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 №3 «Исследование методов работы с матрицами и векторами с помощью библиотеки 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

```
In [1]: import numpy as np
```

Рисунок 1 – Импорт модуля

```
In [2]: >>> v_hor_np = np.array([1, 2])
        >>> print(v_hor_np )
        [1 2]

In [3]: >>> v_hor_zeros_v1 = np.zeros((5,))
        >>> print(v_hor_zeros_v1 )
        [0. 0. 0. 0. 0.]

In [4]: >>> v_hor_zeros_v2 = np.zeros((1, 5))
        >>> print(v_hor_zeros_v2 )
        [[0. 0. 0. 0. 0.]]

In [5]: >>> v_hor_one_v1 = np.ones((5,))
        >>> print(v_hor_one_v1)
        >>> v_hor_one_v2 = np.ones((1, 5))
        >>> print(v_hor_one_v2)
        [1. 1. 1. 1. 1.]
        [[1. 1. 1. 1. 1.]]
```

Рисунок 2 – Пример работы с вектор строкой

```
In [6]: >>> v_vert_np = np.array([[1], [2]])
        >>> print(v_vert_np)
        [[1]
         [2]]

In [7]: >>> v_vert_zeros = np.zeros((5, 1))
        >>> print(v_vert_zeros)
        [[0.]
         [0.]
         [0.]
         [0.]
         [0.]]

In [8]: >>> v_vert_ones = np.ones((5, 1))
        >>> print(v_vert_ones)
        [[1.]
         [1.]
         [1.]
         [1.]
         [1.]]
```

Рисунок 3 – Пример работы с вектор-столбцом

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

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

In [10]: >>> m_sqr = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
         >>> m_sqr_arr = np.array(m_sqr)
         >>> print(m_sqr_arr)

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

In [11]: >>> m_sqr_mx = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
         >>> print(m_sqr_mx)

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

In [12]: >>> m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')
         >>> print(m_sqr_mx)

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

Рисунок 4 – Пример работы с квадратной матрицей

```
In [13]: >>> m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
         >>> m_diag_np = np.matrix(m_diag)
         >>> print(m_diag_np)

         [[1 0 0]
          [0 5 0]
          [0 0 9]]

In [14]: >>> m_sqr_mx = np.matrix('1 2 3; 4 5 6; 7 8 9')

In [15]: >>> diag = np.diag(m_sqr_mx)
         >>> print(diag)

         [1 5 9]

In [16]: >>> m_diag_np = np.diag(np.diag(m_sqr_mx))
         >>> print(m_diag_np)

         [[1 0 0]
          [0 5 0]
          [0 0 9]]
```

Рисунок 5 – Пример работы с диагональной матрицей

In [17]:
```
>>> m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
>>> m_e_np = np.matrix(m_e)
>>> print(m_e_np)
```

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

In [18]:
```
>>> m_eye = np.eye(3)
>>> print(m_eye)
```

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

In [19]:
```
>>> m_idnt = np.identity(3)
>>> print(m_idnt)
```

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

Рисунок 6 – Пример работы с единичной матрицей

```
>>> m_zeros = np.zeros((3, 3))
>>> print(m_zeros)
```

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

Рисунок 7 – Пример работы с нулевой матрицей

```
>>> m_mx = np.matrix('1 2 3; 4 5 6')
>>> print(m_mx)
```

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

```
>>> m_var = np.zeros((2, 5))
>>> print(m_var)
```

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

Рисунок 8 – Задание матрицы в общем виде

In [23]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> print(A)
```

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

In [24]:
```
>>> A_t = A.transpose()
>>> print(A_t)
```

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

In [25]:
```
print(A.T)
```

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

In [26]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> print(A)
```

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

In [27]:
```
>>> R = (A.T).T
>>> print(R)
```

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

In [28]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('7 8 9; 0 7 5')
>>> L = (A + B).T
>>> R = A.T + B.T
>>> print(L)
```

```
[[ 8  4]
 [10 12]
 [12 11]]
```

In [29]:
```
>>> print(R)
```

```
[[ 8  4]
 [10 12]
 [12 11]]
```

In [30]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = (A.dot(B)).T
>>> R = (B.T).dot(A.T)
>>> print(L)
>>> print(R)
```

```
[[19 43]
 [22 50]]
[[19 43]
 [22 50]]
```

In [31]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> k = 3
>>> L = (k * A).T
>>> R = k * (A.T)
>>> print(L)
>>> print(R)
```

