МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

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

по дисциплине «Технологии распознавания образов»

Выполнила:
Первых Дарья
Александровна,2 курс,
группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ

```
import numpy as np
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]]
```

Рисунок 1 – Пример создания матрицы

```
print(m[1, 0])
```

```
5
```

Рисунок 2 – Пример элементы матрицы с заданными координатами

```
print(m[1, :])
```

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

Рисунок 3 – Пример вывода строки матрицы

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

```
[[3]
 [7]
 [5]]
```

Рисунок 4 – Пример вывода столбца матрицы

```
print(m[1, 2:])
```

```
[[7 8]]
```

Рисунок 5 – Пример вывода части строки матрицы

```
print(m[0:2, 1])
```

```
[[2]
 [6]]
```

Рисунок 6 – Пример вывода части столбца матрицы

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

```
[[2 3]
 [6 7]]
```

Рисунок 7 – Пример вывода непрерывной части матрцицы

```
cols = [0, 1, 3]
print(m[:, cols])
```

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

Рисунок 8 – Пример вывода произвольных столбцов / строк матрицы

```
print(m)
m.shape # Размерность массива
```

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

(3, 4)
```

Рисунок 9 – Пример вывода размерности массива

```
np.max(m)
```

```
9
```

Рисунок 10 – Пример вызова статистики

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

```
[[4]
 [8]
 [9]]
```

Рисунок 11 – Пример поиска максимального элемента в каждой строке

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

```
[[9 6 7 8]]
```

Рисунок 12 – Пример поиска максимального элемента в каждом столбце

```
print(m.mean())
print(m.mean(axis=1))
print(m.sum())
print(m.sum(axis=0))
```

```
4.833333333333333
[[2.5]
 [6.5]
 [5.5]]
58
[[15  9 15 19]]
```

Рисунок 13 – Пример некоторых методов для расчёта статистики

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

Рисунок 14 – Пример использования boolean массива для доступа к ndarray

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

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

Рисунок 15 – Пример построения на базе ndarray

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

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

Рисунок 16 – Пример массивов массивы с элементами типа boolean.

```
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]]
```

Рисунок 17 – Пример работы с массивами большой размерности

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

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

Рисунок 18 – Пример работы функции logical_and()

```
nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print(nums[nums < 5])
```

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

Рисунок 19 – Пример использования массива nums

```
nums[nums < 5] = 10
print(nums)
```

```
[10 10 10 10  5  6  7  8  9 10]
```

Рисунок 20 – Пример присвоения значениям меньшим 5 значение 10

```
m[m > 7] = 25
print(m)
```

```
[[ 1  2  3  4]
 [ 5  6  7 25]
 [25  1  5  7]]
```

Рисунок 21 – Пример заполнения массива числом 25

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

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

Рисунок 22 – Пример использования функции np.arange()

```
print(np.arange(5, 12))
```

```
[ 5  6  7  8  9 10 11]
```

Рисунок 23 – Пример использования np.matrix со списков в Python

```
print(np.arange(1, 5, 0.5))
```

```
[1.  1.5 2.  2.5 3.  3.5 4.  4.5]
```

Рисунок 24 – Пример использования np.array с массивом Numpy

```
print(np.matrix('[1, 2; 3, 4]'))
```

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

Рисунок 25 – Пример использования np.matrix в Matlab

```
print(np.zeros((3, 4)))
```

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

Рисунок 26 – Пример использования np.zeros()

```
print(np.eye(5))
```

```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
```

Рисунок 27 – Пример использования np.eye()

```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A)
```

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

Рисунок 28 – Пример создания двумерной матрицы размера 3x3

```
np.ravel(A)
```

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
>>> np.ravel(A, order='C')
```

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
>>> np.ravel(A, order='F')
```

```
array([1, 4, 7, 2, 5, 8, 3, 6, 9])
```

Рисунок 29 – Пример работы функции np.ravel()

