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 № 3.3 Исследование методов работы с матрицами и векторами с помощью библиотеки 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:

Ввод [1]:
```
import numpy as np
```

Вектор:

Вектор-строка:

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

```
[1 2]
```

Ввод [3]:
```
v_hor_zeros_v1 = np.zeros((5,))
print(v_hor_zeros_v1 )
```

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

Ввод [4]:
```
v_hor_zeros_v2 = np.zeros((1, 5))
print(v_hor_zeros_v2 )
```

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

Ввод [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.]]
```

Вектор-столбец:

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

```
[[1]
 [2]]
```

Ввод [7]:
```
v_vert_zeros = np.zeros((5, 1))
print(v_vert_zeros)
```

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

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

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

Квадратная матрица:

Ввод [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]]
```

Ввод [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]]
```

Ввод [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]]
```

Ввод [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]]
```

Диагональная матрица:

Ввод [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]]
```

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

Ввод [15]:
```
diag = np.diag(m_sqr_mx)
print(diag)
```

```
[1 5 9]
```

Ввод [16]:
```
m_diag_np = np.diag(np.diag(m_sqr_mx))
print(m_diag_np)
```

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

Единичная матрица:

Ввод [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]]
```

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

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

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

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

Нулевая матрица:

Ввод [20]:
```
m_zeros = np.zeros((3, 3))
print(m_zeros)
```

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

Задание матрицы в общем виде:

Ввод [21]:
```
m_mx = np.matrix('1 2 3; 4 5 6')
print(m_mx)
```

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

Ввод [22]:
```
m_var = np.zeros((2, 5))
print(m_var)
```

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

Транспонирование матрицы:

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

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

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

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

Ввод [25]:
```
print(A.T)
```

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

Ввод [26]:
```
A = np.matrix('1 2 3; 4 5 6')
print(A)
R = (A.T).T
print(R)
```

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

Ввод [27]:
```
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)
print(R)
```

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

Ввод [28]:
```
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]]
```

Ввод [29]:
```
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]]
```

Ввод [30]:
```
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
```

Действия над матрицами:

Умножение матрицы на число:

Ввод [31]:
```
A = np.matrix('1 2 3; 4 5 6')
C = 3 * A
print(C)
```

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

Ввод [32]:
```
A = np.matrix('1 2; 3 4')
L = 1 * A
R = A
print(L)
print(R)
```

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

Ввод [33]:
```
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]]
```

Ввод [34]:
```
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]]
```

Ввод [35]:
```
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]]
```

Ввод [36]:
```
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]]
```

Сложение матриц:

Ввод [37]:
```
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]]
```

Ввод [38]:
```
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]]
```

Ввод [39]:
```
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]]
```

Ввод [40]:
```
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]]
```

Умножение матриц:

Ввод [41]:
```
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]]
```

Ввод [42]:
```
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]]
```

Ввод [43]:
```
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]]
```

Ввод [44]:
```
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]]
```

Ввод [45]:
```
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]]
```

Ввод [46]:
```
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]]
```

Определитель матрицы:

Ввод [47]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
```

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

Ввод [48]:
```
np.linalg.det(A)
```

Out[48]:
```
-14.000000000000009
```

Ввод [49]:
```
A = np.matrix('-4 -1 2; 10 4 -1; 8 3 1')
print(A)
print(A.T)
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)
```

```
[[-4 -1  2]
 [10  4 -1]
 [ 8  3  1]]
[[-4 10  8]
 [-1  4  3]
 [ 2 -1  1]]
-14.0
-14.0
```

Ввод [50]:
```
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[50]:
```
0.0
```

Ввод [51]:
```
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[51]: 14.0

Ввод [52]:
```
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[52]: 0.0

Ввод [53]:
```
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[53]: -28.0

Ввод [54]:
```
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[54]: 4.0

Ввод [55]:
```
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[55]: -14.0

Ввод [56]:
```
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[56]:
```
0.0
```

Ввод [57]:
```
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[57]:
```
0.0
```

Обратная матрица:

Ввод [58]:
```
A = np.matrix('1 -3; 2 5')
A_inv = np.linalg.inv(A)
print(A_inv)
```

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

Ввод [59]:
```
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.]]
```

Ввод [60]:
```
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]]
```

Ввод [61]:
```
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]]
```

Ранг матрицы:

Ввод [62]:
```
m_eye = np.eye(4)
print(m_eye)
```

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

Ввод [63]:
```
rank = np.linalg.matrix_rank(m_eye)
print(rank)
```

```
4
```

Ввод [64]:
```
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
```

Первая индивидуальная задача

Примеры со своими значениями:

Ввод [1]:
```
import numpy as np
```

Вектор:

Вектор-строка:

Ввод [2]:
```
v_hor_np = np.array([2, 3])
print(v_hor_np )
```

```
[2 3]
```

Ввод [3]:
```
v_hor_zeros_v1 = np.zeros((6,))
print(v_hor_zeros_v1 )
```

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

Ввод [4]:
```
v_hor_zeros_v2 = np.zeros((1, 6))
print(v_hor_zeros_v2 )
```

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

Ввод [5]:
```
v_hor_one_v1 = np.ones((6,))
print(v_hor_one_v1)
v_hor_one_v2 = np.ones((1, 6))
print(v_hor_one_v2)
```

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

Вектор-столбец:

Ввод [6]:
```
v_vert_np = np.array([[3], [4]])
print(v_vert_np)
```

```
[[3]
 [4]]
```

Ввод [7]:
```
v_vert_zeros = np.zeros((6, 1))
print(v_vert_zeros)
```

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

Ввод [8]:
```
v_vert_ones = np.ones((6, 1))
print(v_vert_ones)
```

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

Квадратная матрица:

Ввод [9]:
```
m_sqr_arr = np.array([[2, 3, 4], [5, 6, 7], [8, 9, 10]])
print(m_sqr_arr)
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

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

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

Ввод [11]:
```
m_sqr_mx = np.matrix([[2, 3, 4], [5, 6, 7], [8, 9, 10]])
print(m_sqr_mx)
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

Ввод [12]:
```
m_sqr_mx = np.matrix('2 3 4; 5 6 7; 8 9 10')
print(m_sqr_mx)
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

Диагональная матрица:

Ввод [13]:
```
m_diag = [[1, 0, 0], [0, 2, 0], [0, 0, 3]]
m_diag_np = np.matrix(m_diag)
print(m_diag_np)
```

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

Ввод [14]:
```
m_sqr_mx = np.matrix('2 3 4; 5 6 7; 8 9 10')
```

Ввод [15]:
```
diag = np.diag(m_sqr_mx)
print(diag)
```

```
[ 2  6 10]
```

Ввод [16]:
```
m_diag_np = np.diag(np.diag(m_sqr_mx))
print(m_diag_np)
```

```
[[ 2  0  0]
 [ 0  6  0]
 [ 0  0 10]]
```

Единичная матрица:

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

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

Ввод [18]:
```
m_eye = np.eye(4)
print(m_eye)
```

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

Ввод [19]:
```
m_idnt = np.identity(4)
print(m_idnt)
```

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

Нулевая матрица:

Ввод [20]:
```
m_zeros = np.zeros((4, 4))
print(m_zeros)
```

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

Задание матрицы в общем виде:

Ввод [21]:
```
m_mx = np.matrix('2 3 4; 5 6 7')
print(m_mx)
```

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

Ввод [22]:
```
m_var = np.zeros((3, 4))
print(m_var)
```

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

Транспонирование матрицы:

Ввод [23]:
```
A = np.matrix('2 3 4; 5 6 7')
print(A)
```

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

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

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

Ввод [25]:
```
print(A.T)
```

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

Ввод [26]:
```
A = np.matrix('2 3 4; 5 6 7')
print(A)
R = (A.T).T
print(R)
```

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

Ввод [27]:
```
A = np.matrix('2 3 4; 5 6 7')
B = np.matrix('4 5 6; 7 8 9')
L = (A + B).T
R = A.T + B.T
print(L)
print(R)
```