```
[[ 3 12]
 [ 6 15]
 [ 9 18]]
[[ 3 12]
 [ 6 15]
 [ 9 18]]
```

In [32]:
```
>>> A = np.matrix('1 2; 3 4')
>>> A_det = np.linalg.det(A)
>>> A_T_det = np.linalg.det(A.T)
>>> print(format(A_det, '.9g'))
>>> print(format(A_T_det, '.9g'))
```

```
-2
-2
```

Рисунок 9 – Транспонирование матрицы

In [33]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> C = 3 * A
>>> print(C)
```

```
[[ 3  6  9]
 [12 15 18]]
```

In [34]:
```
>>> A = np.matrix('1 2; 3 4')
>>> L = 1 * A
>>> R = A
>>> print(L)
>>> print(R)
```

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

In [35]:
```
>>> A = np.matrix('1 2; 3 4')
>>> Z = np.matrix('0 0; 0 0')
>>> L = 0 * A
>>> R = Z
>>> print(L)
>>> print(R)
```

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

In [36]:
```
>>> A = np.matrix('1 2; 3 4')
>>> p = 2
>>> q = 3
>>> L = (p + q) * A
>>> R = p * A + q * A
>>> print(L)
>>> print(R)
```

```
[[ 5 10]
 [15 20]]
[[ 5 10]
 [15 20]]
```

In [37]:
```
>>> A = np.matrix('1 2; 3 4')
>>> p = 2
>>> q = 3
>>> L = (p * q) * A
>>> R = p * (q * A)
>>> print(L)
>>> print(R)
```

```
[[ 6 12]
 [18 24]]
[[ 6 12]
 [18 24]]
```

In [38]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> k = 3
>>> L = k * (A + B)
>>> R = k * A + k * B
>>> print(L)
>>> print(R)
```

```
[[18 24]
 [30 36]]
[[18 24]
 [30 36]]
```

Рисунок 10 – Умножение матрицы на число

In [39]:
```
>>> A = np.matrix('1 6 3; 8 2 7')
>>> B = np.matrix('8 1 5; 6 9 12')
>>> C = A + B
>>> print(C)
```

```
[[ 9  7  8]
 [14 11 19]]
```

In [40]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> L = A + B
>>> R = B + A
>>> print(L)
>>> print(R)
```

```
[[ 6  8]
 [10 12]]
[[ 6  8]
 [10 12]]
```

In [41]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> C = np.matrix('1 7; 9 3')
>>> L = A + (B + C)
>>> R = (A + B) + C
>>> print(L)
>>> print(R)
```

```
[[ 7 15]
 [19 15]]
[[ 7 15]
 [19 15]]
```

In [42]:
```
>>> A = np.matrix('1 2; 3 4')
>>> Z = np.matrix('0 0; 0 0')
>>> L = A + (-1)*A
>>> print(L)
>>> print(Z)
```

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

Рисунок 11 – Сложение матриц

In [43]:
```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('7 8; 9 1; 2 3')
>>> C = A.dot(B)
>>> print(C)
```

```
[[31 19]
 [85 55]]
```

In [44]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> C = np.matrix('2 4; 7 8')
>>> L = A.dot(B.dot(C))
>>> R = (A.dot(B)).dot(C)
>>> print(L)
>>> print(R)
```

```
[[192 252]
 [436 572]]
[[192 252]
 [436 572]]
```

In [45]:
```
>>> A = np.matrix('1 2; 3 4')
>>> B = np.matrix('5 6; 7 8')
>>> C = np.matrix('2 4; 7 8')
>>> L = A.dot(B + C)
>>> R = A.dot(B) + A.dot(C)
>>> print(L)
>>> print(R)
```

```
[[35 42]
 [77 94]]
[[35 42]
 [77 94]]
```

```
In [46]: >>> A = np.matrix('1 2; 3 4')
         >>> B = np.matrix('5 6; 7 8')
         >>> L = A.dot(B)
         >>> R = B.dot(A)
         >>> print(L)
         >>> print(R)

         [[19 22]
          [43 50]]
         [[23 34]
          [31 46]]

In [47]: >>> A = np.matrix('1 2; 3 4')
         >>> E = np.matrix('1 0; 0 1')
         >>> L = E.dot(A)
         >>> R = A.dot(E)
         >>> print(L)
         >>> print(R)
         >>> print(A)

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

In [48]: >>> A = np.matrix('1 2; 3 4')
         >>> Z = np.matrix('0 0; 0 0')
         >>> L = Z.dot(A)
         >>> R = A.dot(Z)
         >>> print(L)
         >>> print(R)
         >>> print(Z)

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

Рисунок 12 – Умножение матриц

```
In [49]: >>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         >>> print(A)