```
>>> a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> np.where(a % 2 == 0, a * 10, a / 10)
```

```
array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])
```

```
a = np.random.rand(10)
print(a)
print(np.where(a > 0.5, True, False))
np.where(a > 0.5, 1, -1)
```

```
[0.33923728 0.34197629 0.34730127 0.58923732 0.14441369 0.43423286
 0.7348455  0.28702118 0.62950441 0.91576355]
[False False False  True False False  True False  True  True]

array([-1, -1, -1,  1, -1, -1,  1, -1,  1,  1])
```

Рисунок 30 – Пример работы функции np.whare()

```
>>> x = np.linspace(0, 1, 5)
print(x)
>>> y = np.linspace(0, 2, 5)
>>> y
```

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

array([0. , 0.5, 1. , 1.5, 2. ])
```

Рисунок 31 – Пример работы функции np.linspace()

```
>>> xg, yg = np.meshgrid(x, y)
print(xg)
>>> yg
```

```
[[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.  ]]

array([[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 32 – Пример работы функции np.meshgrid()

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.plot(xg, yg, color="r", marker="*", linestyle="none")
```

```
[<matplotlib.lines.Line2D at 0x17015cc1b20>,
 <matplotlib.lines.Line2D at 0x17015cc1c10>,
 <matplotlib.lines.Line2D at 0x17015cc1be0>,
 <matplotlib.lines.Line2D at 0x17015cc1c70>,
 <matplotlib.lines.Line2D at 0x17015cc1d60>]
```

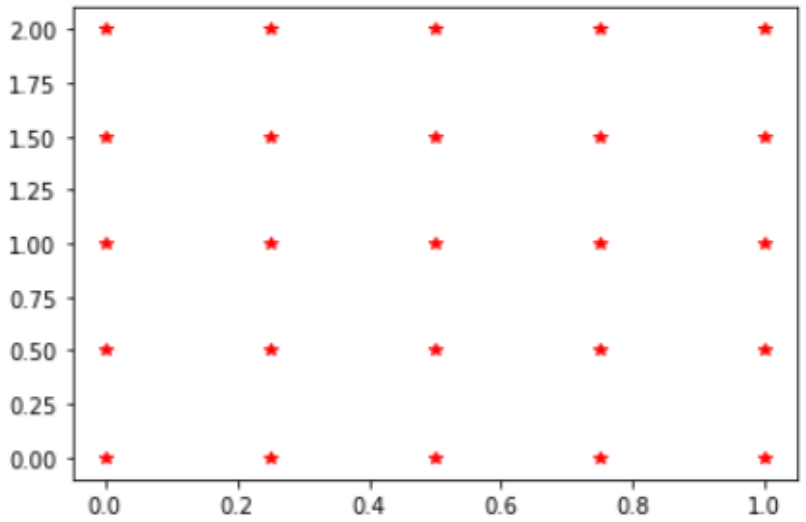


Рисунок 33 – Пример использования matplotlib

```
print(np.random.permutation(7))
>>> a = ['a', 'b', 'c', 'd', 'e']
>>> np.random.permutation(a)
```

```
[2 5 4 0 1 3 6]

array(['b', 'a', 'd', 'c', 'e'], dtype='<U1')
```

Рисунок 34 – Пример использования np.random.permutation()

```
>>> arr = np.linspace(0, 10, 5)
>>> arr
```

```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

Рисунок 35 – Пример использования np.linspace()

```
>>> arr_mix = np.random.permutation(arr)
>>> arr_mix
```

```
array([ 5. ,  7.5, 10. ,  0. ,  2.5])
```

Рисунок 36 – Пример использования np.random.permutation()

```
index_mix = np.random.permutation(len(arr_mix))
print(index_mix)
arr[index_mix]
```

```
[3 1 2 4 0]
```

```
array([ 7.5,  2.5,  5. , 10. ,  0. ])
```

Рисунок 37 – Пример использования np.random.permutation()

Арифметические операции, в отличие от операций над списками, применяются поэлементно:

