
            if Row == 0:
                if Col == (nCol-1):
                    Matr[Row][Col] = (Matr[Row][Col-1]+Matr[Row+1][Col])/2
            if Row > 0 and Row < (nRow-1):
                if Col == 0:
                    Matr[Row][Col] = (Matr[Row][Col+1]+Matr[Row-1][Col]+Matr[Row+1][Col])/3
            if Row > 0 and Row < (nRow-1):
                if Col > 0 and Col < (nCol-1):
                    Matr[Row][Col] = (Matr[Row][Col+1]+Matr[Row][Col-1]+Matr[Row+1][Col]+Matr[Row-1][Col])/4
            if Row > 0 and Row < (nRow-1):
                if Col == (nCol-1):
                    Matr[Row][Col] = (Matr[Row][Col-1]+Matr[Row-1][Col]+Matr[Row+1][Col])/3
            if Row == (nRow-1):
                if Col == 0:
                    Matr[Row][Col] = (Matr[Row][Col+1]+Matr[Row-1][Col])/2
            if Row == (nRow-1):
                if Col > 0 and Col < (nCol-1):
                    Matr[Row][Col] = (Matr[Row][Col+1]+Matr[Row][Col-1]+Matr[Row-1][Col])/3
            if Row == (nRow-1):
                if Col == (nCol-1):
                    Matr[Row][Col] = (Matr[Row][Col-1]+Matr[Row-1][Col])/2

    return Matr


def PrintMatr(Matr):
    (nRow, nCol) = Matr.shape
    for Row in range(nRow):
        for Col in range(nCol):
            print("{0: 7.3f}".format(Matr[Row][Col]), end=" ")
        print()
    print()
    totalSum = sum([Matr[Row][Col] for i in range(1,len(Matr)) for j in range(Row)])
    print("Сумма:  ", totalSum)


n = int(input("Введите размер матрицы (NxN): "))
MyMatr=MakeMatr(n, -7, 7)
NMatr = CorrectMatr(MyMatr)
PrintMatr(NMatr)
```

Рисунок 1 – Листинг программы

```
Введите размер матрицы (NxN): 7
 -0.026   0.745   1.718   0.057  -0.574   2.741   2.011
 -1.140   2.946   1.504  -1.475   1.782   0.052   1.082
  2.064   2.851  -0.958   0.587  -1.955  -0.966  -1.420
  0.430  -1.666  -0.640  -1.648  -1.845  -0.507  -0.556
 -1.337  -1.553  -0.166   0.384   1.346   0.231  -0.430
 -0.713   2.021   0.120   1.347   1.238   1.718  -0.162
  3.079  -0.087   1.077   1.619   3.075   1.006   0.422

Сумма:  15.19165032499176
```

Рисунок 2 – Результат работы

Вывод: исследовал базовые возможности библиотеки NumPy языка программирования Python.

Ответы на вопросы

1. Каково назначение библиотеки NumPy?

NumPy — это библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй. Полное название библиотеки — Numerical Python extensions, или «Числовые расширения Python».

Назначение:

Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами. В результате любой алгоритм, который может быть выражен в виде последовательности операций над массивами (матрицами) и реализованный с использованием NumPy, работает так же быстро, как эквивалентный код, выполняемый в MATLAB.

Где используется NumPy:

Научные вычисления. NumPy пользуются ученые для решения многомерных задач в математике и физике, биоинформатике, вычислительной химии и даже когнитивной психологии.

Создание новых массивных библиотек. На основе NumPy появляются новые типы массивов, возможности которых выходят за рамки того, что предлагает библиотека. Например, библиотеки Dask, CuPy или XND.

Data Science. В основе экосистемы для анализа данных лежит NumPy. Библиотека используется на всех этапах работы с данными: извлечение и преобразование, анализ, моделирование и оценка, репрезентация.

Machine Learning. Библиотеки для машинного обучения scikit-learn и SciPy тоже работают благодаря вычислительным мощностям NumPy.

Визуализация данных. По сравнению непосредственно с Python возможности NumPy позволяют исследователям визуализировать наборы данных, которые гораздо больше по размеру. Например, библиотека лежит в основе системы PyViz, которая включает в себя десятки программ для визуализации.

2. Что такое массивы ndarray?

Основной элемент библиотеки NumPy — объект ndarray (что значит N-размерный массив). Этот объект является многомерным однородным массивом с заранее заданным количеством элементов. Однородный — потому что практически все объекты в нем одного размера или типа. На самом деле, тип данных определен другим объектом NumPy, который называется dtype (тип-данных). Каждый ndarray ассоциирован только с одним типом dtype.

Количество размерностей и объектов массива определяются его размерностью (shape), кортежем N-положительных целых чисел. Они указывают размер каждой размерности. Размерности определяются как оси, а количество осей — как ранг.

Еще одна странность массивов NumPy в том, что их размер фиксирован, а это значит, что после создания объекта его уже нельзя поменять. Это поведение отличается от такового у списков Python, которые могут увеличиваться и уменьшаться в размерах.

Простейший способ определить новый объект ndarray — использовать функцию array(), передав в качестве аргумента Python-список элементов.

3. Как осуществляется доступ к частям многомерного массива?

Доступ к элементам массива осуществляется по целочисленным индексами, начинается отсчет с 0:

Если представить многомерный массив как систему вложенных одномерных массивов (линейный массив, элементы которого могут быть

линейными массивами), становится очевидной возможность получать доступ к подмассивам с использованием неполного набора индексов

При использовании неполного набора индексов, недостающие индексы неявно заменяются списком всех возможных индексов вдоль соответствующей оси. Сделать это явным образом можно, поставив ":"

Индексы могут принимать отрицательные целые значения. В этом случае отсчет ведется от конца массива

4. Как осуществляется расчет статистик по данным?

Расчет статистик по строкам или столбцам массива

Для начала создадим матрицу, которая нам понадобится в работе.