         [[-4 -1  2]
          [10  4 -1]
          [ 8  3  1]]

In [50]: >>> np.linalg.det(A)

Out[50]: -14.000000000000009

In [51]: >>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
         >>> print(A)
         >>> print(A.T)

         [[-4 -1  2]
          [10  4 -1]
          [ 8  3  1]]
         [[-4 10  8]
          [-1  4  3]
          [ 2 -1  1]]
```

In [52]:
```
>>> det_A = round(np.linalg.det(A), 3)
>>> det_A_t = round(np.linalg.det(A.T), 3)
>>> print(det_A)
>>> print(det_A_t)
```

```
-14.0
-14.0
```

In [53]:
```
>>> A = np.matrix('-4 -1 2; 0 0 0; 8 3 1')
>>> print(A)
>>> np.linalg.det(A)
```

```
[[-4 -1  2]
 [ 0  0  0]
 [ 8  3  1]]
```

Out[53]: 0.0

In [54]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
>>> B = np.matrix('10 4 -1; -4 -1 2; 8 3 1')
>>> print(B)
>>> round(np.linalg.det(A), 3)
>>> round(np.linalg.det(B), 3)
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[10  4 -1]
 [-4 -1  2]
 [ 8  3  1]]
```

Out[54]: 14.0

In [55]:
```
>>> A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
>>> print(A)
>>> np.linalg.det(A)
```

```
[[-4 -1  2]
 [-4 -1  2]
 [ 8  3  1]]
```

Out[55]: 0.0

In [56]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
>>> k = 2
>>> B = A.copy()
>>> B[2, :] = k * B[2, :]
>>> print(B)
>>> det_A = round(np.linalg.det(A), 3)
>>> det_B = round(np.linalg.det(B), 3)
>>> det_A * k
>>> det_B
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 -1  2]
 [10  4 -1]
 [16  6  2]]
```

Out[56]: -28.0

In [57]:
```
>>> A = np.matrix('-4 -1 2; -4 -1 2; 8 3 1')
>>> B = np.matrix('-4 -1 2; 8 3 2; 8 3 1')
>>> C = A.copy()
>>> C[1, :] += B[1, :]
>>> print(C)
>>> print(A)
>>> print(B)
>>> round(np.linalg.det(C), 3)
>>> round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3)
```

```
[[-4 -1  2]
 [ 4  2  4]
 [ 8  3  1]]
[[-4 -1  2]
 [-4 -1  2]
 [ 8  3  1]]
[[-4 -1  2]
 [ 8  3  2]
 [ 8  3  1]]
```

Out[57]: 4.0

In [58]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> k = 2
>>> B = A.copy()
>>> B[1, :] = B[1, :] + k * B[0, :]
>>> print(A)
>>> print(B)
>>> round(np.linalg.det(A), 3)
>>> round(np.linalg.det(B), 3)
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 -1  2]
 [ 2  2  3]
 [ 8  3  1]]
```

Out[58]: -14.0

In [59]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
>>> k = 2
>>> A[1, :] = A[0, :] + k * A[2, :]
>>> round(np.linalg.det(A), 3)
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
```

Out[59]: 0.0

In [60]:
```
>>> A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
>>> print(A)
>>> k = 2
>>> A[1, :] = k * A[0, :]
>>> print(A)
>>> round(np.linalg.det(A), 3)
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 -1  2]
 [-8 -2  4]
 [ 8  3  1]]
```

Out[60]: 0.0

Рисунок 13 – Определитель матрицы

In [61]:
```
>>> A = np.matrix('1 -3; 2 5')
>>> A_inv = np.linalg.inv(A)
>>> print(A_inv)
```

```
[[ 0.45454545  0.27272727]
 [-0.18181818  0.09090909]]
```

In [62]:
```
>>> A = np.matrix('1. -3.; 2. 5.')
>>> A_inv = np.linalg.inv(A)
>>> A_inv_inv = np.linalg.inv(A_inv)
>>> print(A)
>>> print(A_inv_inv)
```

```
[[ 1. -3.]
 [ 2.  5.]]
[[ 1. -3.]
 [ 2.  5.]]
```

In [63]:
```
>>> A = np.matrix('1. -3.; 2. 5.')
>>> L = np.linalg.inv(A.T)
>>> R = (np.linalg.inv(A)).T
>>> print(L)
>>> print(R)
```