```
list1 = [1, 2, 3]
array1 = np.array([1, 2, 3])

print("list1:", list1)
print('\tlist1 * 3:', list1 * 3)
print('\tlist1 + [1]:', list1 + [1])

print('array1:', array1)
print('\tarray1 * 3:', array1 * 3)
print('\tarray1 + 1:', array1 + 1)
```

```
list1: [1, 2, 3]
        list1 * 3: [1, 2, 3, 1, 2, 3, 1, 2, 3]
        list1 + [1]: [1, 2, 3, 1]
array1: [1 2 3]
        array1 * 3: [3 6 9]
        array1 + 1: [2 3 4]
```

Создайте массив из 5 чисел. Возведите каждый элемент массива в степень 3

```
c = np.array([55, 77, 27, 12, 9])
print("c = ", c)
print('\tc ** 3:', c ** 3)
```

```
c =  [55 77 27 12  9]
        c ** 3: [166375 456533  19683   1728    729]
```

Если в операции участвуют 2 массива (по умолчанию -- одинакового размера), операции считаются для соответствующих пар:

```
print("a + b =", a + b)
print("a * b =", a * b)
```

```
a + b = [1.1 2.2 3.3 4.4 5.5]
a * b = [0.1 0.4 0.9 1.6 2.5]
```

```
# вот это разность
print("a - b =", a - b)

# вот это деление
print("a / b =", a / b)

# вот это целочисленное деление
print("a // b =", a // b)

# вот это квадрат
print("a ** 2 =", a ** 2)
```

```
a - b = [0.9 1.8 2.7 3.6 4.5]
a / b = [10. 10. 10. 10. 10.]
a // b = [ 9.  9. 10.  9. 10.]
a ** 2 = [ 1  4  9 16 25]
```

Создайте два массива одинаковой длины. Выведите массив, полученный делением одного массива на другой.

```
a1 = np.array([27, 9, 2, 7, 5])
a2 = np.array([12, 22, 5, 77, 9])
print("a1 / a2 =", a1 / a2)
```

```
a1 / a2 = [2.25       0.40909091 0.4        0.09090909 0.55555556]
```

Создайте 2 массива из 5 элементов. Проверьте условие "Элементы первого массива меньше 6, элементы второго массива делятся на 3"

```
a1 = np.array([27, 9, 2, 7, 5])
a2 = np.array([12, 22, 5, 77, 9])
print("a1 =", a1)
print("a2 =", a2)
print("\t(a1 < 6) & (a2 % 3 == 0):  ", (a1<6) & (a2 % 3 == 0))
```

```
a1 = [27  9  2  7  5]
a2 = [12 22  5 77  9]
        (a1 < 6) & (a2 % 3 == 0):   [False False False False  True]
```

Теперь проверьте условие "Элементы первого массива делятся на 2 или элементы второго массива больше 2"

```
print("(a1 % 2 == 0) | (a2 > 2):  ", (a1%2==0) | (a2>2))
```

```
(a1 % 2 == 0) | (a2 > 2):   [ True  True  True  True  True]
```

Зачем это нужно? Чтобы выбирать элементы массива, удовлетворяющие какому-нибудь условию:

```
print("a =", a)
print("a > 2:", a > 2)
# индексация - выбираем элементы из массива в тех позициях, где True
print("a[a > 2]:", a[a > 2])
```

```
a = [1 2 3 4 5]
a > 2: [False False  True  True  True]
a[a > 2]: [3 4 5]
```

Создайте массив с элементами от 1 до 20. Выведите все элементы, которые больше 5 и не делятся на 2

Подсказка: создать массив можно с помощью функции np.arange(), действие которой аналогично функции range, которую вы уже знаете.