```
>>> m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
>>> print(m)
[[1 2 3 4]
[5 6 7 8]
[9 1 5 7]]
```

В этом случае будет создан объект типа matrix.

```
>>> type(m)
<class 'numpy.matrixlib.defmatrix.matrix'>
```

Matix можно превратить в ndarray вот так:

```
>>> m = np.array(m)
>>> type(m)
<class 'numpy.ndarray'>
```

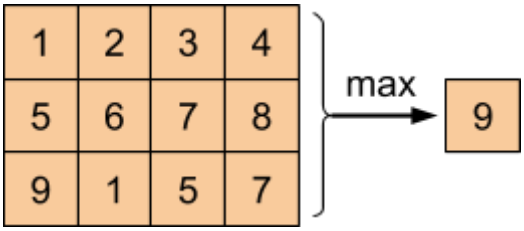
В любом случае наша таблица чисел будет выглядеть следующим образом.

1	2	3	4
5	6	7	8
9	1	5	7

Вызовем функцию вычисления статистики (максимальный элемент) без аргументов.

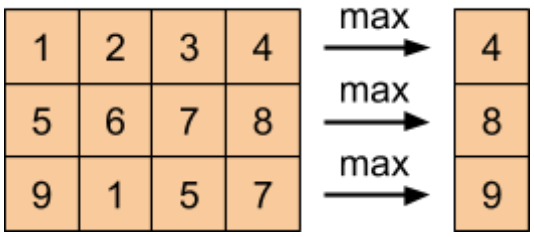
```
>>> m.max()
9
```

В этом случает будут обработаны все элементы массива.



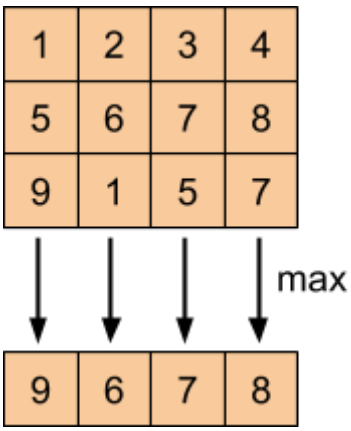
Если необходимо найти максимальный элемент в каждой строке, то для этого нужно передать в качестве аргумента параметр *axis=1*.

```
>>> m.max(axis=1)
matrix([[4],
[8],
[9]])
```



Для вычисления статистики по столбцам, передайте в качестве параметра аргумент *axis=0*.

```
>>> m.max(axis=0)
matrix([[9, 6, 7, 8]])
```



Функции (методы) для расчета статистик в *Numpy*

Ниже, в таблице, приведены методы объекта *ndarray* (или *matrix*), которые, как мы помним из раздела выше, могут быть также вызваны как функции библиотеки *Numpy*, для расчета статистик по данным массива.

Имя метода	Описание
argmax	Индексы элементов с максимальным значением (по осям)
argmin	Индексы элементов с минимальным значением (по осям)
max	Максимальные значения элементов (по осям)
min	Минимальные значения элементов (по осям)
mean	Средние значения элементов (по осям)
prod	Произведение всех элементов (по осям)
std	Стандартное отклонение (по осям)
sum	Сумма всех элементов (по осям)
var	Дисперсия (по осям)

5. Как выполняется выборка данных из массивов ndarray?

Создадим матрицу, с которой будем работать.