```
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
[[ 0.45454545 -0.18181818]
 [ 0.27272727  0.09090909]]
```

In [64]:
```
>>> A = np.matrix('1. -3.; 2. 5.')
>>> B = np.matrix('7. 6.; 1. 8.')
>>> L = np.linalg.inv(A.dot(B))
>>> R = np.linalg.inv(B).dot(np.linalg.inv(A))
>>> print(L)
>>> print(R)
```

```
[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
[[ 0.09454545  0.03272727]
 [-0.03454545  0.00727273]]
```

Рисунок 14 – Обратная матрица

In [65]:
```
>>> m_eye = np.eye(4)
>>> print(m_eye)
>>> rank = np.linalg.matrix_rank(m_eye)
>>> print(rank)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
4
```

In [66]:
```
>>> m_eye[3][3] = 0
>>> print(m_eye)
>>> rank = np.linalg.matrix_rank(m_eye)
>>> print(rank)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
3
```

Рисунок 15 – Ранг матрицы

Транспонирование

Транспонированная матрица равна исходной матрице

In [4]:
```
>>> A = np.random.randint(-20, 70, (2, 2))
>>> print(A)
>>> R = (A.T).T
>>> print(R)
```

```
[[ 61  21]
 [ 18 -13]]
[[ 61  21]
 [ 18 -13]]
```

Транспоринование суммы матриц равно сумме транспонированных матриц

In [5]:
```
A = np.matrix('8 7 5; 5 4 3')
B = np.matrix('1 2 3; 7 7 7')
L = (A + B).T
R = A.T + B.T
print("Транспонирование суммы:\n",L)
print("Сумма транспонированных матриц:\n",R)
```

```
Транспонирование суммы:
 [[ 9 12]
 [ 9 11]
 [ 8 10]]
Сумма транспонированных матриц:
 [[ 9 12]
 [ 9 11]
 [ 8 10]]
```

Транспонирование произведения матриц равно произведению транспонированных матриц расставленных в обратном порядке

n [6]:
```
A = np.matrix('7 5; 7 4')
B = np.matrix('7 6; 7 7')
L = (A.dot(B)).T
R = (B.T).dot(A.T)
print("Транспонирование произведения матриц:\n",L)
print("Произведение транспонированных матриц расставленных в обратном порядке:\n",R)
```

```
Транспонирование произведения матриц:
 [[84 77]
 [77 70]]
Произведение транспонированных матриц расставленных в обратном порядке:
 [[84 77]
 [77 70]]
```

Транспонирование произведения матрицы на число равно произведению этого числа на транспонированную матрицу

In [7]:
```
A = np.matrix('1 2 3; 4 5 6')
k = 70
L = (k * A).T
R = k * (A.T)
print("Транспонирование произведения матрицы на число:\n",L)
print("Произведение этого числа на транспонированную матрицу:\n",R)
```

```
Транспонирование произведения матрицы на число:
 [[ 70 280]
 [140 350]
 [210 420]]
Произведение этого числа на транспонированную матрицу:
 [[ 70 280]
 [140 350]
 [210 420]]
```

Определители исходной и транспонированной матрицы совпадают

In [8]:
```
A = np.matrix('55 100; 1 5')
A_det = np.linalg.det(A)
A_T_det = np.linalg.det(A.T)
print("Определитель исходной матрицы:",format(A_det, '.9g'))
-2
print("Определитель транспонированной матрицы:",format(A_T_det, '.9g'))
```

```
Определитель исходной матрицы: 175
Определитель транспонированной матрицы: 175
```

Рисунок 16 – Свойства транспонирования матриц

Умножение

Произведение единицы и любой заданной матрицы равно заданной матрице

In [9]:
```
A = np.matrix('1 2; 3 4')
L = 1 * A
print("Матрица умноженная на единицу:\n", L)
```

```
Матрица умноженная на единицу:
 [[1 2]
 [3 4]]
```

Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы

In [10]:
```
A = np.matrix('1 2; 3 4')
L = 0 * A
print("Произведение матрицы на ноль:\n", L)
```

```
Произведение матрицы на ноль:
 [[0 0]
 [0 0]]
```

Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел

In [11]:
```
A = np.matrix('1 2; 1 2')
p = 70
q = 5
L = (p + q) * A
R = p * A + q * A
print("Произведение матрицы на сумму чисел:\n", L)
print("Сумма произведений матрицы на каждое из этих чисел:\n", R)
```

```
Произведение матрицы на сумму чисел:
 [[ 75 150]
 [ 75 150]]
Сумма произведений матрицы на каждое из этих чисел:
 [[ 75 150]
 [ 75 150]]
```

Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному на первое число

In [12]:
```
A = np.matrix('1 2; 1 2')
p = 5
q = 100
L = (p * q) * A
R = p * (q * A)
print("Произведение матрицы на произведение двух чисел:\n", L)
print("Произведение второго числа и заданной матрицы, умноженному на первое число:\n", R)
```

