Тема “Визуализация данных в Matplotlib”

Задание 1

Загрузите модуль pyplot библиотеки matplotlib с псевдонимом plt, а также библиотеку numpy с псевдонимом np. Примените магическую функцию %matplotlib inline для отображения графиков в Jupyter Notebook и настройки конфигурации ноутбука со значением 'svg' для более четкого отображения графиков. Создайте список под названием x с числами 1, 2, 3, 4, 5, 6, 7 и список y с числами 3.5, 3.8, 4.2, 4.5, 5, 5.5, 7. С помощью функции plot постройте график, соединяющий линиями точки с горизонтальными координатами из списка x и вертикальными - из списка y. Затем в следующей ячейке постройте диаграмму рассеяния (другие названия - диаграмма разброса, scatter plot).

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
%matplotlib inline
```

In [3]:
```
%config InlineBackend.figure_format = 'svg'
```

In [4]:
```
x = np.arange(1, 8)
```

In [5]:
```
x
```

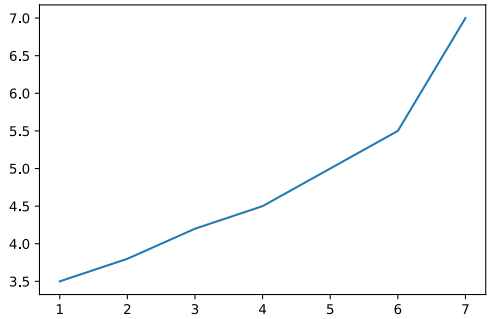
Out[5]:
```
array([1, 2, 3, 4, 5, 6, 7])
```

In [6]:
```
y = np.asarray([3.5, 3.8, 4.2, 4.5, 5, 5.5, 7])
```

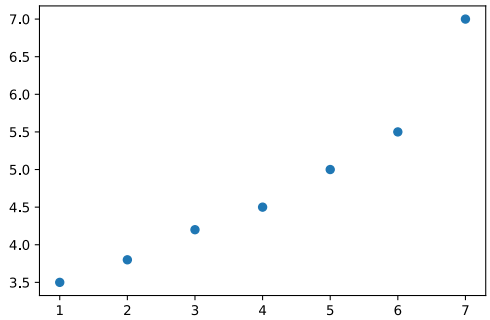
In [7]:
```
y
```

Out[7]:
```
array([3.5, 3.8, 4.2, 4.5, 5. , 5.5, 7. ])
```

In [8]:
```
plt.plot(x, y)
plt.show()
```



In [9]:
```
plt.scatter(x, y)
plt.show()
```



Задание 2

С помощью функции linspace из библиотеки Numpy создайте массив t из 51 числа от 0 до 10 включительно. Создайте массив Numpy под названием f, содержащий косинусы элементов массива t. Постройте линейную диаграмму, используя массив t для координат по горизонтали,а массив f - для координат по вертикали. Линия графика должна быть зеленого цвета. Выведите название диаграммы - 'График f(t)'. Также добавьте названия для горизонтальной оси - 'Значения t' и для вертикальной - 'Значения f'. Ограничьте график по оси x значениями 0.5 и 9.5, а по оси y - значениями -2.5 и 2.5.