```
[[ 6 12]
 [ 8 14]
 [10 16]]
[[ 6 12]
 [ 8 14]
 [10 16]]
```

Ввод [28]:
```
A = np.matrix('2 3; 5 6')
B = np.matrix('4 5; 7 8')
L = (A.dot(B)).T
R = (B.T).dot(A.T)
print(L)
print(R)
```

```
[[29 62]
 [34 73]]
[[29 62]
 [34 73]]
```

Ввод [29]:
```
A = np.matrix('4 5 6; 7 8 9')
k = 3
L = (k * A).T
R = k * (A.T)
print(L)
print(R)
```

```
[[12 21]
 [15 24]
 [18 27]]
[[12 21]
 [15 24]
 [18 27]]
```

Ввод [30]:
```
A = np.matrix('2 3; 5 6')
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'))
```

```
-3
-3
```

Действия над матрицами:

Умножение матрицы на число:

Ввод [31]:
```
A = np.matrix('4 5 6; 7 8 9')
C = 3 * A
print(C)
```

```
[[12 15 18]
 [21 24 27]]
```

Ввод [32]:
```
A = np.matrix('2 3; 5 6')
L = 1 * A
R = A
print(L)
print(R)
```

```
[[2 3]
 [5 6]]
[[2 3]
 [5 6]]
```

Ввод [33]:
```
A = np.matrix('2 3; 5 6')
Z = np.matrix('0 0; 0 0')
L = 0 * A
R = Z
print(L)
print(R)
```

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

Ввод [34]:
```
A = np.matrix('2 3; 5 6')
p = 2
q = 3
L = (p + q) * A
R = p * A + q * A
print(L)
print(R)
```

```
[[10 15]
 [25 30]]
[[10 15]
 [25 30]]
```

Ввод [35]:
```
A = np.matrix('2 3; 5 6')
p = 2
q = 3
L = (p * q) * A
R = p * (q * A)
print(L)
print(R)
```

```
[[12 18]
 [30 36]]
[[12 18]
 [30 36]]
```

Ввод [36]:
```
A = np.matrix('2 3; 5 6')
B = np.matrix('4 5; 7 8')
k = 3
L = k * (A + B)
R = k * A + k * B
print(L)
print(R)
```

```
[[18 24]
 [36 42]]
[[18 24]
 [36 42]]
```

Сложение матриц:

Ввод [37]:
```
A = np.matrix('4 5 6; 7 8 9')
B = np.matrix('8 1 5; 6 9 12')
C = A + B
print(C)
```

```
[[12  6 11]
 [13 17 21]]
```

Ввод [38]:
```
A = np.matrix('2 3; 5 6')
B = np.matrix('5 6; 7 8')
L = A + B
R = B + A
print(L)
print(R)
```

```
[[ 7  9]
 [12 14]]
[[ 7  9]
 [12 14]]
```

Ввод [39]:
```
A = np.matrix('2 3; 5 6')
B = np.matrix('5 6; 7 8')
C = np.matrix('7 8; 9 3')
L = A + (B + C)
R = (A + B) + C
print(L)
print(R)
```

```
[[14 17]
 [21 17]]
[[14 17]
 [21 17]]
```

Ввод [40]:
```
A = np.matrix('5 6; 7 8')
Z = np.matrix('0 0; 0 0')
L = A + (-1)*A
print(L)
print(Z)
```

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

Умножение матриц:

Ввод [41]:
```
A = np.matrix('1 2; 4 5')
B = np.matrix('7; 9')
C = A.dot(B)
print(C)
```

```
[[25]
 [73]]
```

Ввод [42]:
```
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]]
```

Ввод [43]:
```
A = np.matrix('2 3; 5 6')
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)
```

```
[[ 56  68]
 [119 146]]
[[ 56  68]
 [119 146]]
```

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