```
>>> m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
>>> print(m)
[[1 2 3 4]
[5 6 7 8]
[9 1 5 7]]
```

Получим вот такую таблицу чисел.

1	2	3	4
5	6	7	8
9	1	5	7

Извлечем элемент из нашей матрицы с координатами (1, 0), 1 – это номер строки, 0 – номер столбца.

Здесь и далее, элементы, с которыми мы работаем в матрице будут окрашены в оранжевый цвет на соответствующей картинке.

1	2	3	4
5	6	7	8
9	1	5	7

```
>>> m[1, 0]
5
```

В приведенной записи, в квадратных скобках указывается номер строки – первой цифрой и номер столбца – второй.

Строка матрицы

Получим вторую строчку матрицы.

1	2	3	4
5	6	7	8
9	1	5	7

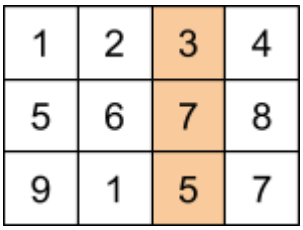
```
>>> m[1, :]
```

matrix([[5, 6, 7, 8]])

Двоеточие означает “все элементы”, в приведенном примере, первый элемент – это номер строки, второй – указание на то, что необходимо взять элементы всех столбцов матрицы.

Столбец матрицы

Извлечем третий столбец матрицы.



>>> m[:, 2]

matrix([[3],

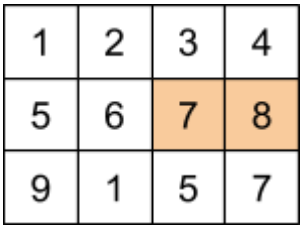
[7],

[5]])

Здесь, также как в предыдущем примере, используется двоеточие. Первый элемент в квадратных скобках означает, что мы возьмем по элементу из каждой строки, которые находится в столбце, указанном во втором элементе.

Часть строки матрицы

Иногда возникает задача взять не все элементы строки, а только часть: рассмотрим пример, когда нам из второй строки нужно извлечь все элементы, начиная с третьего.



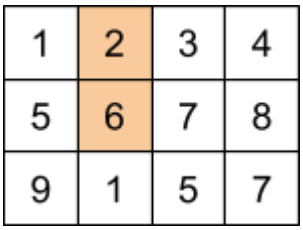
>>> m[1, 2:]

matrix([[7, 8]])

Запись “**2:”** означает, что начиная с третьего столбца включительно (т.к. нумерация начинается с 0, то третий элемент имеет индекс 2) взять все оставшиеся в ряду элементы .

Часть столбца матрицы

Аналогично предыдущему примеру, можно извлечь только часть столбца матрицы.

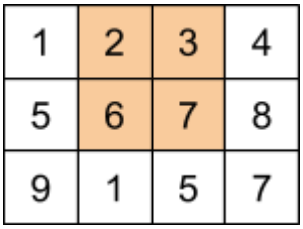



```
>>> m[0:2, 1]
matrix([[2],
[6]])
```

В приведенной записи **“0:2”** означает: взять все элементы столбца начиная с индекса 0, заканчивая индексом 2, но последний элемент в результат не включать.

Непрерывная часть матрицы

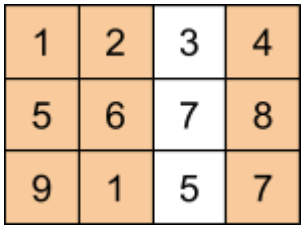
Извлечем из заданной матрицы матрицу, располагающуюся так как показано на рисунке ниже.




```
>>> m[0:2, 1:3]
matrix([[2, 3],
[6, 7]])
```

Произвольные столбцы / строки матрицы

Numpy позволяет извлекать произвольный набор столбцов или строк матрицы, строки (столбцы) которые нужно извлечь передаются в виде списка.



```
>>> cols = [0, 1, 3]
>>> m[:, cols]
matrix([[1, 2, 4],
[5, 6, 8],
[9, 1, 7]])
```