```
Произведение матрицы на произведение двух чисел:
 [[ 500 1000]
 [ 500 1000]]
Произведение второго числа и заданной матрицы, умноженному на первое число:
 [[ 500 1000]
 [ 500 1000]]
```

Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число

In [13]:
```
A = np.matrix('1 2; 1 2')
B = np.matrix('3 4; 5 5')
k = 100
L = k * (A + B)
R = k * A + k * B
print("Произведение суммы матриц на число:\n", L)
print("Сумма произведений этих матриц на заданное число:\n", R)
```

```
Произведение суммы матриц на число:
 [[400 600]
 [600 700]]
Сумма произведений этих матриц на заданное число:
 [[400 600]
 [600 700]]
```

Рисунок 17 – Свойства умножения матриц

Сложение

Коммутативность сложения. От перестановки матриц их сумма не изменяется

In [14]:
```
A = np.matrix('1000 2000; 300 400')
B = np.matrix('500 600; 7000 8000')
L = A + B
R = B + A
print("Сумма A + B:\n", L)
print("Сумма B + A:\n", R)
```

```
Сумма A + B:
 [[1500 2600]
 [7300 8400]]
Сумма B + A:
 [[1500 2600]
 [7300 8400]]
```

Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котором эта операция будет выполняться

In [15]:
```
A = np.matrix('1 1; 1 1')
B = np.matrix('2 2; 2 2')
C = np.matrix('3 3; 3 3')
L = A + (B + C)
R = (A + B) + C
print("Сумма A + (B + C):\n", L)
print("Сумма (A + B) + C:\n", R)
```

```
Сумма A + (B + C):
 [[6 6]
 [6 6]]
Сумма (A + B) + C:
 [[6 6]
 [6 6]]
```

Для любой матрицы существует противоположная ей, такая, что их сумма является нулевой матрицей

In [16]:
```
A = np.matrix('20000000 30000000; 40000000 50000000')
L = A + (-1)*A
print("Нулевая матрица:\n", L)
```

```
Нулевая матрица:
 [[0 0]
 [0 0]]
```

Рисунок 18 – Свойства сложения матриц

Определитель матрицы

Определитель матрицы остается неизменным при ее транспонировании

In [22]:
```
A = np.matrix('1 10 2; 10 2 -1; 5 8 11')
R = A.T
det_A = round(np.linalg.det(A), 3)
det_R = round(np.linalg.det(R), 3)
print("Опеределитель матрицы:", det_A)
print("Определитель транспонированной матрицы:", det_R)
```

```
Опеределитель матрицы: -980.0
Определитель транспонированной матрицы: -980.0
```

Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю

In [23]:
```
A = np.matrix('1 10 2; 0 0 0; 5 8 11')
print("Опеределитель матрицы:", np.linalg.det(A))
```

```
Опеределитель матрицы: 0.0
```

При перестановке строк матрицы знак ее определителя меняется на противоположный

In [24]:
```
A = np.matrix('1 10 2; 10 2 -1; 5 8 11')
B = np.matrix('10 2 -1; 1 10 2; 5 8 11')
R = round(np.linalg.det(A), 3)
L = round(np.linalg.det(B), 3)
print("Определитель матрицы A:", R)
print("Определитель матрицы B:", L)
```

```
Определитель матрицы A: -980.0
Определитель матрицы B: 980.0
```

Если у матрицы есть две одинаковые строки, то ее определитель равен нулю

In [25]:
```
A = np.matrix('1 10 2; 1 10 2; 5 8 11')
R = round(np.linalg.det(A), 3)
print("Определитель матрицы A:", R)
```

```
Определитель матрицы A: 0.0
```

Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число

In [26]:
```
A = np.matrix('1 10 2; 1 11 2; 5 8 11')
k = 100
B[2, :] = k * B[2, :]
det_A = round(np.linalg.det(A), 3)
det_B = round(np.linalg.det(B), 3)
print("Опеределитель матрицы:", det_A)
print("Определитель матрицы умноженной на число:", det_B)
```

```
Опеределитель матрицы: 1.0
Определитель матрицы умноженной на число: 98000.0
```

Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен сумме определителей двух соответствующих матриц

In [27]:
```
A = np.matrix('-1 -1 3; -1 -1 3; 2 2 3')
B = np.matrix('-1 -1 3; 2 2 7; 2 2 3')
C = A.copy()
C[1, :] += B[1, :]
R = round(np.linalg.det(C), 3)
L = round(np.linalg.det(A), 3) + round(np.linalg.det(B), 3)
print("Определитель матрицы C:", R)
print("Определитель суммы двух матриц:", L)
```