```
[[31 36]
 [67 78]]
[[40 51]
 [54 69]]
```

Ввод [45]:
```
A = np.matrix('2 3; 5 6')
E = np.matrix('1 0; 0 1')
L = E.dot(A)
R = A.dot(E)
print(L)
print(R)
print(A)
```

```
[[2 3]
 [5 6]]
[[2 3]
 [5 6]]
[[2 3]
 [5 6]]
```

Ввод [46]:
```
A = np.matrix('2 3; 5 6')
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]]
```

Определитель матрицы:

Ввод [47]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 3 1')
print(A)
```

```
[[-4 -1  1]
 [ 1  4 -1]
 [ 4  3  1]]
```

Ввод [48]:
```
np.linalg.det(A)
```

Out[48]:
```
-36.0
```

Ввод [49]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 3 1')
print(A)
print(A.T)
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)
```

```
[[-4 -1  1]
 [ 1  4 -1]
 [ 4  3  1]]
[[-4  1  4]
 [-1  4  3]
 [ 1 -1  1]]
-36.0
-36.0
```

Ввод [50]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 3 1')
print(A)
np.linalg.det(A)
```

```
[[-4 -1  1]
 [ 1  4 -1]
 [ 4  3  1]]
```

Out[50]:
```
-36.0
```

Ввод [51]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 3 1')
print(A)
B = np.matrix('10 3 -1; -4 -1 2; 2 3 1')
print(B)
round(np.linalg.det(A), 3)
round(np.linalg.det(B), 3)
```

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

Out[51]:
```
-36.0
```

Ввод [52]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 3 1')
print(A)
np.linalg.det(A)
```

```
[[-4 -1  1]
 [ 1  4 -1]
 [ 4  3  1]]
```

Out[52]: -36.0

Ввод [53]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 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  1]
 [ 1  4 -1]
 [ 4  3  1]]
[[-4 -1  1]
 [ 1  4 -1]
 [ 8  6  2]]
```

Out[53]: -72.0

Ввод [54]:
```
A = np.matrix('-5 -1 2; -3 -1 2; 8 3 1')
B = np.matrix('-4 -1 1; 1 4 -1; 4 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)
```

```
[[-5 -1  2]
 [-2  3  1]
 [ 8  3  1]]
[[-5 -1  2]
 [-3 -1  2]
 [ 8  3  1]]
[[-4 -1  1]
 [ 1  4 -1]
 [ 4  3  1]]
```

Out[54]: -22.0

Ввод [55]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 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  1]
 [ 1  4 -1]
 [ 4  3  1]]
[[-4 -1  1]
 [-7  2  1]
 [ 4  3  1]]
```

Out[55]: -36.0

Ввод [56]:
```
A = np.matrix('-4 -1 1; 1 4 -1; 4 3 1')
print(A)
k = 2
A[1, :] = A[0, :] + k * A[2, :]
round(np.linalg.det(A), 3)
```

```
[[-4 -1  1]
 [ 1  4 -1]
 [ 4  3  1]]
```

Out[56]: 0.0

Ввод [57]:
```
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[57]: 0.0

Обратная матрица:

Ввод [58]:
```
A = np.matrix('1 -3 4; 2 5 2; 2 -1 2')
A_inv = np.linalg.inv(A)
print(A_inv)
```

```
[[-0.33333333 -0.05555556  0.72222222]
 [-0.          0.16666667 -0.16666667]
 [ 0.33333333  0.13888889 -0.30555556]]
```

Ввод [59]:
```
A = np.matrix('1 -3 4; 2 5 2; 2 -1 2')
A_inv = np.linalg.inv(A)
A_inv_inv = np.linalg.inv(A_inv)
print(A)
print(A_inv_inv)
```

```
[[ 1 -3  4]
 [ 2  5  2]
 [ 2 -1  2]]
[[ 1. -3.  4.]
 [ 2.  5.  2.]
 [ 2. -1.  2.]]
```

Ввод [60]:
```
A = np.matrix('1 -3 4; 2 5 2; 2 -1 2')
L = np.linalg.inv(A.T)
R = (np.linalg.inv(A)).T
print(L)
print(R)
```

