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

Отчет о лабораторной работе № 3.2 Основы работы с библиотекой NumPy

Выполнил:
Шальнев Владимир Сергеевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

Проработанные примеры:

Примеры лабораторной работы 3.2

Ввод [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]]
```

Ввод [2]:
```
m[1, 0]
```

Out[2]: 5

Ввод [3]:
```
m[1, :]
```

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

Ввод [4]:
```
m[:, 2]
```

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

Ввод [5]:
```
m[1, 2:]
```

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

Ввод [6]:
```
m[0:2, 1]
```

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

Ввод [7]:
```
m[0:2, 1:3]
```

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

Ввод [8]:
```
cols = [0, 1, 3]
m[:, cols]
```

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

Ввод [9]:
```
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]]
```

Ввод [10]:
```
type(m)
```

Out[10]: numpy.matrix

Ввод [11]:
```
m = np.array(m)
type(m)
```

Out[11]: numpy.ndarray

Ввод [12]:
```
m.shape
```

Out[12]: (3, 4)

Ввод [13]:
```
m.max()
```

Out[13]: 9

Ввод [14]:
```
np.max(m)
```

Out[14]: 9

Ввод [15]:
```
m.max()
```

Out[15]: 9

Ввод [16]:
```
m.max(axis=1)
```

Out[16]:
```
array([4, 8, 9])
```

Ввод [17]:
```
m.max(axis=0)
```

Out[17]:
```
array([9, 6, 7, 8])
```

Ввод [18]:
```
m.mean()
```

Out[18]:
```
4.833333333333333
```

Ввод [19]:
```
m.mean(axis=1)
```

Out[19]:
```
array([2.5, 6.5, 5.5])
```

Ввод [20]:
```
m.sum()
```

Out[20]:
```
58
```

Ввод [21]:
```
m.sum(axis=0)
```

Out[21]:
```
array([15,  9, 15, 19])
```

Ввод [22]:
```
nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])
```

Ввод [23]:
```
a = True
```

Ввод [24]:
```
b = 5 > 7
print(b)
```

```
False
```

Ввод [25]:
```
less_then_5 = nums < 5
less_then_5
```

Out[25]:
```
array([ True,  True,  True,  True, False, False, False, False, False,
       False])
```

Ввод [26]:
```
pos_a = letters == 'a'
pos_a
```

Out[26]:
```
array([ True, False, False, False,  True, False, False])
```

Ввод [27]:
```
less_then_5 = nums < 5
less_then_5
```

Out[27]:
```
array([ True,  True,  True,  True, False, False, False, False, False,
       False])
```

Ввод [28]:
```
nums[less_then_5]
```

Out[28]:
```
array([1, 2, 3, 4])
```

Ввод [29]:
```
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]]
```

Ввод [30]:
```
mod_m = np.logical_and(m>=3, m<=7)
mod_m
```

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

Ввод [31]:
```
m[mod_m]
```

Out[31]:
```
matrix([[3, 4, 5, 6, 7, 5, 7]])
```

Ввод [32]:
```
nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
nums[nums < 5]
```

Out[32]:
```
array([1, 2, 3, 4])
```

Ввод [33]:
```
nums[nums < 5] = 10
print(nums)
```

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

Ввод [34]:
```
m[m > 7] = 25
print(m)
```

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

Ввод [35]:
```
np.arange(10)
```

Out[35]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Ввод [36]:
```
np.arange(5, 12)
```

Out[36]:
```
array([ 5,  6,  7,  8,  9, 10, 11])
```

Ввод [37]:
```
np.arange(1, 5, 0.5)
```

Out[37]:
```
array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

Ввод [38]:
```
a = [[1, 2], [3, 4]]
np.matrix(a)
```

Out[38]:
```
matrix([[1, 2],
        [3, 4]])
```

Ввод [39]:
```
b = np.array([[5, 6], [7, 8]])
np.matrix(b)
```

Out[39]:
```
matrix([[5, 6],
        [7, 8]])
```

Ввод [40]:
```
np.matrix('[1, 2; 3, 4]')
```

Out[40]:
```
matrix([[1, 2],
        [3, 4]])
```

Ввод [41]:
```
np.zeros((3, 4))
```

Out[41]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

Ввод [42]:
```
np.eye(3)
```

Out[42]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Ввод [43]:
```
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
A
```