```
a3 = np.arange(5,27)
print("a3 =", a3)
print("a3[a3 > 5 & a3 % 2 != 0]:", a3[np.logical_and(a3 > 5, a3 % 2 != 0)])
```

```
a3 = [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26]
a3[a3 > 5 & a3 % 2 != 0]: [ 7  9 11 13 15 17 19 21 23 25]
```

Пора еще немного потренироваться с NumPy.

Выполните операции, перечисленные ниже:

```
print("Разность между a и b:", a - b
     )
print("Квадраты элементов b:", b ** 2
     )
print("Половины произведений элементов массивов a и b:", a * b / 2
     )

print()
print("Максимальный элемент b:", np.max(b)
     )
print("Сумма элементов массива b:", np.sum(b)
     )
print("Индекс максимального элемента b:", np.argmax(b)
     )
```

```
Разность между a и b: [0.9 1.8 2.7 3.6 4.5]
Квадраты элементов b: [0.01 0.04 0.09 0.16 0.25]
Половины произведений элементов массивов a и b: [0.05 0.2  0.45 0.8  1.25]

Максимальный элемент b: 0.5
Сумма элементов массива b: 1.5
Индекс максимального элемента b: 4
```

Задайте два массива: [5, 2, 3, 12, 4, 5] и ['f', 'o', 'o', 'b', 'a', 'r']

Выведите буквы из второго массива, индексы которых соответствуют индексам чисел из первого массива, которые больше 1, меньше 5 и делятся на 2

```
arr = np.array([5, 2, 3, 12, 4, 5])
letters = np.array(['f', 'o', 'o', 'b', 'a', 'r'])
print(letters[np.logical_and(arr > 1, arr < 5, arr % 2 == 0)])
```

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

Рисунок 38 – Результат выполнения заданий

Задание №1 ¶

Создайте два массива: в первом должны быть четные числа от 2 до 12 включительно, а в другом числа 7, 11, 15, 18, 23, 29.

1. Сложите массивы и возведите элементы получившегося массива в квадрат:

```
import numpy as np
a = np.arange(2, 8)
b = np.array([7, 11, 15, 18, 23, 29])
print("Сумма массивов: ",a + b)
print("Элементы в квадрате: ",(a + b) ** 2)
```

```
Сумма массивов:  [ 9 14 19 23 29 36]
Элементы в квадрате:  [  81  196  361  529  841 1296]
```

2. Выведите все элементы из первого массива, индексы которых соответствуют индексам тех элементов второго массива, которые больше 12 и дают остаток 3 при делении на 5.

```
print(a[np.logical_and(b > 12, b % 5 == 3)])
```

```
[5 6]
```

3. Проверьте условие "Элементы первого массива делятся на 4, элементы второго массива меньше 14". (Подсказка: в результате должен получиться массив с True и False)

```
print((a % 4 == 0) & (b < 14))
```

Задание №2

- Найдите интересный для вас датасет. Например, можно выбрать датасет тут. http://data.un.org/Explorer.aspx (выбираете датасет, жмете на view data, потом download, выбирайте csv формат)
- Рассчитайте подходящие описательные статистики для признаков объектов в выбранном датасете
- Проанализируйте и прокомментируйте содержательно получившиеся результаты
- Все комментарии оформляйте строго в ячейках формата markdown

```
import csv
import numpy as np
```

```
with open('rost.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    men_height = []
    women_height = []
    last10_men = []
    for row in data:
        if row[0] != "Rank" and int(row[0]) <= 10:
            men_height.append(float(row[2]))
            women_height.append(float(row[3]))
        if row[0] != "Rank" and 180 < int(row[0]) <= 190:
            last10_men.append(float(row[2]))
```

```
men_arr = np.array(men_height)
women_arr = np.array(women_height)
print(f"Среднее значение роста мужчин из топ 10 стран: {np.mean(men_arr)}")
print(f"Среднее значение роста женщин из топ 10 стран: {np.mean(women_arr)}")
print(f"Среднее отклонение для мужчин: {np.std(men_arr)}")
print(f"Среднее отклонение для женщин: {np.std(women_arr)}")
```

