МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»

Отчет по лабораторной работе №2
«Основы работы с библиотекой NumPy»

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-20-1
Бокань И.П. « » _____ 2022г.
Подпись студента _____
Работа защищена « »_____2022г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2022

1. Вывод

Расчет статистик по данным в массиве

m =

21	13	12	4
16	12	8	2
6	17	0	7
5	25	3	41

Создание объект типа matrix:

```
m = np.matrix('21 13 12 4; 16 12 8 2; 6 17 0 7; 5 25 3 41')
print(m)
```

```
[[21 13 12  4]
 [16 12  8  2]
 [ 6 17  0  7]
 [ 5 25  3 41]]
```

Рисунок 1.1 - Результаты примера создание матрица

Определение типа объекта:

```
print(type(m))
```

```
<class 'numpy.matrix'>
```

```
m_arr = np.array(m)
print(type(m_arr))
```

```
<class 'numpy.ndarray'>
```

Определения размерности массива:

```
print(m.shape)
```

```
(4, 4)
```

```
print(m_arr.shape)
```

```
(4, 4)
```

Рисунок 1.2 - Результаты примера определение объекта

Вызов функции расчета статистики.

```
print(m.max()) # тоже самле что np.max(m)
```

```
41
```

```
print(m.max(axis=0)) # строкам
```

```
[[21 25 12 41]]
```

```
print(m.max(axis=1)) # столбцам
```

```
[[21]
 [16]
 [17]
 [41]]
```

```
m.sum() # сумма
```

```
192
```

Рисунок 1.3 - Результаты примера расчета статики

Использование boolean массива для доступа к ndarray.

```
nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])

print("nums(", type(nums), "): ", nums)
print("nums(", type(letters), "): ", letters)
```

```
nums( <class 'numpy.ndarray'> ):  [ 1  2  3  4  5  6  7  8  9 10]
nums( <class 'numpy.ndarray'> ):  ['a' 'b' 'c' 'd' 'a' 'e' 'b']
```

```
less_then_5 = nums < 5
print(less_then_5)
```

```
[ True  True  True  True False False False False False False]
```

```
less_then_a = letters == 'a'
print(less_then_a)
```

```
[ True False False False  True False False]
```

```
print(nums[less_then_5]) # може самое что nums[nums < 5]
```

```
[1 2 3 4]
```

```
print(letters[less_then_a]) # може самое что letters[letters == 'a']
```

```
['a' 'a']
```

```
mod_m = np.logical_and(m>=3, m<=7)
print(mod_m)
```

```
[[False False False  True]
 [False False False False]
 [ True False False  True]
 [ True False  True False]]
```

```
print(m[mod_m])
```

```
[[4 6 7 5 3]]
```

Рисунок 1.4 - Результаты примера использование boolean массива для доступа к ndarray

Элемент матрицы с заданными координатами.

m	0	1	2	3
0	21	13	12	4
1	16	12	8	2
2	6	17	0	7
3	5	25	3	41

```
print(m[2, 3])
```

```
7
```

Рисунок 1.5 - Результаты примера элемент матрицы с заданными координатами

Строка и столбец матрицы.

Строка

m	0	1	2	3
0	21	13	12	4
1	16	12	8	2
2	6	17	0	7
3	5	25	3	41

Столбец

m	0	1	2	3
0	21	13	12	4
1	16	12	8	2
2	6	17	0	7
3	5	25	3	41

```
: print("Строка:")
  print(m[2, :])
```

```
Строка:
[[ 6 17  0  7]]
```

```
: print("\nСтолбец:")
  print(m[:, 3])
```

```
Столбец:
[[ 4]
 [ 2]
 [ 7]
 [41]]
```

Рисунок 1.6 - Результаты примера элемент матрицы с заданными строки и столбец координатами

Часть строки, столбца и непрерывная матрицы.

