

**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
Симоненко А.С. « » _____ 2022г.
Подпись студента _____
Работа защищена « » _____ 2022г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь, 2022 г.

Примеры

```
In [1]: 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]]
```

```
In [2]: m[1, 0]
```

```
Out[2]: 5
```

```
In [3]: m[1, :]
```

```
Out[3]: matrix([[5, 6, 7, 8]])
```

```
In [4]: m[:, 2]
```

```
Out[4]: matrix([[3],
                [7],
                [5]])
```

```
In [6]: m[1, 2:]
```

```
Out[6]: matrix([[7, 8]])
```

```
In [7]: m[0:2, 1]
```

```
Out[7]: matrix([[2],
                [6]])
```

```
In [8]: m[0:2, 1:3]
```

```
Out[8]: matrix([[2, 3],
                [6, 7]])
```

```
In [9]: cols = [0, 1, 3]
m[:, cols]
```

```
Out[9]: matrix([[1, 2, 4],
                [5, 6, 8],
                [9, 1, 7]])
```

```
In [10]: 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]]
```

```
In [11]: type(m)
```

```
Out[11]: numpy.matrix
```

```
In [12]: m = np.array(m)
type(m)
```

```
Out[12]: numpy.ndarray
```

```
In [13]: m.shape
```

```
Out[13]: (3, 4)
```

```
In [14]: m.max()
```

```
Out[14]: 9
```

```
In [15]: np.max(m)
```

```
Out[15]: 9
```

```
In [16]: m.max()
```

```
Out[16]: 9
```

```
In [17]: m.max(axis=1)
```

```
Out[17]: array([4, 8, 9])
```

```
In [18]: m.max(axis=0)
```

```
Out[18]: array([9, 6, 7, 8])
```

```
In [19]: m.mean()
```

```
Out[19]: 4.833333333333333
```

```
In [20]: m.mean(axis=1)
```

```
Out[20]: array([2.5, 6.5, 5.5])
```

```
In [21]: m.sum()
```

```
Out[21]: 58
```

```
In [22]: m.sum(axis=0)
```

```
Out[22]: array([15, 9, 15, 19])
```

```
In [25]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])
```

```
In [26]: a = True
```

```
In [27]: b = 5 > 7
print(b)

False
```

```
In [28]: less_than_5 = nums < 5
less_than_5
```

```
Out[28]: array([ True,  True,  True,  True, False, False, False, False, False,
        False])
```

```
In [29]: pos_a = letters == 'a'
pos_a
```

```
Out[29]: array([ True, False, False, False,  True, False, False])
```

```
In [30]: less_than_5 = nums < 5
less_than_5
```

```
Out[30]: array([ True,  True,  True,  True, False, False, False, False, False,
        False])
```

```
In [31]: nums[less_than_5]
```

```
Out[31]: array([1, 2, 3, 4])
```

```
In [33]: 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]]
```

```
In [34]: mod_m = np.logical_and(m>=3, m<=7)
mod_m
```

```
Out[34]: matrix([[False, False,  True,  True],
 [ True,  True,  True, False],
 [False, False,  True,  True]])
```

```
In [35]: m[mod_m]
```

```
Out[35]: matrix([[3, 4, 5, 6, 7, 5, 7]])
```

```
In [37]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
nums[nums < 5]
```

```
Out[37]: array([1, 2, 3, 4])
```

```
In [38]: nums[nums < 5] = 10
print(nums)
```

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

```
In [39]: m[m > 7] = 25
print(m)
```

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

```
In [40]: np.arange(10)
```

```
Out[40]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [41]: np.arange(5, 12)
```

```
Out[41]: array([ 5,  6,  7,  8,  9, 10, 11])
```

```
In [42]: np.arange(1, 5, 0.5)
```

```
Out[42]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