```
Определитель матрицы C: 0.0
Определитель суммы двух матриц: 0.0
```

Если к элементам одной строки прибавить элементы другой строки, умноженные на одно и тоже число, то определитель матрицы не изменится

In [28]:
```
A = np.matrix('-1 -1 3; -1 0 3; 2 2 3')
k = 70
B = A.copy()
B[1, :] = B[1, :] + k * B[0, :]
R = round(np.linalg.det(A), 3)
L = round(np.linalg.det(B), 3)
print("Определитель матрицы A:", R)
print("Определитель матрицы B:", L)
```

```
Определитель матрицы A: -9.0
Определитель матрицы B: -9.0
```

Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы равен нулю

In [29]:
```
A = np.matrix('-3 -1 5; -1 0 3; 2 2 3')
k = 70
A[1, :] = A[0, :] + k * A[2, :]
R = round(np.linalg.det(A), 3)
print("Определитель матрицы A:", R)
```

```
Определитель матрицы A: 0.0
```

Если матрица содержит пропорциональные строки, то ее определитель равен нулю

In [30]:
```
A = np.matrix('-3 -1 5; -1 0 3; 2 2 3')
k = 70
A[1, :] = k * A[0, :]
R = round(np.linalg.det(A), 3)
print("Определитель матрицы A:", R)
```

```
Определитель матрицы A: 0.0
```

Рисунок 19 – Определитель матрицы

Обратная матрица

Обратная матрица обратной матрицы есть исходная матрица

In [31]:
```
A = np.matrix('8. -7.; 5. 4.')
A_inv = np.linalg.inv(A)
A_inv_inv = np.linalg.inv(A_inv)
print("Определитель исходной матрицы:\n", A_inv)
print("Определитель обратной обратной матрицы A:\n", A_inv_inv)
```

```
Определитель исходной матрицы:
 [[ 0.05970149  0.10447761]
 [-0.07462687  0.11940299]]
Определитель обратной обратной матрицы A:
 [[ 8. -7.]
 [ 5.  4.]]
```

Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы

In [32]:
```
A = np.matrix('8. -7.; 5. 4.')
L = np.linalg.inv(A.T)
R = (np.linalg.inv(A)).T
print("Обратная матрица транспонированной матрицы:\n",L)
print("Транспонированная матрица от обратной матрицы:\n", R)
```

```
Обратная матрица транспонированной матрицы:
 [[ 0.05970149 -0.07462687]
 [ 0.10447761  0.11940299]]
Транспонированная матрица от обратной матрицы:
 [[ 0.05970149 -0.07462687]
 [ 0.10447761  0.11940299]]
```

Обратная матрица произведения матриц равна произведению обратных матриц

In [33]:
```
A = np.matrix('8. -7.; 5. 4.')
B = np.matrix('15. -7.; 22. 2.')
L = np.linalg.inv(A.dot(B))
R = np.linalg.inv(B).dot(np.linalg.inv(A))
print("Обратная матрица произведения матриц:\n",L)
print("Произведение обратных матриц:\n", R)
```

```
Обратная матрица произведения матриц:
 [[-0.00219014  0.00567813]
 [-0.01322193 -0.00275795]]
Произведение обратных матриц:
 [[-0.00219014  0.00567813]
 [-0.01322193 -0.00275795]]
```

Рисунок 20 – Обратная матрица

Матричный метод

In [8]:
```
import numpy as np
A = np.matrix('7 1 2; 5 4 2; 5 5 4')
B = np.matrix('5; 1; 4')
if round(np.linalg.det(A), 3) != 0:
    A_inv = np.linalg.inv(A)
    R = A_inv.dot(B)
    i = 1
    for elem in R:
        print(f"x{i} = {elem}")
        i += 1
else:
    print("определитель равен 0, систему решить нельзя")
```