```
[[-3.33333333e-01  1.38777878e-17  3.33333333e-01]
 [-5.55555556e-02  1.66666667e-01  1.38888889e-01]
 [ 7.22222222e-01 -1.66666667e-01 -3.05555556e-01]]
[[-0.33333333 -0.          0.33333333]
 [-0.05555556  0.16666667  0.13888889]
 [ 0.72222222 -0.16666667 -0.30555556]]
```

Ввод [61]:
```
A = np.matrix('1 -3 4; 2 5 2; 2 -1 2')
B = np.matrix('1 -2 4; 3 1 -5; 9 -1 2')
L = np.linalg.inv(A.dot(B))
R = np.linalg.inv(B).dot(np.linalg.inv(A))
print(L)
print(R)
```

```
[[ 0.05882353  0.01960784 -0.07843137]
 [ 0.44444444 -0.00925926 -0.71296296]
 [ 0.12418301 -0.02342048 -0.15631808]]
[[ 0.05882353  0.01960784 -0.07843137]
 [ 0.44444444 -0.00925926 -0.71296296]
 [ 0.12418301 -0.02342048 -0.15631808]]
```

Ранг матрицы:

Ввод [62]:
```
m_eye = np.eye(3)
print(m_eye)
```

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

Ввод [63]:
```
rank = np.linalg.matrix_rank(m_eye)
print(rank)
```

```
3
```

Ввод [64]:
```
m_eye[2][2] = 0
print(m_eye)
rank = np.linalg.matrix_rank(m_eye)
print(rank)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 0.]]
2
```

Решение первой индивидуальной задачи

Вторая индивидуальная задача

Решение СЛАУ

Задание начальных условий:

Ввод [1]:
```
import numpy as np
```

Ввод [2]:
```
n = int(input("Количество неизвестных: "))
```

```
Количество неизвестных: 4
```

Ввод [3]:
```
A = np.random.randint(-10, 10, size=(n, n))
B = np.random.randint(-10, 10, size=(n, 1))
print(A)
print(B)
```

```
[[-10  -2  -5   2]
 [ -2   6   4  -9]
 [  7  -3  -7  -1]
 [ -2   4   0  -3]]
[[-8]
 [-4]
 [-9]
 [ 1]]
```

Матричный метод:

Ввод [4]:
```
ans = np.linalg.inv(A).dot(B)
print(ans.reshape(1, n))
```

```
[[0.46909871 1.91888412 0.65922747 1.91244635]]
```

Метод Крамера:

Ввод [5]:
```
ans = np.array([])
A_det = np.linalg.det(A)
for i in range(n):
    A_cp = A.copy()
    A_cp[:, i] = B.reshape(1, n)
    det_x = np.linalg.det(A_cp)
    x = det_x / A_det
    ans = np.append(ans, x)
print(ans)
```