Out[43]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Ввод [44]:
```
np.ravel(A)
```

Out[44]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Ввод [45]:
```
np.ravel(A, order='C')
```

Out[45]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Ввод [46]:
```
np.ravel(A, order='F')
```

Out[46]:
```
array([1, 4, 7, 2, 5, 8, 3, 6, 9])
```

Ввод [47]:
```
a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
np.where(a % 2 == 0, a * 10, a / 10)
```

Out[47]:
```
array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])
```

Ввод [48]:
```
a = np.random.rand(10)
a
```

Out[48]:
```
array([0.69822327, 0.13136168, 0.17973289, 0.71766829, 0.61578219,
       0.27879863, 0.29516817, 0.09025228, 0.77594051, 0.9537352 ])
```

Ввод [49]:
```
np.where(a > 0.5, True, False)
```

Out[49]:
```
array([ True, False, False,  True,  True, False, False, False,  True,
        True])
```

Ввод [50]:
```
np.where(a > 0.5, 1, -1)
```

Out[50]:
```
array([ 1, -1, -1,  1,  1, -1, -1, -1,  1,  1])
```

Ввод [51]:
```
x = np.linspace(0, 1, 5)
x
```

Out[51]:
```
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

Ввод [52]:
```
y = np.linspace(0, 2, 5)
y
```

Out[52]:
```
array([0. , 0.5, 1. , 1.5, 2. ])
```

Ввод [53]:
```
xg, yg = np.meshgrid(x, y)
xg
```

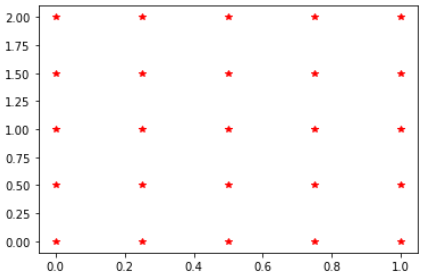
Out[53]:
```
array([[0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ],
       [0.  , 0.25, 0.5 , 0.75, 1.  ]])
```

Ввод [54]:
```
yg
```

Out[54]:
```
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. ]])
```

Ввод [55]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(xg, yg, color="r", marker="*", linestyle="none")
```

Out[55]:
```
[<matplotlib.lines.Line2D at 0x1ae4cbbeb50>,
 <matplotlib.lines.Line2D at 0x1ae4cbbec40>,
 <matplotlib.lines.Line2D at 0x1ae4cbbec10>,
 <matplotlib.lines.Line2D at 0x1ae4cbbeca0>,
 <matplotlib.lines.Line2D at 0x1ae4cbbed90>]
```



Ввод [56]:
```
np.random.permutation(7)
```

Out[56]:
```
array([2, 3, 0, 5, 1, 4, 6])
```

Ввод [57]:
```
a = ['a', 'b', 'c', 'd', 'e']
np.random.permutation(a)
```

Out[57]:
```
array(['d', 'b', 'a', 'e', 'c'], dtype='<U1')
```

Ввод [58]:
```
arr = np.linspace(0, 10, 5)
arr
```

Out[58]:
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

Ввод [59]:
```
arr_mix = np.random.permutation(arr)
arr_mix
```

Out[59]:
```
array([ 5. ,  2.5, 10. ,  0. ,  7.5])
```

Ввод [60]:
```
index_mix = np.random.permutation(len(arr_mix))
index_mix
```

Out[60]:
```
array([0, 1, 4, 2, 3])
```

Ввод [61]:
```
arr[index_mix]
```

Out[61]:
```
array([ 0. ,  2.5, 10. ,  5. ,  7.5])
```

Задача:

Для заданной матрицы размером 8 на 8 найти такие k, что k-я строка матрицы совпадает с k-м столбцом.