In [10]:
```
t = np.linspace(0,10,51)
```

In [11]:
```
t
```

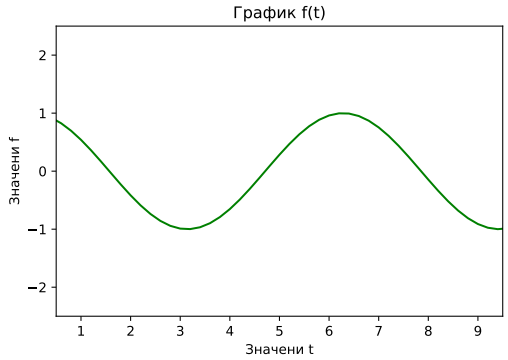
Out[11]:
```
array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ,  1.2,  1.4,  1.6,  1.8,  2. ,
        2.2,  2.4,  2.6,  2.8,  3. ,  3.2,  3.4,  3.6,  3.8,  4. ,  4.2,
        4.4,  4.6,  4.8,  5. ,  5.2,  5.4,  5.6,  5.8,  6. ,  6.2,  6.4,
        6.6,  6.8,  7. ,  7.2,  7.4,  7.6,  7.8,  8. ,  8.2,  8.4,  8.6,
        8.8,  9. ,  9.2,  9.4,  9.6,  9.8, 10. ])
```

In [12]:
```
f = np.cos(t)
f
```

Out[12]:
```
array([ 1.        ,  0.98006658,  0.92106099,  0.82533561,  0.69670671,
        0.54030231,  0.36235775,  0.16996714, -0.02919952, -0.22720209,
       -0.41614684, -0.58850112, -0.73739372, -0.85688875, -0.94222234,
       -0.9899925 , -0.99829478, -0.96679819, -0.89675842, -0.79096771,
       -0.65364362, -0.49026082, -0.30733287, -0.11215253,  0.08749898,
        0.28366219,  0.46851667,  0.63469288,  0.77556588,  0.88551952,
        0.96017029,  0.9965421 ,  0.99318492,  0.95023259,  0.86939749,
```

```
 0.75390225,  0.60835131,  0.43854733,  0.25125984,  0.05395542,
-0.14550003, -0.33915486, -0.51928865, -0.67872005, -0.81109301,
-0.91113026, -0.97484362, -0.99969304, -0.98468786, -0.93042627,
-0.83907153])
```

In [13]:
```
plt.plot(t,f,color='green')
plt.title('График f(t)')
plt.xlabel('Значени t')
plt.ylabel('Значени f')
plt.axis([0.5, 9.5, -2.5, 2.5])
plt.show()
```



Задание 3

С помощью функции linspace библиотеки Numpy создайте массив x из 51 числа от -3 до 3 включительно. Создайте массивы y1, y2, y3, y4 по следующим формулам: y1 = x*2 y2 = 2 x + 0.5 y3 = -3 * x - 1.5 y4 = sin(x) Используя функцию subplots модуля matplotlib.pyplot, создайте объект matplotlib.figure.Figure с названием fig и массив объектов Axes под названием ax,причем так, чтобы у вас было 4 отдельных графика в сетке, состоящей из двух строк и двух столбцов. В каждом графике массив x используется для координат по горизонтали.В левом верхнем графике для координат по вертикали используйте y1,в правом верхнем - y2, в левом нижнем - y3, в правом нижнем - y4.Дайте название графикам: 'График y1', 'График y2' и т.д. Для графика в левом верхнем углу установите границы по оси x от -5 до 5. Установите размеры фигуры 8 дюймов по горизонтали и 6 дюймов по вертикали. Вертикальные и горизонтальные зазоры между графиками должны составлять 0.3.

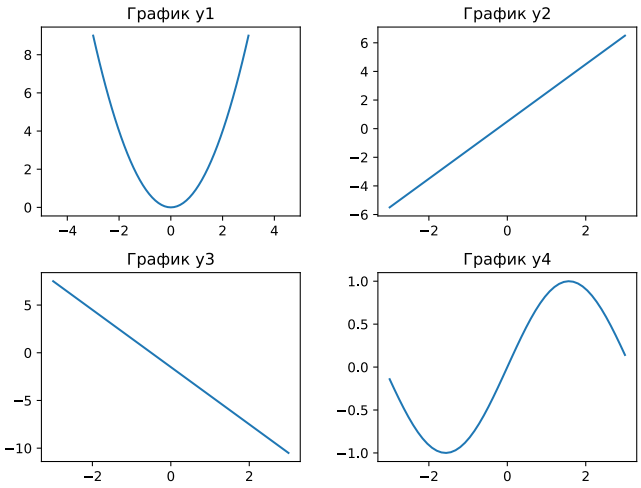
In [14]:
```
x = np.linspace(-3, 3, 51)
x
```

Out[14]:
```
array([-3.  , -2.88, -2.76, -2.64, -2.52, -2.4 , -2.28, -2.16, -2.04,
       -1.92, -1.8 , -1.68, -1.56, -1.44, -1.32, -1.2 , -1.08, -0.96,
       -0.84, -0.72, -0.6 , -0.48, -0.36, -0.24, -0.12,  0.  ,  0.12,