```
x1 = [[0.28571429]]
x2 = [[-1.14285714]]
x3 = [[2.07142857]]
```

Метод Крамера

In [9]:
```
import numpy as np
```

In [12]:
```
A = np.matrix('7 1 0; 5 4 2; 7 5 3')
B = np.matrix('5; 1; 4')
if round(np.linalg.det(A), 3) != 0:
    K = A.copy()
    K[:, 0] = B[:, 0]
    x1 = round(np.linalg.det(C), 3) / round(np.linalg.det(A), 3)
    F = A.copy()
    F[:, 1] = B[:, 0]
    x2 = round(np.linalg.det(D), 3) / round(np.linalg.det(A), 3)
    J = A.copy()
    J[:, 2] = B[:, 0]
    x3 = round(np.linalg.det(F), 3) / round(np.linalg.det(A), 3)

    print("x1:", x1, "x2:", x2, "x3:", x3)
```

```
x1: 1.3846153846153846 x2: -4.6923076923076925 x3: -3.076923076923077
```

Рисунок 21 – Решение уравнений

ВОПРОСЫ

1. Приведите основные виды матриц и векторов. Опишите способы их создания в языке Python.

   Способ задания вектора-строки:

```
>>> v_hor_np = np.array([1, 2])
>>> print(v_hor_np )
[1 2]
```

   Способ создания нулевого вектора-строки:

```
>>> v_hor_zeros_v1 = np.zeros((5,))
>>> print(v_hor_zeros_v1 )
[0. 0. 0. 0. 0.]
```

   Способ создания единичного вектора-строки:

```
>>> v_hor_one_v1 = np.ones((5,))
>>> print(v_hor_one_v1)
[1. 1. 1. 1. 1.]
```

   Способ создания вектора-столбца:

```
>>> v_vert_np = np.array([[1], [2]])
>>> print(v_vert_np)
[[1]
[2]]
```

   Способ создания нулевого вектора-столбца:

```
>>> v_vert_zeros = np.zeros((5, 1))
>>> print(v_vert_zeros)
[[0.]
[0.]
[0.]
[0.]
[0.]]
```

   Способ создания единичного вектора-столбца:

```
>>> v_vert_ones = np.ones((5, 1))
>>> print(v_vert_ones)
[[1.]
[1.]
[1.]
[1.]
[1.]]
```

   Способ создания квадратной матрицы:

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

   Способ создания диагональной матрицы:

```
>>> m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
>>> m_diag_np = np.matrix(m_diag)
>>> print(m_diag_np)
[[1 0 0]
[0 5 0]
[0 0 9]]
```

Способ создания единичной матрицы:

```
>>> m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
>>> m_e_np = np.matrix(m_e)
>>> print(m_e_np)
[[1 0 0]
[0 1 0]
[0 0 1]]
```

Способ создания нулевой матрицы:

```
>>> m_zeros = np.zeros((3, 3))
>>> print(m_zeros)
[[ 0. 0. 0.]
[ 0. 0. 0.]
[ 0. 0. 0.]]
```

2. Как выполняется транспонирование матриц?

```
>>> A_t = A.transpose()
>>> print(A_t)
[[1 4]
[2 5]
[3 6]]
```

```
>>> print(A.T)
[[1 4]
[2 5]
[3 6]]
```

3. Приведите свойства операции транспонирования матриц.

Свойство 1. Дважды транспонированная матрица равна исходной матрице.

Свойство 2. Транспонирование суммы матриц равно сумме транспонированных матриц.

Свойство 3. Транспонирование произведения матриц равно произведению транспонированных матриц, расставленных в обратном порядке.

Свойство 4. Транспонирование произведения матрицы на число равно произведению этого числа на транспонированную матрицу.

4. Какие имеются средства в библиотеке NumPy для выполнения транспонирования матриц?

В библиотеки NumPy для транспонирования двумерных матриц используется метод transpose().

5. Какие существуют основные действия над матрицами?

Умножение матрицы на число

Сложение матриц

Умножение матриц

6. Как осуществляется умножение матрицы на число?

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> C = 3 * A
>>> print(C)
[[ 3  6 9]
[12 15 18]]
```

7. Какие свойства операции умножения матрицы на число?

Свойство 1. Произведение единицы и любой заданной матрицы равнозаданной матрице.
Свойство 2. Произведение нуля и любой матрицы равно нулевой матрице, размерность которой равна исходной матрицы.
Свойство 3. Произведение матрицы на сумму чисел равно сумме произведений матрицы на каждое из этих чисел.
Свойство 4. Произведение матрицы на произведение двух чисел равно произведению второго числа и заданной матрицы, умноженному на первое число.
Свойство 5. Произведение суммы матриц на число равно сумме произведений этих матриц на заданное число.

8. Как осуществляется операции сложения и вычитания матриц?

Сложение матриц:

```
>>> A = np.matrix('1 6 3; 8 2 7')
>>> B = np.matrix('8 1 5; 6 9 12')
>>> C = A + B
>>> print(C)
[[ 9  7 8]
[14 11 19]]
```

Вычитание:

```
>>> import numpy as np
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('1 2 3; 4 5 6')
>>> C = A - B
>>> print(C)
[[0 0 0]
 [0 0 0]]
```

9. Каковы свойства операций сложения и вычитания матриц?