Первая индивидуальная задача

Ввод [1]:
```
import numpy as np
```

Ввод [2]:
```
matrix = np.matrix('-1 2 3 4 5 6 7 8;'
                   '2 3 4 5 6 7 8 5;'
                   '3 4 5 6 7 8 9 5;'
                   '4 5 6 7 8 9 0 5;'
                   '5 6 7 8 9 0 1 5;'
                   '6 7 8 9 0 1 2 5;'
                   '7 8 9 0 1 2 -3 4;'
                   '8 9 0 1 2 3 4 5')
```

Ввод [3]:
```
print(matrix)
```

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

Ввод [4]:
```
condition = matrix == matrix.transpose()
condition = condition.prod(axis=0)
condition = np.ravel(np.where(condition == 1, True, False))
k = np.arange(condition.size)[condition]

print(f"Значения k, такие, что k-я строка "
      f"матрицы совпадает с k-м столбцом: {k}")
```

```
Значения k, такие, что k-я строка матрицы совпадает с k-м столбцом: [0 6]
```

Решение первой индивидуальной задачи

Задача:

Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

Вторая индивидуальная задача

Ввод [5]:
```
condition = matrix < 0
condition = condition.sum(axis=0)
condition = np.ravel(np.where(condition == 1, True, False))
ans = np.ravel(matrix[condition].sum(axis = 1))
print(f"Суммы элементов в строках, которые содержат "
      f"хотя бы один отрицательный элемент: {ans}")
```

```
Суммы элементов в строках, которые содержат хотя бы один отрицательный элемент:
[34 28]
```

Решение второй индивидуальной задачи

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, 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]
```

Решение выданных ноутбуков

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.

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

```
array([ 8, 10])
```

Решение выданных ноутбуков

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

```
log1 = a % 4 == 0
log2 = b < 14
print(log1 + log2)
```

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

Решение выданных ноутбуков

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

Ищем зависимость числа рождения от уровня грамотности Число рождения - зависимая y Уровень грамотности - независимая x

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

```
# Данные формата Страна, Год, детей рождено, уровень грамотности
with open('Годовое число рождений.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    birth = [["Country", "Year", "Value"]]
    for row in data:
        if row[5].isdigit():
            birth.append([row[1], int(row[5]), int(row[11])])

with open('Уровень грамотности взрослого населения.csv', 'r', newline='') as csv
    data = csv.reader(csvfile, delimiter=',')
    literacy = [["Country", "First year", "Last year", "Value"]]
    for row in data:
        if row[11].isdigit():
            f_year, l_year = map(int, row[5].split('-'))
            literacy.append([row[1], f_year, l_year, int(row[11])])

data = [["Country", "Year", "Birth", "Literacy"]]
for row1 in birth:
    for row2 in literacy:
        if row1[0] == row2[0] and row1[1] >= row2[1] and row1[1] <= row2[2]:
            data.append([row1[0], int(row1[1]), int(row1[2]), int(row2[3])])
data = np.array(data)
```

Решение выданных ноутбуков

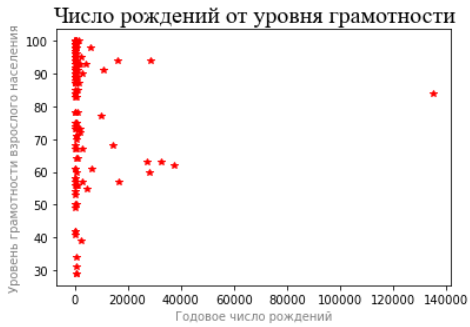
```
birth = np.int_(np.array(data[1:, 2]))
literacy = np.int_(np.array(data[1:, 3]))
print(birth)
print(literacy)
```

```
[    41    712    803    693     47    184     23   3016    107    356
     15    264     32     47   2996      8     75    730    288    317
    716     10   5823    156    511    245  16364    910     28     73
    679     43    110     13    348   2912    216  28448  14399    298
   1886    126     26    193     16   2613     42     67     51    776
    117    473    394     59    266    205    100  27098   4331   1255
   1144    557     50    154    345   1560     50    131    140  10790
     24  28334     65     60    157    144     35    747    686    579
      5    728      4    118     16   2195  10017     65      8    620
    889    824     60    722    138    777   6458     50   4764     70
    208    158    591   2358    410     97     21     44    221   1689
    449      4      5    605    471    110    227     47     20   1052
  37402    499    373    137  32584     10     35    466    194    824
     22     44    195      3     20    179   1289    109   1545    494
     94   1913     49    589      7    598   1458  16712 135056    940
    622]
[ 96  73  70  98 100 100  92  57 100  42  53  91  98  84  90  95  98  29
  67  74  71  84  98  56  34  99  94  93  75  96  56  99 100  98 100  67
  90  94  68  92  72  84  94  68 100  39  88  50 100  67  97  75  41  54
  49  85  99  63  93  85  78  99  87  93 100  87  94  99  73  91 100  60
  90  90  61  89 100  64  75  93  98  31  92  58  89  93  77  97  98  56
  56  92  89  60  78  29  61  87  55  94  61  94  90  95 100  95  96  99
  98 100  71  99  89  87  50  98  42  96 100  89  62  98  91  95  63  95
  87  83 100  94  97  58  57  99  99  78  91 100  73 100  90  73  98  99
  83  96  93  57  84  64  71]
```

Решение выданных ноутбуков