        0.24,  0.36,  0.48,  0.6 ,  0.72,  0.84,  0.96,  1.08,  1.2 ,
        1.32,  1.44,  1.56,  1.68,  1.8 ,  1.92,  2.04,  2.16,  2.28,
        2.4 ,  2.52,  2.64,  2.76,  2.88,  3.  ])
```

In [15]:
```
y1 = x**2
y2 = 2 * x + 0.5
y3 = -3 * x -1.5
y4 = np.sin(x)
y1, y2, y3, y4
```

Out[15]:
```
(array([9.    , 8.2944, 7.6176, 6.9696, 6.3504, 5.76  , 5.1984, 4.6656,
        4.1616, 3.6864, 3.24  , 2.8224, 2.4336, 2.0736, 1.7424, 1.44  ,
        1.1664, 0.9216, 0.7056, 0.5184, 0.36  , 0.2304, 0.1296, 0.0576,
        0.0144, 0.    , 0.0144, 0.0576, 0.1296, 0.2304, 0.36  , 0.5184,
        0.7056, 0.9216, 1.1664, 1.44  , 1.7424, 2.0736, 2.4336, 2.8224,
        3.24  , 3.6864, 4.1616, 4.6656, 5.1984, 5.76  , 6.3504, 6.9696,
        7.6176, 8.2944, 9.    ]),
 array([-5.5 , -5.26, -5.02, -4.78, -4.54, -4.3 , -4.06, -3.82, -3.58,
        -3.34, -3.1 , -2.86, -2.62, -2.38, -2.14, -1.9 , -1.66, -1.42,
        -1.18, -0.94, -0.7 , -0.46, -0.22,  0.02,  0.26,  0.5 ,  0.74,
         0.98,  1.22,  1.46,  1.7 ,  1.94,  2.18,  2.42,  2.66,  2.9 ,
         3.14,  3.38,  3.62,  3.86,  4.1 ,  4.34,  4.58,  4.82,  5.06,
         5.3 ,  5.54,  5.78,  6.02,  6.26,  6.5 ]),
 array([  7.5 ,   7.14,   6.78,   6.42,   6.06,   5.7 ,   5.34,   4.98,
          4.62,   4.26,   3.9 ,   3.54,   3.18,   2.82,   2.46,   2.1 ,
          1.74,   1.38,   1.02,   0.66,   0.3 ,  -0.06,  -0.42,  -0.78,
         -1.14,  -1.5 ,  -1.86,  -2.22,  -2.58,  -2.94,  -3.3 ,  -3.66,
         -4.02,  -4.38,  -4.74,  -5.1 ,  -5.46,  -5.82,  -6.18,  -6.54,
         -6.9 ,  -7.26,  -7.62,  -7.98,  -8.34,  -8.7 ,  -9.06,  -9.42,
         -9.78, -10.14, -10.5 ]),
 array([-0.14112001, -0.25861935, -0.37239904, -0.48082261, -0.58233065,
        -0.67546318, -0.75888071, -0.83138346, -0.89192865, -0.93964547,
        -0.97384763, -0.9940432 , -0.99994172, -0.99145835, -0.9687151 ,
        -0.93203909, -0.88195781, -0.81919157, -0.74464312, -0.65938467,
        -0.56464247, -0.46177918, -0.35227423, -0.23770263, -0.11971221,
         0.        ,  0.11971221,  0.23770263,  0.35227423,  0.46177918,
         0.56464247,  0.65938467,  0.74464312,  0.81919157,  0.88195781,
         0.93203909,  0.9687151 ,  0.99145835,  0.99994172,  0.9940432 ,
         0.97384763,  0.93964547,  0.89192865,  0.83138346,  0.75888071,
         0.67546318,  0.58233065,  0.48082261,  0.37239904,  0.25861935,
         0.14112001]))
```

In [16]:
```
fig, ax = plt.subplots(nrows=2, ncols=2)
ax1, ax2, ax3, ax4 = ax.flatten()
ax1
ax1.plot(x, y1)
ax1.set_title('График y1')
ax1.set_xlim([-5, 5])
ax2.plot(x, y2)
ax2.set_title('График y2')
ax3.plot(x, y3)
ax3.set_title('График y3')
ax4.plot(x, y4)
ax4.set_title('График y4')