Часть строка

m	0	1	2	3
0	21	13	12	4
1	16	12	8	2
2	6	17	0	7
3	5	25	3	41

Часть столбца

m	0	1	2	3
0	21	13	12	4
1	16	12	8	2
2	6	17	0	7
3	5	25	3	41

Непрерывная часть

m	0	1	2	3
0	21	13	12	4
1	16	12	8	2
2	6	17	0	7
3	5	25	3	41

```
print("Часть строка:")
print(m[2, 1:])
```

```
Часть строка:
[[17  0  7]]
```

```
print("Часть столбца:")
print(m[0:2, 3])
```

```
Часть столбца:
[[4]
 [2]]
```

```
print("Непрерывная часть:")
print(m[0:2, 1:3])
```

```
Непрерывная часть:
[[13 12]
 [12  8]]
```

Рисунок 1.7 - Результаты примера элемент матрицы с заданными по частям

Произвольные столбцы и строки матрицы.

m	0	1	2	3
0	21	13	12	4
1	16	12	8	2
2	6	17	0	7
3	5	25	3	41

```
rows = [0, 2]
cols = [0, 2]
print(m[rows, cols])
```

```
[[21  0]]
```

Рисунок 1.8 - Результаты примера элемент матрицы с заданными произвольные столбцы и строки матрицы

Arange() - аналог range() :

```
print(np.arange(10))
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
print(np.arange(1,9))
```

```
[1 2 3 4 5 6 7 8]
```

```
print(np.arange(0.1, 1.0, 0.2))
```

```
[0.1 0.3 0.5 0.7 0.9]
```

Рисунок 1.9 - Результаты примера элемент матрицы с дополнительным функцией “arange”

Matrix() :

```
# Python - список
a = [[1, 2], [3, 4]]
print(a)
```

```
[[1, 2], [3, 4]]
```

```
# Numpy - массив
b = np.array([[5, 6], [7, 8]])
print(b)
```

```
[[5 6]
 [7 8]]
```

```
# Matlab - стиль
c = np.matrix('[1, 2; 3, 4]')
print(c)
```

```
[[1 2]
 [3 4]]
```

Рисунок 1.10 - Результаты примера элемент матрицы с дополнительным функцией “matrix”

Np.[zeros | eye]() :

```
print(np.zeros((2, 4)))
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

```
np.eye(4)
```

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

Рисунок 1.11 - Результаты примера элемент матрицы с дополнительным функцией “np.zeros и np.eye”

Np.ravel() :

```
print(np.ravel(m))
```

```
[21 13 12  4 16 12  8  2  6 17  0  7  5 25  3 41]
```

```
print(np.ravel(m, order='C'))
```

```
[21 13 12  4 16 12  8  2  6 17  0  7  5 25  3 41]
```

```
print(np.ravel(m, order='F'))
```

```
[21 16  6  5 13 12 17 25 12  8  0  3  4  2  7 41]
```

Рисунок 1.12 - Результаты примера элемент матрицы с дополнительным функцией “np.ravel”

Np.meshgrid() :

```
import matplotlib.pyplot as plt

x = np.linspace(0, 1, 5)
y = np.linspace(0, 2, 5)
xg, yg = np.meshgrid(x, y)
plt.plot(xg, yg, color="r", marker="*", linestyle="none")
print("xg: \n", xg, "\n\nyg: \n", yg)
```

```
xg:
 [[0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]
 [0.   0.25 0.5  0.75 1.  ]]

yg:
 [[0.  0.  0.  0.  0. ]
 [0.5 0.5 0.5 0.5 0.5]
 [1.  1.  1.  1.  1. ]
 [1.5 1.5 1.5 1.5 1.5]
 [2.  2.  2.  2.  2. ]]
```

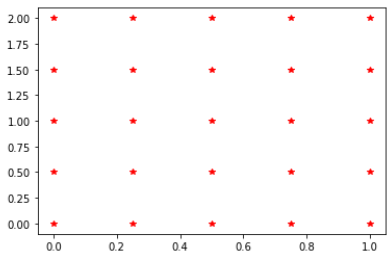

Рисунок 1.13 - Результаты примера элемент матрицы с дополнительным функцией “np.meshgrid”

Np.where() :

```
print(np.where(m % 2 == 0, m * 10, m / 10)) # np.where(условие, верно, не верно)
```