Свойство 1. Коммутативность сложения. От перестановки матриц их сумма не изменяется.
Свойство 2. Ассоциативность сложения. Результат сложения трех и более матриц не зависит от порядка, в котором эта операция будет выполняться.
Свойство 3. Для любой матрицы существует противоположная ей, такая, что их сумма является нулевой матрицей.

10. Какие имеются средства в библиотеке NumPy для выполнения операций сложения и вычитания матриц?

Для сложения и вычитания используется + и -.

11. Как осуществляется операция умножения матриц?

```
>>> A = np.matrix('1 2 3; 4 5 6')
>>> B = np.matrix('7 8; 9 1; 2 3')
>>> C = A.dot(B)
>>> print(C)
[[31 19]
[85 55]]
```

12. Каковы свойства операции умножения матриц?

Свойство 1. Ассоциативность умножения. Результат умножения матриц не зависит от порядка, в котором будет выполняться эта операция.

Свойство 2. Дистрибутивность умножения. Произведение матрицы на сумму матриц равно сумме произведений матриц.

Свойство 3. Умножение матриц в общем виде не коммутативно. Это означает, что для матриц не выполняется правило независимости произведения от перестановки множителей.

Свойство 4. Произведение заданной матрицы на единичную равно исходной матрице.

Свойство 5. Произведение заданной матрицы на нулевую матрицу равно нулевой матрице.

13. Какие имеются средства в библиотеке NumPy для выполнения операции умножения матриц?

Функция dot().

14. Что такое определитель матрицы? Каковы свойства определителя матрицы?

Определитель матрицы размера (n-го порядка) является одной из ее численных характеристик. Определитель матрицы A обозначается как |A| или det(A), его также называют детерминантом.

Свойство 1. Определитель матрицы остается неизменным при ее транспонировании.

Свойство 2. Если у матрицы есть строка или столбец, состоящие из нулей, то определитель такой матрицы равен нулю.

Свойство 3. При перестановке строк матрицы знак ее определителя меняется на противоположный.

Свойство 4. Если у матрицы есть две одинаковые строки, то ее определитель равен нулю.

Свойство 5. Если все элементы строки или столбца матрицы умножить на какое-то число, то и определитель будет умножен на это число.

Свойство 6. Если все элементы строки или столбца можно представить как сумму двух слагаемых, то определитель такой матрицы равен сумме определителей двух соответствующих матриц.

Свойство 7. Если к элементам одной строки прибавить элементы другой строки, умноженные на одно и тоже число, то определитель матрицы не изменится.

Свойство 8. Если строка или столбец матрицы является линейной комбинацией других строк (столбцов), то определитель такой матрицы равен нулю

15. Какие имеются средства в библиотеке NumPy для нахождения значения определителя матрицы?

Функция linalg.det().

16. Что такое обратная матрица? Какой алгоритм нахождения обратной матрицы?

Обратной матрицей матрицы называют матрицу, удовлетворяющую следующему равенству: A * A^-1 = E

Для того, чтобы у квадратной матрицы A была обратная матрица необходимо и достаточно чтобы определитель $|A|$ был не равен нулю. Введем понятие **союзной матрицы**. Союзная матрица A строится на базе исходной A путем замены всех элементов матрицы A на их алгебраические дополнения.

Исходная матрица:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Союзная ей матрица A:

$$A^* = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \dots & \dots & \dots & \dots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}.$$

Транспонируя матрицу A, мы получим так называемую присоединенную матрицу A^T:

$$A^{*T} = \begin{pmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{pmatrix}.$$

Теперь, зная как вычислять определитель и присоединенную матрицу, мы можем определить матрицу A^{-1}, обратную матрице A:

$$A^{-1} = \frac{1}{det(A)} \times A^{*T}.$$

17. Каковы свойства обратной матрицы?

Свойство 1. Обратная матрица обратной матрицы есть исходная матрица.

Свойство 2. Обратная матрица транспонированной матрицы равна транспонированной матрице от обратной матрицы.

Свойство 3. Обратная матрица произведения матриц равна произведению обратных матриц.

18. Какие имеются средства в библиотеке NumPy для нахождения обратной матрицы?

```
>>> A = np.matrix('1 -3; 2 5')
>>> A_inv = np.linalg.inv(A)
>>> print(A_inv)
[[ 0.45454545 0.27272727]
 [-0.18181818 0.09090909]]
```