fig.set_size_inches(8, 6)
fig.subplots_adjust(wspace=0.3, hspace=0.3)
```



Задание 4

В этом задании мы будем работать с датасетом, в котором приведены данные по мошенничеству с кредитными данными: Credit Card Fraud Detection (информация об авторах: Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015). Ознакомьтесь с описанием и скачайте датасет creditcard.csv с сайта Kaggle.com. Данный датасет является примером несбалансированных данных, так как мошеннические операции с картами встречаются реже обычных. Импортируйте библиотеку Pandas, а также используйте для графиков стиль "fivethirtyeight". Посчитайте с помощью метода value_counts количество наблюдений для каждого значения целевой переменной Class и примените к полученным данным метод plot, чтобы построить столбчатую диаграмму. Затем постройте такую же диаграмму, используя логарифмический масштаб. На следующем графике постройте две гистограммы по значениям признака V1 - одну для мошеннических транзакций (Class равен 1) и другую - для обычных (Class равен 0). Подберите значение аргумента density так, чтобы по вертикали графика было расположено не число наблюдений, а плотность распределения. Число бинов должно равняться 20 для обеих гистограмм, а коэффициент alpha сделайте равным 0.5, чтобы гистограммы были полупрозрачными и не загораживали друг друга. Создайте легенду с двумя значениями: "Class 0" и "Class 1". Гистограмма обычных транзакций должна

In [17]:
```
import pandas as pd
```

In [18]:
```
plt.style.use('fivethirtyeight')
```

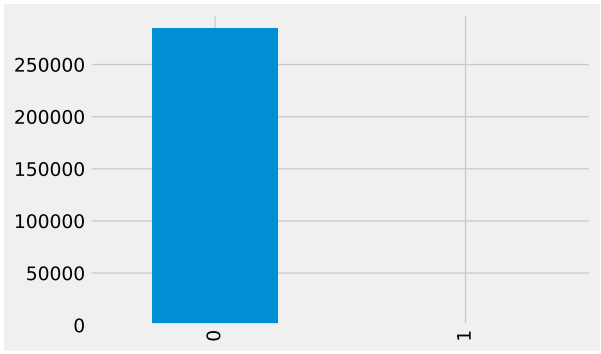
In [19]:
```
df = pd.read_csv('creditcard.csv')
```

In [20]:
```
class_counts = df.Class.value_counts()
class_counts
```

Out[20]:
```
0    284315
1       492
Name: Class, dtype: int64
```

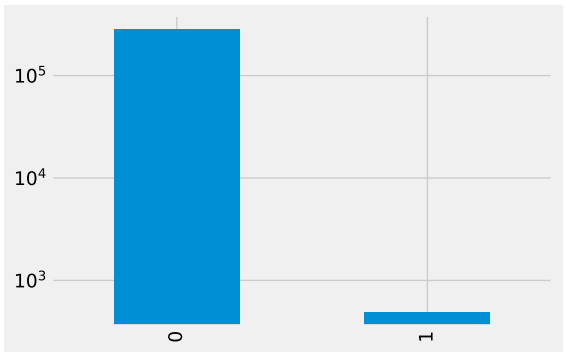
In [21]:
```
class_counts.plot(kind='bar')
```

Out[21]: <AxesSubplot:>



In [22]:
```
class_counts.plot(kind='bar', logy=True)
```

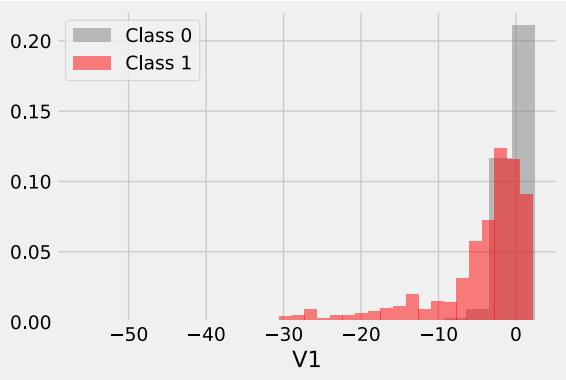
Out[22]: <AxesSubplot:>



In [23]:
```
plt.hist(df[df.Class==0]['V1'],color='grey', density=True, bins=20, alpha = 0.5)
plt.hist(df[df.Class==1]['V1'],color='red', density=True, bins=20, alpha = 0.5)
```

```
plt.legend(labels=['Class 0', 'Class 1'])
plt.xlabel('V1')
```

Out[23]:
```
Text(0.5, 0, 'V1')
```



Задание на повторение материала

1. Создать одномерный массив Numpy под названием a из 12 последовательных целых чисел чисел от 12 до 24 невключительно

In [24]:
```
a = np.arange(12, 24)
a
```

Out[24]:
```
array([12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23])
```

1. Создать 5 двумерных массивов разной формы из массива a. Не использовать в аргументах метода reshape число -1.