```
[[  2.1   1.3 120.   40. ]
 [160.  120.   80.   20. ]
 [ 60.    1.7   0.    0.7]
 [  0.5   2.5   0.3   4.1]]
```

Np.meshgrid() :

Рисунок 1.14 - Результаты примера элемент матрицы с дополнительным функцией “np.where”

```
print(np.random.permutation(m))
```

```
[[16 12  8  2]
 [21 13 12  4]
 [ 5 25  3 41]
 [ 6 17  0  7]]
```

Рисунок 1.15 - Результаты примера элемент матрицы с дополнительным функцией “np.random.permutation”

2. Индивидуальные задание

Индивидуальное задание

Дана целочисленная прямоугольная матрица.

Определить:

1) Количество столбцов, не содержащих ни одного нулевого элемента.
2) Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов.
3) Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

In [277]:
```
import numpy as np

def inv_1(mx: str):
    mx = np.matrix(mx)

    mx_element_count_null = np.sum(mx == 0)
    mx_row_sum = np.sum(mx, axis=1)
    mx_row_sort_sum = np.sort(mx_row_sum, axis=0)
    mx_sort_sum = mx[np.argsort(mx_row_sum, axis=0).ravel(), :]
    return [mx_element_count_null, mx, mx_row_sum, mx_sort_sum, mx_row_sort_sum]

def input_mx():
    print("Введите матрица:")
    x = inv_1(input("  Введите элементы массива (пример: \'1 2; 3 4\'): mx ="))
    print("\n Результат:\n")
    print("Элементы массива:\n", x[1])
    print("Элементы массива (сумма):\n", x[2])
    print("\n\nКоличество столбцов не содержащих ни одного 0 элемента:", x[0])
    print("\n\nУпорядоченная:")
    print("\nЭлементы массива:\n", x[3])
    print("\nЭлементы массива (сумма):\n", x[4])

input_mx()
```

```
Введите матрица:
  Введите элементы массива (пример: '1 2; 3 4'): mx =10 11 0 13 14; 5 6 0 8 9; 10 0 12 10 4; 10 21 12 43 14; 1 2 3 4 0

 Результат:

Элементы массива:
 [[10 11  0 13 14]
 [ 5  6  0  8  9]
 [10  0 12 10  4]
 [10 21 12 43 14]
 [ 1  2  3  4  0]]
Элементы массива (сумма):
 [[ 48]
 [ 28]
 [ 36]
 [100]
 [ 10]]


Количество столбцов не содержащих ни одного 0 элемента: 4


Упорядоченная:

Элементы массива:
 [[[ 1  2  3  4  0]
  [ 5  6  0  8  9]
  [10  0 12 10  4]
  [10 11  0 13 14]
  [10 21 12 43 14]]]

Элементы массива (сумма):
 [[ 10]
 [ 28]
 [ 36]
 [ 48]
 [100]]
```

Рисунок 2.1 - Результаты индивидуальные задание

3. Самостоятельно задание

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 7 * np.pi, 0.01)

# Синус
plt.subplot(3, 1, 1)
plt.plot(x, np.sin(x))
plt.title('Синус')

# Косинус
plt.subplot(3, 1, 3)
plt.plot(x, np.cos(x))
plt.title('Косинус')

plt.show()
```

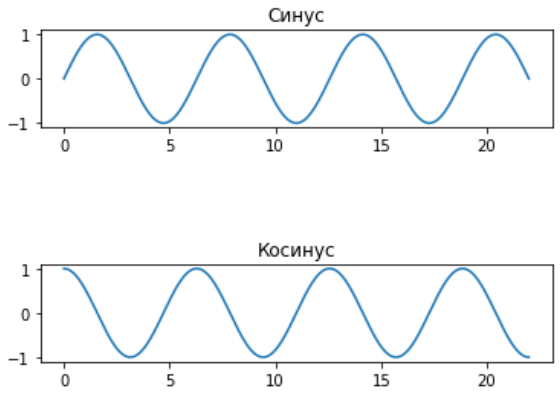

Рисунок 3.1 - Результаты математика “Синус и косинус” с помощью NumPy