```
In [44]: a = [[1, 2], [3, 4]]
np.matrix(a)
```

```
Out[44]: matrix([[1, 2],
 [3, 4]])
```

```
In [45]: b = np.array([[5, 6], [7, 8]])
np.matrix(b)
```

```
Out[45]: matrix([[5, 6],
 [7, 8]])
```

```
In [46]: np.matrix('1, 2; 3, 4')
```

```
Out[46]: matrix([[1, 2],
 [3, 4]])
```

```
In [47]: np.zeros((3, 4))
```

```
Out[47]: array([[0., 0., 0., 0.],  
               [0., 0., 0., 0.],  
               [0., 0., 0., 0.]])
```

```
In [48]: np.eye(3)
```

```
Out[48]: array([[1., 0., 0.],  
               [0., 1., 0.],  
               [0., 0., 1.]])
```

```
In [49]: A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
A
```

```
Out[49]: array([[1, 2, 3],  
               [4, 5, 6],  
               [7, 8, 9]])
```

```
In [50]: np.ravel(A)
```

```
Out[50]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [51]: np.ravel(A, order='C')
```

```
Out[51]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [52]: np.ravel(A, order='F')
```

```
Out[52]: array([1, 4, 7, 2, 5, 8, 3, 6, 9])
```

```
In [54]: a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])  
np.where(a % 2 == 0, a * 10, a / 10)
```

```
Out[54]: array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])
```

```
In [55]: a = np.random.rand(10)  
a
```

```
Out[55]: array([0.2902933 , 0.11386424, 0.38329769, 0.99187813, 0.61577763,  
               0.48305957, 0.87908297, 0.41960383, 0.81566398, 0.60689004])
```

```
In [56]: np.where(a > 0.5, True, False)
```

```
Out[56]: array([False, False, False,  True,  True, False,  True, False,  True,  
                True])
```

```
In [57]: np.where(a > 0.5, 1, -1)
```

```
Out[57]: array([-1, -1, -1,  1,  1, -1,  1, -1,  1,  1])
```

```
In [59]: x = np.linspace(0, 1, 5)  
x
```

```
Out[59]: array([0. , 0.25, 0.5 , 0.75, 1.  ])
```

```
In [60]: y = np.linspace(0, 2, 5)  
y
```

```
Out[60]: array([0. , 0.5, 1. , 1.5, 2.  ])
```

```
In [61]: xg, yg = np.meshgrid(x, y)  
xg
```

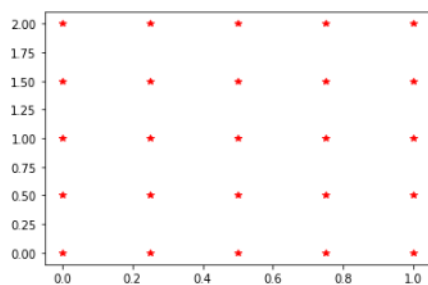
```
Out[61]: array([[0. , 0.25, 0.5 , 0.75, 1.  ],  
               [0. , 0.25, 0.5 , 0.75, 1.  ],  
               [0. , 0.25, 0.5 , 0.75, 1.  ],  
               [0. , 0.25, 0.5 , 0.75, 1.  ],  
               [0. , 0.25, 0.5 , 0.75, 1.  ]])
```

```
In [62]: yg
```

```
Out[62]: 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.  ]])
```

```
In [64]: import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(xg, yg, color="r", marker="*", linestyle="none")
```

```
Out[64]: [<matplotlib.lines.Line2D at 0xffff65d764f0>,
<matplotlib.lines.Line2D at 0xffff65d76610>,
<matplotlib.lines.Line2D at 0xffff65d76730>,
<matplotlib.lines.Line2D at 0xffff65d76850>,
<matplotlib.lines.Line2D at 0xffff65d76970>]
```