In [25]:
```
a1 = a.reshape(2, 6)
a2 = a.reshape(3, 4)
a3 = a.reshape(4, 3)
a4 = a.reshape(6, 2)
a5 = a.reshape(12, 1)
a1, a2, a3, a4, a5
```

Out[25]:
```
(array([[12, 13, 14, 15, 16, 17],
        [18, 19, 20, 21, 22, 23]]),
 array([[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]),
 array([[12, 13, 14],
        [15, 16, 17],
        [18, 19, 20],
        [21, 22, 23]]),
 array([[12, 13],
        [14, 15],
        [16, 17],
        [18, 19],
        [20, 21],
        [22, 23]]),
 array([[12],
        [13],
        [14],
        [15],
        [16],
        [17],
        [18],
        [19],
        [20],
        [21],
        [22],
        [23]]))
```

1. Создать 5 двумерных массивов разной формы из массива a. Использовать в аргументах метода reshape число -1 (в трех примерах - для обозначения числа столбцов, в двух - для строк).

In [26]:
```
a1 = a.reshape(2, -1)
a2 = a.reshape(3, -1)
a3 = a.reshape(4, -1)
a4 = a.reshape(-1, 2)
a5 = a.reshape(-1, 1)
a1, a2, a3, a4, a5
```

Out[26]:
```
(array([[12, 13, 14, 15, 16, 17],
        [18, 19, 20, 21, 22, 23]]),
 array([[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]),
 array([[12, 13, 14],
        [15, 16, 17],
        [18, 19, 20],
        [21, 22, 23]]),
 array([[12, 13],
        [14, 15],
        [16, 17],
        [18, 19],
        [20, 21],
        [22, 23]]),
 array([[12],
        [13],
        [14],
        [15],
        [16],
        [17],
        [18],
        [19],
        [20],
        [21],
        [22],
        [23]]))
```

1. Можно ли массив Numpy, состоящий из одного столбца и 12 строк, назвать одномерным?

In [27]:
```
#Нет, так как с точки зрения питру у него будет два измерения.
a5.shape, a5.ndim
#Но с точки зрения математики это будет одномерный массив.
```

Out[27]:
```
((12, 1), 2)
```

1. Создать массив из 3 строк и 4 столбцов, состоящий из случайных чисел с плавающей запятой из нормального распределения со средним, равным 0 и среднеквадратичным отклонением, равным 1.0. Получить из этого массива одномерный массив с таким же атрибутом size, как и исходный массив.

In [28]:
```
m = np.random.randn(3, 4)
m
```

Out[28]:
```
array([[-1.5525887 , -0.32985865,  0.58890867, -0.8634315 ],
       [-0.36171061, -0.91357574,  0.14787646, -0.23008706],
       [ 0.10393998,  1.25373514,  0.98292881,  0.29379665]])
```

In [29]:
```
m.size
```

Out[29]:
```
12
```

In [30]:
```
m_1 = m.flatten()
m_1
```

Out[30]:
```
array([-1.5525887 , -0.32985865,  0.58890867, -0.8634315 , -0.36171061,
       -0.91357574,  0.14787646, -0.23008706,  0.10393998,  1.25373514,
        0.98292881,  0.29379665])
```

In [31]:
```
m_1.shape
```

Out[31]:
```
(12,)
```

In [32]:
```
m_1.size
```

Out[32]:
```
12
```

1. Создать массив a, состоящий из целых чисел, убывающих от 20 до 0 невключительно с интервалом 2.

In [33]:
```
a = np.arange(20, 0, -2)
a
```

Out[33]:
```
array([20, 18, 16, 14, 12, 10,  8,  6,  4,  2])
```

1. Создать массив b, состоящий из 1 строки и 10 столбцов: целых чисел, убывающих от 20 до 1 невключительно с интервалом 2. В чем разница между массивами a и b?

In [34]:
```
b = np.arange(20, 0, -2).reshape(10,1)
b
```

Out[34]:
```
array([[20],
       [18],
       [16],
       [14],
       [12],
       [10],
       [ 8],
       [ 6],
       [ 4],
       [ 2]])
```

In [35]:
```
#У массива b  два измерения, когда как у массива a - одно. Массив b представляет вертикальный вектор,
#а массив a - горизонтальный.
```

1. Вертикально соединить массивы a и b. a - двумерный массив из нулей, число строк которого больше 1 и на 1 меньше, чем число строк двумерного массива b, состоящего из единиц. Итоговый массив v должен иметь атрибут size, равный 10.