```
plt.title('Число рождений от уровня грамотности', fontsize=20, fontname='Times N
plt.xlabel('Годовое число рождений', color='gray')
plt.ylabel('Уровень грамотности взрослого населения',color='gray')
plt.plot(birth, literacy, color="r", marker="*", linestyle="none")
```

```
[<matplotlib.lines.Line2D at 0x1bbf0ea3070>]
```



Решение выданных ноутбуков

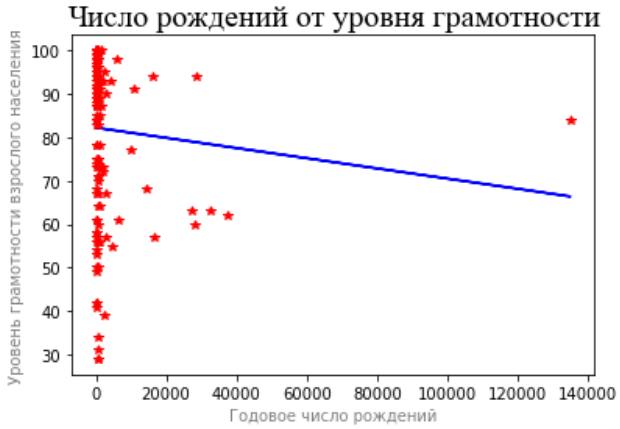
```
print(f"Среднее значение годового числа рождения: {np.mean(birth)}")
print(f"Среднее значение уровеня грамотности взрослого населения: {np.mean(liter
print(f"Среднее отклонение годового числа рождения: {np.std(birth)}")
print(f"Среднее отклонение уровеня грамотности взрослого населения: {np.std(lite
print(f"Дисперсия годового числа рождения: {np.var(birth)}")
print(f"Дисперсия уровеня грамотности взрослого населения: {np.var(literacy)}")
print(f"Коэффициент парной корреляции:\n {np.corrcoef(literacy, birth)}")
```

```
Среднее значение годового числа рождения: 2918.4039735099336
Среднее значение уровеня грамотности взрослого населения: 81.88079470198676
Среднее отклонение годового числа рождения: 12323.610323535137
Среднее отклонение уровеня грамотности взрослого населения: 18.465979871468555
Дисперсия годового числа рождения: 151871371.4063418
Дисперсия уровеня грамотности взрослого населения: 340.99241261348186
Коэффициент парной корреляции:
 [[ 1.         -0.07823878]
 [-0.07823878  1.        ]]
```

Решение выданных ноутбуков

```
plt.title('Число рождений от уровня грамотности', fontsize=20, fontname='Times N
plt.xlabel('Годовое число рождений', color='gray')
plt.ylabel('Уровень грамотности взрослого населения',color='gray')
A = np.vstack([birth, np.ones(len(birth))]).T
m, c = np.linalg.lstsq(A, literacy, rcond=None)[0]
plt.plot(birth, m*birth + c, 'b')
plt.plot(birth, literacy, color="r", marker="*", linestyle="none")
```

```
[<matplotlib.lines.Line2D at 0x1bbf1a6ca90>]
```



Решение выданных ноутбуков

```
# подключение модуля numpy под именем np
import numpy as np
```

```
# 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:

```
c = np.random.randint(1, 10, 5)
c
```

```
array([6, 8, 3, 5, 2])
```