```
In [65]: np.random.permutation(7)
```

```
Out[65]: array([6, 4, 2, 0, 1, 5, 3])
```

```
In [66]: a = ['a', 'b', 'c', 'd', 'e']
```

```
In [67]: np.random.permutation(a)
```

```
Out[67]: array(['d', 'e', 'a', 'c', 'b'], dtype='<U1'))
```

```
In [69]: arr = np.linspace(0, 10, 5)
arr
```

```
Out[69]: array([ 0. , 2.5, 5. , 7.5, 10. ])
```

```
In [70]: arr_mix = np.random.permutation(arr)
arr_mix
```

```
Out[70]: array([ 2.5, 0. , 7.5, 10. , 5. ])
```

```
In [72]: index_mix = np.random.permutation(len(arr_mix))
index_mix
```

```
Out[72]: array([4, 2, 1, 0, 3])
```

```
In [73]: arr[index_mix]
```

```
Out[73]: array([10. , 5. , 2.5, 0. , 7.5])
```

Индивидуальное задание

1. Дана целочисленная прямоугольная матрица. Определить:

- количество строк, не содержащих ни одного нулевого элемента;
- максимальное из чисел, встречающихся в заданной матрице более одного раза.

```
In [106]: import numpy as np
import heapq as hq

# Дана целочисленная прямоугольная матрица. Определить:
# количество строк, не содержащих ни одного нулевого элемента;
# максимальное из чисел, встречающихся в заданной матрице более одного раза

mass = [
[3, 5, 6],
[0, 1, 0],
[8, 5, 2]]

np_mass = np.array(mass) #NumPy - Array

print("mass = \n", np_mass, "\n")
print("\t1) количество строк, не содержащих ни одного нулевого элемента:", np.sum(np.all(np_mass != 0, axis=1)) )
print("\t2) максимальное из чисел, встречающихся в заданной матрице более одного раза:", np.max(np_mass))

mass =
[[3 5 6]
[0 1 0]
[8 5 2]]
```

- 1) количество строк, не содержащих ни одного нулевого элемента: 2
- 2) максимальное из чисел, встречающихся в заданной матрице более одного раза: 8

Лабораторная работа 3.2. Знакомство с NumPy

```
In [15]: # подключение модуля numpy под именем np
import numpy as np
```

```
In [16]: # основная структура данных - массив
a = np.array([1, 2, 3, 4, 5])
b = np.array([0.1, 0.2, 0.3, 0.4, 0.5])
```

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

```
a = [1 2 3 4 5]
b = [0.1 0.2 0.3 0.4 0.5]
```

Создайте массив с 5 любыми числами:

```
In [29]: c = np.random.randint(1, 10, 5)
c
```

```
Out[29]: array([9, 8, 6, 8, 8])
```

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

```
In [30]: 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

```
In [31]: c = np.random.randint(1, 10, 5)
print(c)
print(f"c^3: {c**3}")

[1 9 4 1 5]
c^3: [ 1 729 64 1 125]
```

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

```
In [32]: 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]
```

```
In [33]: # 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.

```
In [35]: d = np.random.randint(1, 10, 5)
e = np.random.randint(1, 10, 5)
print(f"d / e = {d/e}")

d / e = [3.         1.         3.         1.66666667 1.         ]
```

Л — логика

К элементам массива можно применять логические операции.

Возвращаемое значение -- массив, содержащий результаты вычислений для каждого элемента (True -- "да" или False -- "нет").