```
Среднее значение роста мужчин из топ 10 стран: 182.06900000000002
Среднее значение роста женщин из топ 10 стран: 168.59199999999998
Среднее отклонение для мужчин: 0.9499310501294319
Среднее отклонение для женщин: 1.080932930389304
```

Рисунок 39 – Результат выполнения заданий

ВОПРОСЫ

1. Каково назначение библиотеки NumPy?

Numpy – это библиотека для языка программирования Python, которая предоставляет в распоряжение разработчика инструменты для эффективной работы с многомерными массивами и высокопроизводительные вычислительные алгоритмы.

2. Что такое массивы ndarray?

Ndarray — это (обычно фиксированный размер) многомерный контейнер элементов одного типа и размера. Количество измерений и элементов в массиве

определяется его формой, которая является кортежем из N натуральных чисел, которые определяют размеры каждого измерения.

3. Как осуществляется доступ к частям многомерного массива?

Элементы матрицы с заданными координатами: m[1,0]

Строка матрицы: m[1, :]

Столбец матрицы: m[:, 1]

Часть строки матрицы: m[1, 2:]

Часть столбца матрицы: m[0:2, 1]

Непрерывная часть матрицы: m[0:2, 1:3]

Произвольные столбцы / строки матрицы: cols = [0, 1, 2]; m[:, cols]

4. Как осуществляется расчет статистик по данным?

Размерность массива: m.shape

Вызов функции расчёта статистики: m.max()

Расчёт статистики по строкам или столбцам массива: m.max(axis=1); m.max(axis=0)

Индексы элементов с максимальным значением (по осям): argmax

Индексы элементов с минимальными значением (по осям): argmin

Максимальные значения элементов (по осям): max

Минимальные значения элементов (по осям): min

Средние значения элементов (по осям): mean

Произведение всех элементов (по осям): prod

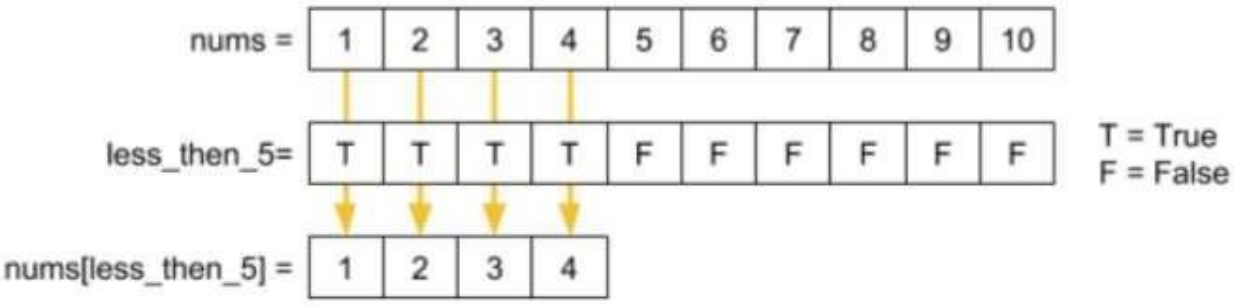
Стандартное отклонение (по осям): std

Сумма всех элементов (по осям): sum Дисперсия (по осям): var

5. Как выполняется выборка данных из массивов ndarray?

```
>>> less_then_5 = nums < 5
>>> less_then_5
array([ True,  True,  True,  True, False, False, False, False, False,
       False])
```

Если мы переменную *less_then_5* передадим в качестве списка индексов для *nums*, то получим массив, в котором будут содержаться элементы из *nums* с индексами равными индексам *True* позиций массива *less_then_5*, графически это будет выглядеть так.




```
>>> nums[less_then_5]
array([1, 2, 3, 4])
```