```
[0.46909871 1.91888412 0.65922747 1.91244635]
```

Решение второй индивидуальной задачи

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ:

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

Вектор-строка

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

Вектор-столбец

v_vert_np = np.array([[1], [2]])
Квадратная матрица
m_sqr_arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
Диагональная матрица
m_diag = [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
Единичная матрица
m_e = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
Нулевая матрица
m_zeros = np.zeros((3, 3))
Прямоугольная матрциа
m_pr_arr = np.array([[1, 2, 3], [4, 5, 6]])

2. Как выполняется транспонирование матриц?

Транспонирование матрицы – это процесс замены строк матрицы на ее столбцы, а столбцов соответственно на строки. Полученная в результате матрица называется транспонированной.

3. Приведите свойства операции транспонирования матриц.

$$(A^T)^T = A.$$

$$(A + B)^T = A^T + B^T.$$

$$(AB)^T = B^T A^T.$$

$$(\lambda A)^T = \lambda A^T.$$

$$|A| = \left|A^T\right|.$$

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

A.transpose()
A.T

5. Какие существуют основные действия над матрицами?

Умножение матрицы на число
Сложение матриц
Умножение матриц
Определитель матрицы
Обратная матрица
Ранг матрицы

6. Как осуществляется умножение матрицы на число?

При умножении матрицы на число, все элементы матрицы умножаются на это число:

7. Какие свойства операции умножения матрицы на число?

$$1 \cdot A = A.$$

$$0 \cdot A = Z.$$

$$(\alpha + \beta) \cdot A = \alpha \cdot A + \beta \cdot A.$$

$$(\alpha \cdot \beta) \cdot A = \alpha \cdot (\beta \cdot A).$$

$$\lambda \cdot (A + B) = \lambda \cdot A + \lambda \cdot B.$$

8. Как осуществляется операции сложения и вычитания матриц?

Складывать можно только матрицы одинаковой размерности — то есть матрицы, у которых совпадает количество столбцов и строк.

9. Каковы свойства операций сложения и вычитания матриц?

$$A + B = B + A.$$

$$A + (B + C) = (A + B) + C.$$

$$A + (-A) = Z.$$

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

C = A + B

11. Как осуществляется операция умножения матриц?

Каждый элемент cij новой матрицы является суммой произведений элементов i-ой строки первой матрицы и j-го столбца второй матрицы.

12. Каковы свойства операции умножения матриц?

$$A \times (B \times C) = (A \times B) \times C.$$

$$A \times (B + C) = A \times B + A \times C.$$

$$A \times B \neq B \times A.$$

$$E \times A = A \times E = A.$$

$$Z \times A = A \times Z = Z.$$

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

C = A.dot(B)

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

Определитель матрицы размера (n-го порядка) является одной из ее численных характеристик.

В общем виде вычислить определитель матрицы можно через разложение определителя по элементам строки или столбца. Суть в том, что определитель равен сумме произведений элементов любой строки или столбца на их алгебраические дополнения.

$$det(A) = det(A^T).$$

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

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

$$det(A) = -det(A').$$

$$\begin{vmatrix} a_1 & a_2 & \dots & a_n \\ a_1 & a_2 & \dots & a_n \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \lambda \cdot a_{21} & \lambda \cdot a_{22} & \dots & \lambda \cdot a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \lambda \cdot det(A).$$

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

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21}+\beta \cdot a_{11} & a_{22}+\beta \cdot a_{12} & \dots & a_{2n}+\beta \cdot a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}.$$

$$a_{2i} = \alpha \cdot a_{1i} + \beta \cdot a_{3i}; \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \beta \cdot a_{11} & \beta \cdot a_{12} & \dots & \beta \cdot a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} = 0.$$

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

np.linalg.det(A)

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

Обратной матрицей матрицы называют матрицу, удовлетворяющую следующему равенству:

$$A \times A^{-1} = A^{-1} \times A = E,$$

где – E это единичная матрица.

Для того, чтобы у квадратной матрицы A была обратная матрица необходимо и достаточно чтобы определитель |A| был не равен нулю.

17. Каковы свойства обратной матрицы?

$$(A^{-1})^{-1} = A.$$

$$(A^T)^{-1} = (A^{-1})^T.$$

$$(A_1 \times A_2)^{-1} = A_2^{-1} \times A_1^{-1}.$$

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

np.linalg.inv(A)

19. Самостоятельно изучите метод Крамера для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений методом Крамера средствами библиотеки NumPy.

Матричный метод:

Ввод [4]:
```
ans = np.linalg.inv(A).dot(B)
print(ans.reshape(1, n))
```

```
[[0.46909871 1.91888412 0.65922747 1.91244635]]
```

20. Самостоятельно изучите матричный метод для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений матричным методом средствами библиотеки NumPy.

Метод Крамера:

Ввод [5]:
```
ans = np.array([])
A_det = np.linalg.det(A)
for i in range(n):
    A_cp = A.copy()
    A_cp[::, i] = B.reshape(1, n)
    det_x = np.linalg.det(A_cp)
    x = det_x / A_det
    ans = np.append(ans, x)
print(ans)
```

```
[0.46909871 1.91888412 0.65922747 1.91244635]
```