```
In [36]: print("a =", a)
print("\ta > 1: ", a > 1)
print("\nb =", b)
print("\tb < 0.5: ", b < 0.5)

print("\nОдновременная проверка условий:")
print("\t(a > 1) & (b < 0.5): ", (a > 1) & (b < 0.5))
print("А вот это проверяет, что a > 1 ИЛИ b < 0.5: ", (a > 1) | (b < 0.5))

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

b = [0.1 0.2 0.3 0.4 0.5]
b < 0.5: [ True True True True False]

Одновременная проверка условий:
(a > 1) & (b < 0.5): [False True True True False]
А вот это проверяет, что a > 1 ИЛИ b < 0.5: [ True True True True True]
```

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

```
In [37]: d = np.random.randint(1, 10, 5)
e = np.random.randint(1, 10, 5)
log1 = d < 6
log2 = e % 3 == 0
print(d)
print(e)
print(log1)
print(log2)

[4 2 7 6 8]
[5 5 1 5 4]
[ True  True False False False]
[False False False False False]
```

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

```
In [38]: log3 = np.logical_or(d % 2 == 0, e > 2)
print(log3)

[ True  True False  True  True]
```

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

```
In [39]: 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.

```
In [40]: d = np.arange(1, 21)
print(d)
d = d[np.logical_and(d > 5, d % 2 != 0)]
print(d)

[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
[ 7  9 11 13 15 17 19]
```

А ещё NumPy умеет...

Все операции NumPy оптимизированы для быстрых вычислений над целыми массивами чисел и в методах `np.array` реализовано множество функций, которые могут вам понадобиться:

```
In [41]: # теперь можно считать средний размер котиков в одну строку!
print("np.mean(a) =", np.mean(a))
# минимальный элемент
print("np.min(a) =", np.min(a))
# индекс минимального элемента
print("np.argmin(a) =", np.argmin(a))
# вывести значения массива без дубликатов
print("np.unique(['male', 'male', 'female', 'female', 'male']) =", np.unique(['male', 'male', 'female', 'female', 'male']))

# и ещё много всяких методов
# Google в помощь

np.mean(a) = 3.0
np.min(a) = 1
np.argmin(a) = 0
np.unique(['male', 'male', 'female', 'female', 'male']) = ['female' 'male']
```

Пора еще немного потренироваться с NumPy.

Выполните операции, перечисленные ниже:

```
In [42]: print("Разность между a и b:", # YOUR CODE
)
print("Квадраты элементов b:", # YOUR CODE
)
print("Половины произведений элементов массивов a и b:", # YOUR CODE
)

print()
print("Максимальный элемент b:", # YOUR CODE
)
print("Сумма элементов массива b:", # YOUR CODE
)
print("Индекс максимального элемента b:", # YOUR CODE
)
```

Разность между a и b:
Квадраты элементов b:
Половины произведений элементов массивов a и b:

Максимальный элемент b:
Сумма элементов массива b:
Индекс максимального элемента b:

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

```
In [43]: d = np.array([5, 2, 3, 12, 4, 5])
e = np.array(['f', 'o', 'o', 'b', 'a', 'r'])
log1 = np.logical_and(d > 1, d < 5, d % 2 == 0)
print(log1)
print(e[log1])

[False True  True False  True False]
['o' 'o' 'a']
```


Лабораторная работа 3.2. Домашнее задание

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.

```
In [11]: import numpy as np
```

```
In [12]: a = np.arange(2, 13, 2)
b = np.array([7, 11, 15, 18, 23, 29])
print(a)
print(b)

[ 2  4  6  8 10 12]
[ 7 11 15 18 23 29]
```

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.

```
In [13]: loc = np.logical_and(b > 12, b % 5 == 3)
a[loc]
```

```
Out[13]: array([ 8, 10])
```

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

```
In [14]: log1 = a % 4 == 0
log2 = b < 14
print(log1 + log2)

[ True  True False  True False  True]
```

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

```
In [1]: import os.path
import numpy as np
import csv

file_csv = './UNdata_Export_20220403_173018777.csv'

if os.path.isfile(file_csv):
    # Load file
    data = np.array([])
    with open(file_csv, newline='') as csvfile:
        # Данные структура
        data = {
            "CountryOrArea" : np.array([]),
            "Year" : np.array([], dtype=np.int32),
            "Commodity" : np.array([]),
            "Flow" : np.array([]),
            "TradeUSD" : np.array([]),
            "WeightKg" : np.array([]),
            "QuantityName" : np.array([]),
            "Quantitykg" : np.array([]),
        }

    ldata = list(csv.reader(csvfile))

    for line in ldata[1:500]:
        data["CountryOrArea"] = np.append(data["CountryOrArea"], line[0])
        data["Year"] = np.append(data["Year"], int(line[1]))
        data["Commodity"] = np.append(data["Commodity"], line[2])
        data["Flow"] = np.append(data["Flow"], line[3])
        data["TradeUSD"] = np.append(data["TradeUSD"], float(line[4]))
        data["WeightKg"] = np.append(data["WeightKg"], line[5])
        data["QuantityName"] = np.append(data["QuantityName"], line[6])
        data["Quantitykg"] = np.append(data["Quantitykg"], line[7])

    ldata = None
else:
    print("Not found file:", file_csv)
```

```

print("Общий:")
print("\tСтрана или область:")
print("\t\t количество страна:", len(np.unique(data["CountryOrArea"])))
print()

print("\tГод:")
print("\t\t от:", np.min(data["Year"]), " до:", np.max(data["Year"]))
print()

print("\tТовар:") #
print("\t\t \"ALL COMMODITIES\" (кол.) = ", len(data["Commodity"]))
print()

print("\tПоток:", np.unique(data["Flow"]))
print()

print("\tТорговля (долл. США):")
print("\t\t сумма:", np.sum(data["TradeUSD"]), "кг.")
print("\t\t минимальный:", "%.2f" % np.min(data["TradeUSD"]), "кг.")
print("\t\t максимальный:", "%.2f" % np.max(data["TradeUSD"]), "кг.")
print("\t\t среднее арифметическое значений:", "%.2f" % np.mean(data["TradeUSD"]), "кг.")
print("\t\t стандартное отклонение:", np.std(data["TradeUSD"]), "кг.")
print("\t\t дисперсия значений:", "%.2f" % np.var(data["TradeUSD"]), "кг.")
print()

print("\tВес (кг): 0")
print()

print("\tНазвание количества:")
print("\t\t \"No Quantity\" (кол.)\" =", len(data["QuantityName"]))
print()

print("\tКоличество (кг): 0")

```

Общий:

Страна или область:
количество страна: 11

Год:
от: 1992 до: 2019

Товар:
"ALL COMMODITIES" (кол.) = 499

Поток: ['Export' 'Import' 'Re-Export' 'Re-Import']

Торговля (долл. США):
сумма: 10524679880320.248 кг.
минимальный: 392.00 кг.
максимальный: 269423385464.00 кг.
среднее арифметическое значений: 21091542846.33 кг.
стандартное отклонение: 48600048470.16132 кг.
дисперсия значений: 2361964711302029901824.00 кг.

Вес (кг): 0

Название количества:
"No Quantity (кол.)" = 499

Количество (кг): 0

Самостоятельно задание

```
In [3]: from math import sin
from numpy import array, arange
from pylab import plot, xlabel, show
```

```
def f(r,t):
    x = r[0]
    y = r[1]
    fx = x*y - x
    fy = y - x*y + sin(t)**2
    return array([fx,fy],float)
```

```
a = 0.0
b = 10.0
N = 1000
h = (b-a)/N
```

```
tpoints = arange(a,b,h)
xpoints = []
ypoints = []
```

```
r = array([1.0,1.0],float)
for t in tpoints:
    xpoints.append(r[0])
    ypoints.append(r[1])
    k1 = h*f(r,t)
    k2 = h*f(r+0.5*k1,t+0.5*h)
    k3 = h*f(r+0.5*k2,t+0.5*h)
    k4 = h*f(r+k3,t+h)
    r += (k1+2*k2+2*k3+k4)/6
```

```
plot(tpoints,xpoints)
plot(tpoints,ypoints)
xlabel("t")
show()
```