Решение выданных ноутбуков

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.random.randint(1, 10, 5)
print(c)
print(f"c^3: {c**3}")
```

```
[4 5 6 8 1]
c^3: [ 64 125 216 512   1]
```

Решение выданных ноутбуков

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.

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

```
d / e = [5.         0.33333333 1.28571429 4.         0.55555556]
```

Решение выданных ноутбуков

Л — логика

К элементам массива можно применять логические операции.

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

```
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]
```

Решение выданных ноутбуков

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

```
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 5 8 6 1]
[4 8 2 3 8]
[ True  True False False  True]
[False False False  True False]
```

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

```
log3 = np.logical_or(d % 2 == 0, e > 2)
print(log3)
```

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

Решение выданных ноутбуков

```
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.

```
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]
```

Решение выданных ноутбуков

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:

```
# 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(['m

# 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']
```

Решение выданных ноутбуков

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
```

Решение выданных ноутбуков

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

```
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']
```

Решение выданных ноутбуков

Разработать программу, способную решать систему линейных алгебраических уравнений, содержащую n уравнений и n неизвестных. ¶

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1; \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n. \end{cases}$$

Входные данные:

`0≤i,j<n`

В первой строке вводится целое число n, количсетво уравнений и неизвестных.

Далее во входном потоке идет n строк по n вещественных чисел a_{ij}, коэффициенты уравнения, разделенные пробелом.

В последней строке вводятся n вещественных чисел b_i, свободные члены, разделенные пробелом.

Выходные данные:

В одной строке вывести корни данного уравнения, разделенные пробелом.

Пример входных данных:

```
3
2.0 -1.0 0.0
-1.0 1.0 4.0
1.0 2.0 3.0
0.0 13.0 14.0
```

Пример выходных данных:

```
1.0 2.0 3.0
```

Математическое задание

```
import numpy as np
```

```
# Считываем данные

n = int(input())
a = np.array([list(map(float, input().split())) for i in range(n)])
b = np.array(list(map(float, input().split())))
```

```
3
2.0 -1.0 0.0
-1.0 1.0 4.0
1.0 2.0 3.0
0.0 13.0 14.0
```

```
det = np.linalg.det(a)
ans = []
for j in range(n):
    a_sav = a.copy()
    a_sav[::, j] = b[::]
    det_x = np.linalg.det(a_sav)
    x = det_x / det
    ans.append(x)
print(ans)
```

```
[1.0, 1.9999999999999996, 3.0]
```

Решение математической задачи

ОТВЕТЫ НА КОНТРОЛЬНЫЕ 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?

Для начала заметим, что ndarray не то же самое, что класс array стандартной библиотеки Python, который используется только для одномерных массивов.

Дело, в том числе, в особенной структуре хранения данных. Обычные списки в Python хранят указатели на объекты, внутри которых есть указатели на их типы. Таким образом, чтобы прочитать что-то из списка нужно «скакать» по памяти. Numpy-массивы хранят указатель на кусок память, где лежат все данные, плюс метаинформацию об этих данных: типа, размерность и т.д. Тогда, при последовательном обращении к элементам массива эффективно используется кэш процессора, что позволяет снизить количество кэш-промахов, так как работает механизм предвыборки (prefetching).

3. Как осуществляется доступ к частям многомерного массива?

m[1, 0]

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

Получим вторую строчку матрицы.

m[1, :]

Извлечем третий столбец матрицы.

m[:, 2]

из второй строки нужно извлечь все элементы, начиная с третьего.

m[1, 2:]

можно извлечь только часть столбца матрицы.

m[0:2, 1]

часть матрицы

m[0:2, 1:3]

Определенные столбцы

cols = [0, 1, 3]

m[:, cols]

4. Как осуществляется расчет статистик по данным?

Для расчета той или иной статистики, соответствующую функцию можно вызвать как метод объекта, с которым вы работаете.

m.max()

Тот же результат можно получить, вызвав библиотечную функцию с соответствующим именем, передав ей в качестве аргумента массив.

np.max(m)

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

m.max(axis=1)

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?

Мы можем выбрать данные по определенному условию.

У нас будет массив, состоящий из значений True False.

Дальше мы можем вывести массив, где будут выведены только элементы индексы которых совпадает с индексом True.