In [36]:
```
a = np.zeros(shape=(2, 2))
b = np.ones(shape=(3, 2))
v = np.vstack((a, b))
a, b, v, v.size
```

Out[36]:
```
(array([[0., 0.],
        [0., 0.]]),
 array([[1., 1.],
        [1., 1.],
        [1., 1.]]),
 array([[0., 0.],
        [0., 0.],
        [1., 1.],
        [1., 1.],
        [1., 1.]]),
 10)
```

1. Создать одномерный массив a, состоящий из последовательности целых чисел от 0 до 12. Поменять форму этого массива, чтобы получилась матрица A (двумерный массив Numpy), состоящая из 4 строк и 3 столбцов. Получить матрицу At путем транспонирования матрицы A. Получить матрицу B, умножив матрицу A на матрицу At с помощью матричного умножения. Какой размер имеет матрица B? Получится ли вычислить обратную матрицу для матрицы B и почему?

In [37]:
```
a = np.arange(0, 12)
a
```

Out[37]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

In [38]:
```
A = a.reshape(4, 3)
A
```

Out[38]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

In [39]:
```
At = A.T
At
```

Out[39]:
```
array([[ 0,  3,  6,  9],
       [ 1,  4,  7, 10],
       [ 2,  5,  8, 11]])
```

In [40]:
```
B = A @ At
B
```

Out[40]:
```
array([[  5,  14,  23,  32],
       [ 14,  50,  86, 122],
       [ 23,  86, 149, 212],
       [ 32, 122, 212, 302]])
```

In [41]:
```
#Матрица B имеет размер 4x4.
#Не получится вычислить обратную матрицу для матрицы B, так как есть линейная зависимость между строками/столбцами матрицы.
#Например, если добавить 1й и 3й столбец матрицы и поделить на 2, то получится 2й столбец матрицы.
```

1. Инициализируйте генератор случайных числе с помощью объекта seed, равного 42.

In [42]:
```
np.random.seed(42)
```

1. Создайте одномерный массив c, составленный из последовательности 16-ти случайных равномерно распределенных целых чисел от 0 до 16 невключительно.

In [43]:
```
c = np.random.randint(0, 16, 16)
c
```

Out[43]:
```
array([ 6,  3, 12, 14, 10,  7, 12,  4,  6,  9,  2,  6, 10, 10,  7,  4])
```

1. Поменяйте его форму так, чтобы получилась квадратная матрица C. Получите матрицу D, поэлементно прибавив матрицу B из предыдущего вопроса к матрице C, умноженной на 10. Вычислите определитель, ранг и обратную матрицу D_inv для D.

In [44]:
```
C = c.reshape(4, 4)
C
```

Out[44]:
```
array([[ 6,  3, 12, 14],
       [10,  7, 12,  4],
       [ 6,  9,  2,  6],
       [10, 10,  7,  4]])
```

In [45]:
```
D = B + 10*C
D
```

Out[45]:
```
array([[ 65,  44, 143, 172],
       [114, 120, 206, 162],
       [ 83, 176, 169, 272],
       [132, 222, 282, 342]])
```

In [46]:
```
np.linalg.matrix_rank(D)
```

Out[46]:
```
4
```

In [47]:
```
np.linalg.det(D)
```

Out[47]:
```
-28511999.999999944
```

In [48]:
```
D_inv = np.linalg.inv(D)
D_inv
```

Out[48]:
```
array([[ 0.00935396,  0.04486532,  0.05897517, -0.07286055],
       [-0.01503577, -0.00122896, -0.00192971,  0.00967873],
       [-0.00356692, -0.01782828, -0.04152146,  0.04326178],
       [ 0.00909091, -0.00181818,  0.01272727, -0.01090909]])
```

1. Приравняйте к нулю отрицательные числа в матрице D_inv, а положительные - к единице. Убедитесь, что в матрице D_inv остались только нули и единицы. С помощью функции numpy.where, используя матрицу D_inv в качестве маски, а матрицы B и C - в качестве источников данных, получите матрицу E размером 4x4. Элементы матрицы E, для которых соответствующий элемент матрицы D_inv равен 1, должны быть равны соответствующему элементу матрицы B, а элементы матрицы E, для которых соответствующий элемент матрицы D_inv равен 0, должны быть равны соответствующему элементу матрицы C.

In [49]:
```
D_inv[D_inv<=0] = 0
D_inv[D_inv>0] = 1
D_inv
```

Out[49]:
```
array([[1., 1., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 1.],
       [1., 0., 1., 0.]])
```

In [50]:
```
E = np.where(D_inv, B, C)
E
```

Out[50]:
```
array([[  5,  14,  23,  14],
       [ 10,   7,  12, 122],
       [  6,   9,   2, 212],
       [ 32,  10, 212,   4]])
```

In [ ]: