In [1]:

```
#1. Загрузите модуль pyplot библиотеки matplotlib с псевдонимом plt, а также библиотеку num
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
#Примените магическую функцию %matplotlib inline для отображения графиков в Jupyter Noteboo
#и настройки конфигурации ноутбука со значением 'svg' для более четкого отображения графико
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
```

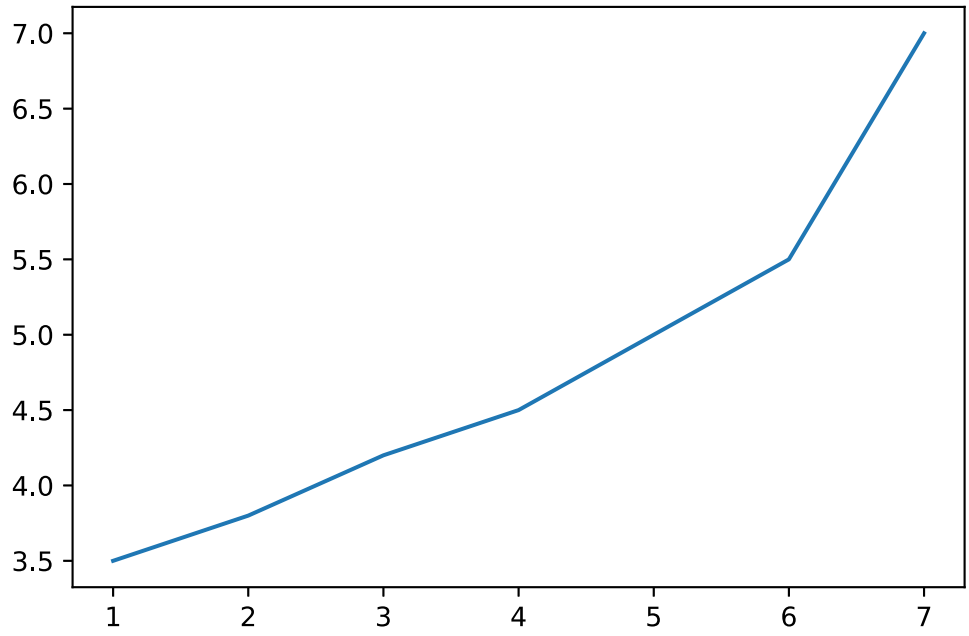
In [3]:

```
#Создайте список под названием x с числами 1, 2, 3, 4, 5, 6, 7 и список y с числами 3.5, 3.
x=[1, 2, 3, 4, 5, 6, 7]
y=[3.5, 3.8, 4.2, 4.5, 5, 5.5, 7.]
print(x)
print(y)
```

```
[1, 2, 3, 4, 5, 6, 7]
[3.5, 3.8, 4.2, 4.5, 5, 5.5, 7.0]
```

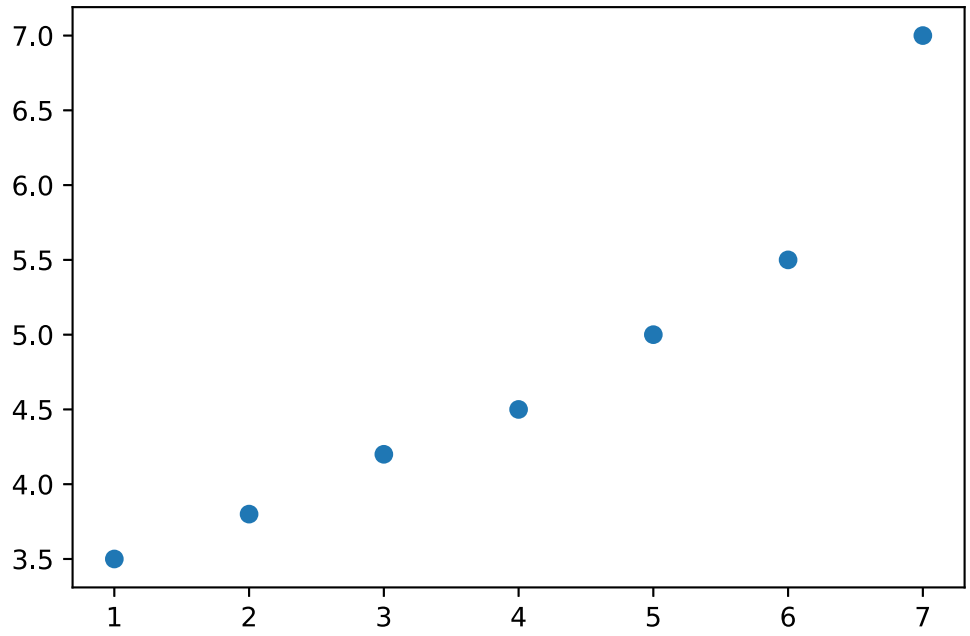
In [4]:

```
#С помощью функции plot постройте график, соединяющий линиями точки с горизонтальными
#координатами из списка x и вертикальными из списка y.
plt.plot(x,y)
plt.show()
```



In [5]:

```
#Затем в следующей ячейке постройте диаграмму рассеяния
#(другие названия – диаграмма разброса, scatter plot).
plt.scatter(x, y)
plt.show()
```



In [6]:

```
#1. С помощью функции linspace из библиотеки Numpy создайте массив t из 51 числа от 0 до 10
t = np.linspace(0, 10, 51)
t
```

Out[6]:

```
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 [7]:

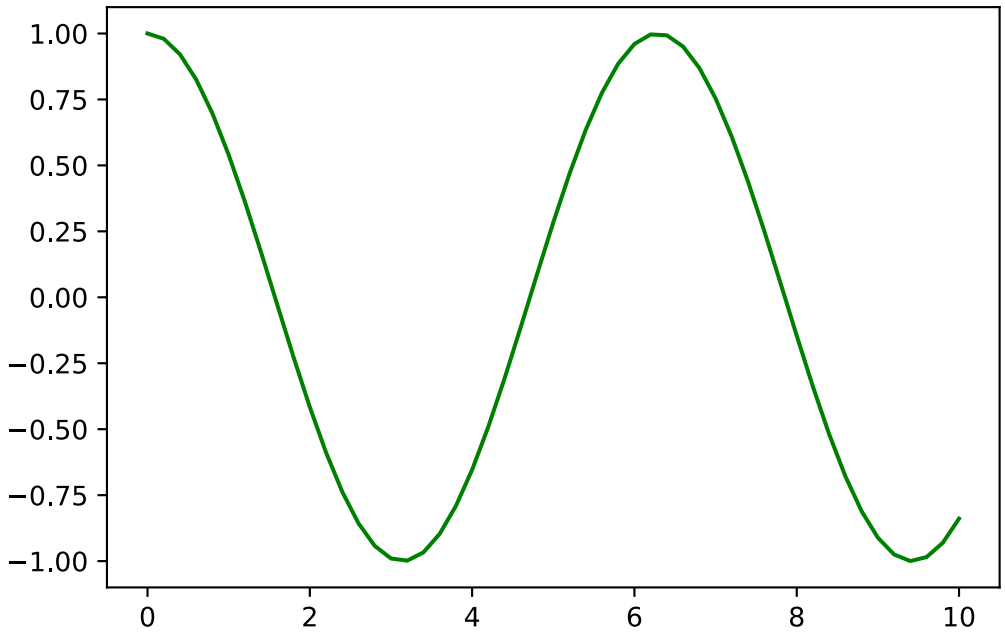
```
#Создайте массив Numpy под названием f, содержащий косинусы элементов массива t.
f=np.cos(t)
f
```

Out[7]:

```
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])
```

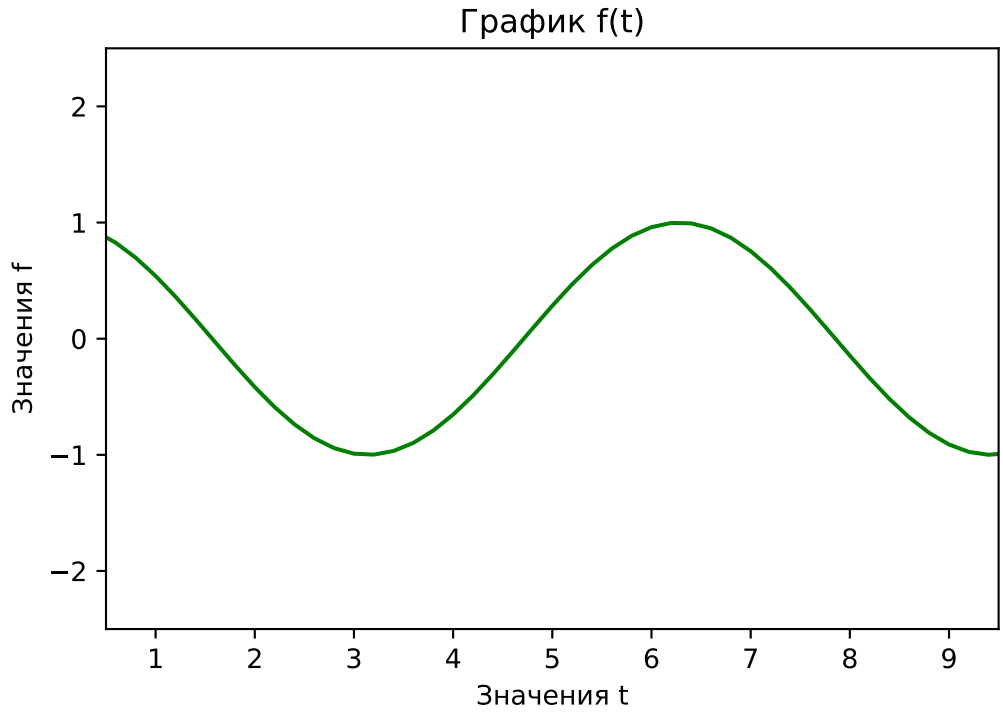
In [8]:

```
#Постройте линейную диаграмму, используя массив t для координат по горизонтали,
#а массив f – для координат по вертикали. Линия графика должна быть зеленого цвета.
plt.plot(t,f,color="green")
plt.show()
```



In [9]:

```
#Выведите название диаграммы - 'График f(t)'. Также добавьте названия для горизонтальной
#оси – 'Значения t' и для вертикальной – 'Значения f'. Ограничьте график по оси x значениям
#0.5 и 9.5, а по оси y - значениями -2.5 и 2.5.
plt.axis([0.5, 9.5, -2.5, 2.5])
plt.plot(t,f,color="green")
plt.title('График f(t)')
plt.xlabel('Значения t')
plt.ylabel('Значения f')
plt.show()
```



In [10]:

```
#3. С помощью функции linspace библиотеки Numpy создайте массив x из 51 числа – от -3 до 3
x = np.linspace(-3, 3, 51)
x
```

Out[10]:

```
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.  ])
```

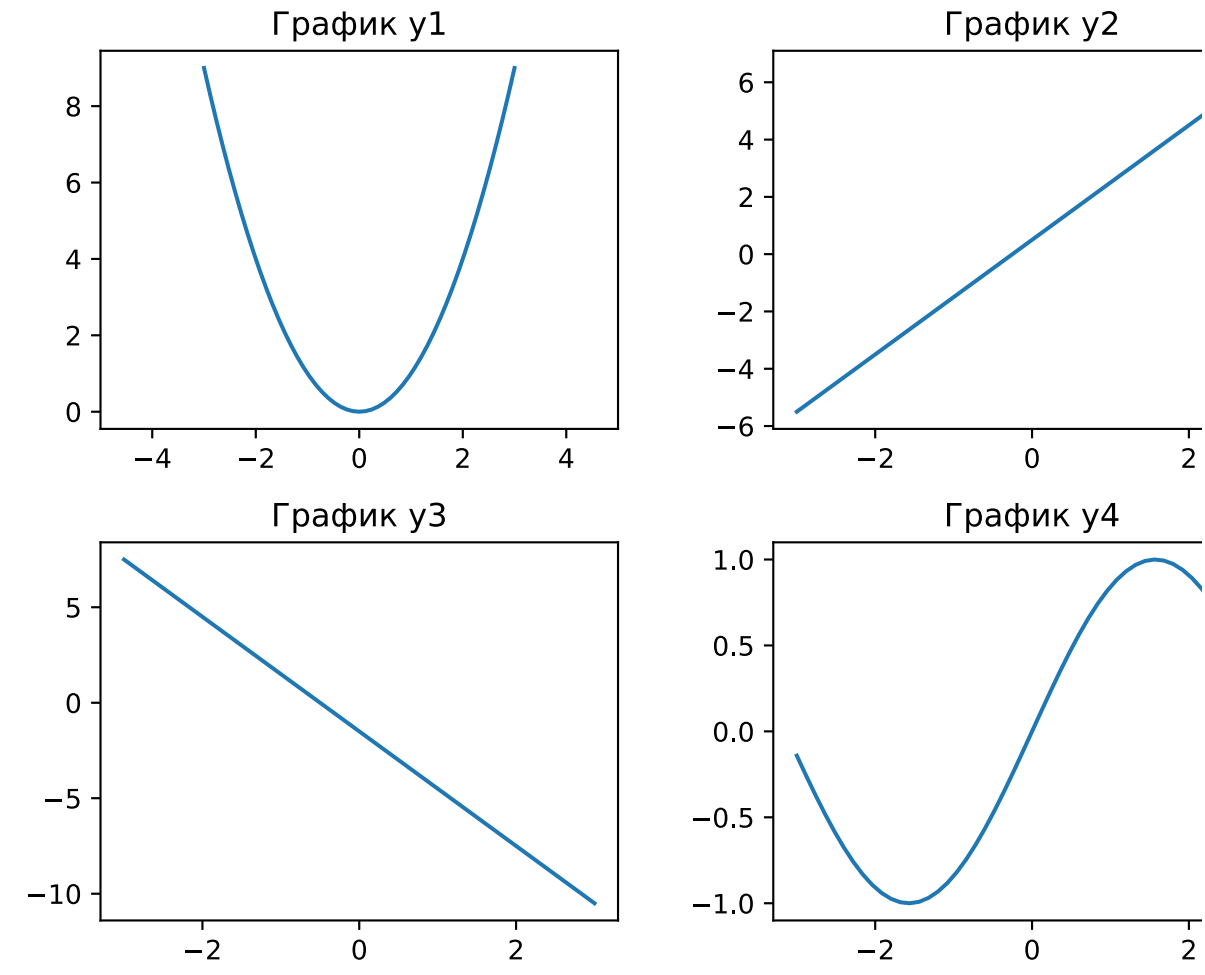
In [11]:

```
#Создайте массивы y1, y2, y3, y4 по следующим формулам: y1 = x**2 y2 = 2 * x + 0.5 y3 = -3
y1 = x**2
y2 = 2 * x + 0.5
y3 = -3 * x - 1.5
y4 = np.sin(x)
print(y1)
print(y2)
print(y3)
print(y4)
```

```
[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.    ]
[-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 ]
[  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 ]
[-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 [12]:

```
#Используя функцию subplots модуля matplotlib.pyplot, создайте объект matplotlib.figure.Fig
#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 масси
#x используется для координат по горизонтали. В левом верхнем графике для координат по верт
#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 размер
#фигуры 8 дюймов по горизонтали и 6 дюймов по вертикали. Вертикальные и горизонтальные зазо
#между графиками должны составлять 0.3.

fig, ax = plt.subplots(nrows=2, ncols=2)
ax1, ax2, ax3, ax4 = ax.flatten()

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)
plt.subplots_adjust(wspace=0.3, hspace=0.3)
```



In [13]:

```
#используйте для графиков стиль fivethirtyeight.
plt.style.use('fivethirtyeight')
```

In [14]:

```
#загрузить датасет
df = pd.read_csv('creditcard.csv')
df.head()
```

Out[14]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

In [15]:

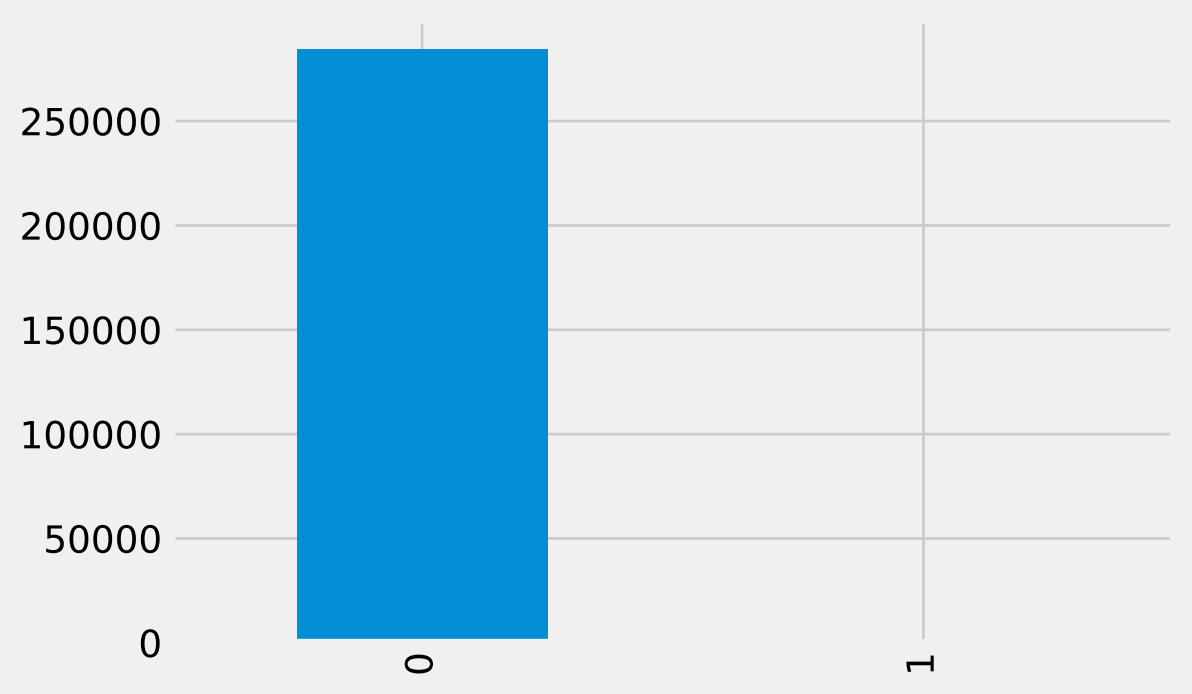
```
#Посчитайте с помощью метода value_counts количество наблюдений для каждого значения целево
#переменной Class
t=df['Class'].value_counts()
t
```

Out[15]:

```
0    284315
1       492
Name: Class, dtype: int64
```

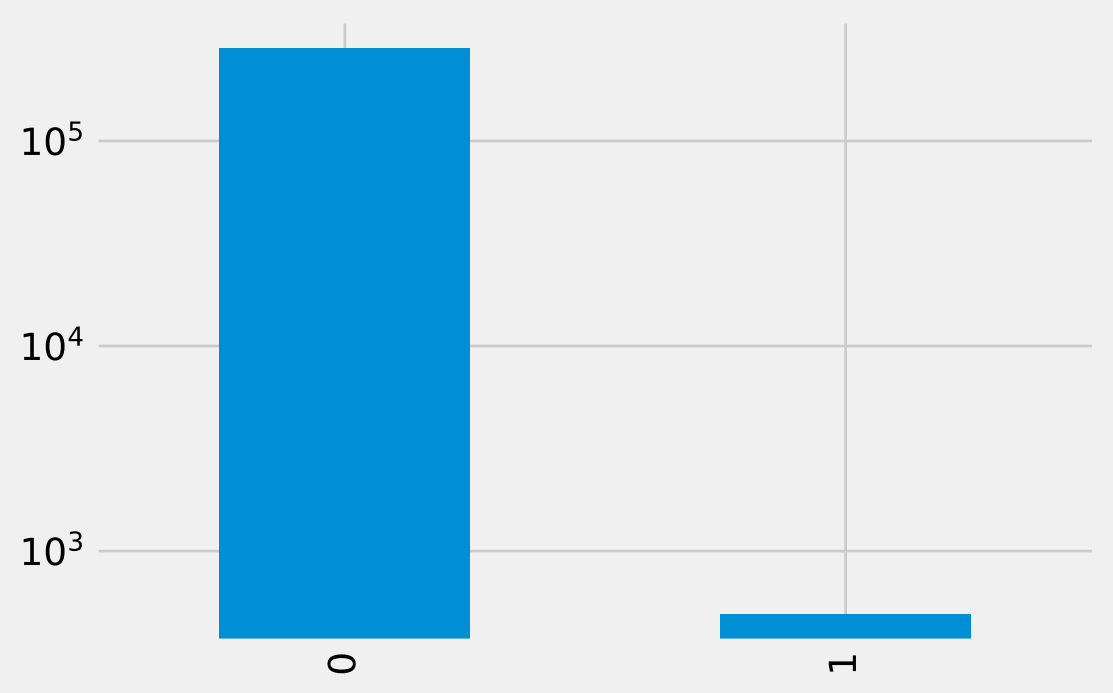
In [16]:

```
#и примените к полученным данным метод plot, чтобы построить столбчатую диаграмму.
df_class_info = pd.Series(t)
df_class_info.plot('bar')
plt.show()
```



In [17]:

```
#Затем постройте такую же диаграмму, используя логарифмический масштаб.
df_class_info.plot(kind='bar', logy=True)
plt.show()
```



In [18]:

```
#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.
v1_class1=df.set_index('Class')['V1'].filter(like='1', axis=0)
v1_class1=v1_class1.reset_index()
v1_class1=v1_class1.drop('Class', axis=1)
v1_class1.head(), v1_class1.count()
```

Out[18]:

```
(         V1
 0 -2.312227
 1 -3.043541
 2 -2.303350
 3 -4.397974
 4  1.234235, V1    492
 dtype: int64)
```

In [19]:

```
v1_class0=df.set_index('Class')['V1'].filter(like='0', axis=0)
v1_class0=v1_class0.reset_index()
v1_class0=v1_class0.drop('Class', axis=1)
v1_class0.head(), v1_class0.count()
```

Out[19]:

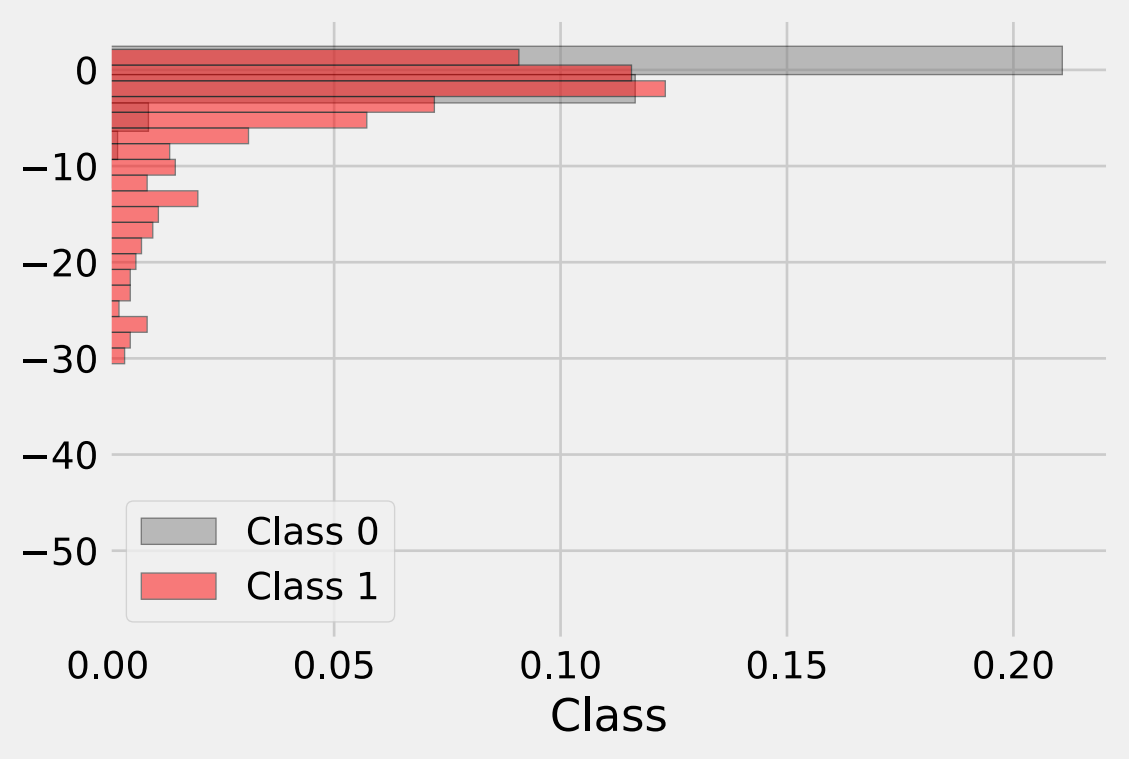
```
(         V1
 0 -1.359807
 1  1.191857
 2 -1.358354
 3 -0.966272
 4 -1.158233, V1    284315
 dtype: int64)
```

In [20]:

```
#Создайте легенду с двумя значениями: Class 0 и Class 1. Гистограмма обычных транзакций
#должна быть серого цвета, а мошеннических – красного. Горизонтальной оси дайте название Cl
plt.hist(v1_class0['V1'], bins=20, color='grey', edgecolor='black', density = True, orienta
plt.hist(v1_class1['V1'], bins=20, color='red', edgecolor='black', density = True, orientat
plt.plot()
plt.xlabel('Class')
plt.legend(labels=['Class 0', 'Class 1'])
```

Out[20]:

```
<matplotlib.legend.Legend at 0x9bea748>
```



#Задания на повторение

In [21]:

```
#Создать одномерный массив Numpy под названием a из 12 последовательных целых чисел от 12 д
a=np.arange(12, 24)
a
```

Out[21]:

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

In [22]:

```
#Создать 5 двумерных массивов разной формы из массива a. Не использовать в аргументах метод
a1=a.reshape(2, 6)
a2=a.reshape(3, 4)
a3=a.reshape(6, 2)
a4=a.reshape(4, 3)
a5=a.reshape(12, 1)
print(a1)
print(a2)
print(a3)
print(a4)
print(a5)
```

```
[[12 13 14 15 16 17]
 [18 19 20 21 22 23]]
[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
[[12 13]
 [14 15]
 [16 17]
 [18 19]
 [20 21]
 [22 23]]
[[12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]
[[12]
 [13]
 [14]
 [15]
 [16]
 [17]
 [18]
 [19]
 [20]
 [21]
 [22]
 [23]]
```

In [23]:

```
#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 ст
a1=a.reshape(2, -1)
a2=a.reshape(3, -1)
a3=a.reshape(-1, 2)
a4=a.reshape(-1, 3)
a5=a.reshape(12, -1)
print(a1)
print(a2)
print(a3)
print(a4)
print(a5)
```

```
[[12 13 14 15 16 17]
 [18 19 20 21 22 23]]
[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
[[12 13]
 [14 15]
 [16 17]
 [18 19]
 [20 21]
 [22 23]]
[[12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]
[[12]
 [13]
 [14]
 [15]
 [16]
 [17]
 [18]
 [19]
 [20]
 [21]
 [22]
 [23]]
```

In [24]:

```
#Можно ли массив Numpy, состоящий из одного столбца и 12 строк, назвать одномерным?
#Чтобы ответить на этот вопрос, сравним два массива: x и y
#Массив будет изначально создан, как одномерный
#Массив y будет создан как reshape от двумерного
x=np.arange(12, 24)
matr=x.reshape(3,4)
y=matr.reshape(12,1)
print(x)
print(y)
#Из примера четко видно, что одномерный массив и массив Numpy, состоящий из одного
#столбца и 12 строк, различаются. Последний, в отличии от первого, имеет номер строки
#и номер столбца, выводится как двумерный массив с дополнительными []. Таким образом,
#массив Numpy, состоящий из одного столбца и 12 строк, НЕЛЬЗЯ назвать одномерным.
```

```
[12 13 14 15 16 17 18 19 20 21 22 23]
[[12]
 [13]
 [14]
 [15]
 [16]
 [17]
 [18]
 [19]
 [20]
 [21]
 [22]
 [23]]
```

In [25]:

```
#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
b=np.random.randn(3,4)
b
```

Out[25]:

```
array([[ 0.27963492, -0.79108324, -0.60295697, -0.49064965],
       [ 0.72789775, -0.78512819, -1.43685167,  0.64765149],
       [-0.61963127, -1.82998334, -0.36956164,  0.23481839]])
```

In [26]:

```
#Получить из этого массива одномерный массив с таким же атрибутом size, как и исходный масс
print(b.size)
b=b.flatten()
print(b.size)
b
```

```
12
12
```

Out[26]:

```
array([ 0.27963492, -0.79108324, -0.60295697, -0.49064965,  0.72789775,
       -0.78512819, -1.43685167,  0.64765149, -0.61963127, -1.82998334,
       -0.36956164,  0.23481839])
```

In [27]:

```
#Создать массив a, состоящий из целых чисел, убывающих от 20 до 0 невключительно с интервал
a=np.arange(20, 0, -2)
a
```

Out[27]:

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

In [28]:

```
#Создать массив b, состоящий из 1 строки и 10 столбцов: целых чисел, убывающих от 20 до 1 н
b=np.arange(20, 1, -2).reshape(1,10)
b
```

Out[28]:

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

In [29]:

```
#В чем разница между массивами a и b?
#Разница между массивами a и b в том, что a - одномерный, a b - двумерный
```

In [30]:

```
#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.
a = np.zeros((2, 2))
a
```

Out[30]:

```
array([[0., 0.],
       [0., 0.]])
```

In [31]:

```
b = np.zeros((3, 2))+1
b
```

Out[31]:

```
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

In [32]:

```
v = np.vstack((a, b))
v
```

Out[32]:

```
array([[0., 0.],
       [0., 0.],
       [1., 1.],
       [1., 1.],
       [1., 1.]])
```

In [33]:

```
a.shape, b.shape, v.shape
```

Out[33]:

```
((2, 2), (3, 2), (5, 2))
```

In [34]:

```
#Итоговый массив v должен иметь атрибут size, равный 10.
v.size
```

Out[34]:

```
10
```

In [35]:

```
#Создать одномерный массив a, состоящий из последовательности целых чисел от 0 до 12.
a=np.arange(0, 12)
a
```

Out[35]:

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

In [36]:

```
#Поменять форму этого массива, чтобы получилась матрица A (двумерный массив Numpy), состоящ
A=a.reshape(4,3)
A
```

Out[36]:

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

In [37]:

```
#Получить матрицу At путем транспонирования матрицы A.
At=A.T
At
```

Out[37]:

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

In [38]:

```
#Получить матрицу B, умножив матрицу A на матрицу At с помощью матричного умножения.
B = np.dot(A, At)
B
```

Out[38]:

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

In [39]:

```
#Какой размер имеет матрица B?
B.shape
```

Out[39]:

```
(4, 4)
```

In [40]:

```
#Получится ли вычислить обратную матрицу для матрицы B и почему?
B_inv=np.linalg.inv(B)
B_inv
#Вычислить обратную матрицу для матрицы B не получится, т.к. это вырожденная (сингулярная)
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-40-8b7568013c09> in <module>
      1 #Получится ли вычислить обратную матрицу для матрицы B и почему?
----> 2 B_inv=np.linalg.inv(B)
      3 B_inv
      4 #Вычислить обратную матрицу для матрицы B не получится, т.к. это выр
ожденная (сингулярная) матрица

d:\Users\Tim\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in inv(a)
    549     signature = 'D->D' if isComplexType(t) else 'd->d'
    550     extobj = get_linalg_error_extobj(_raise_linalgerror_singular)
--> 551     ainv = _umath_linalg.inv(a, signature=signature, extobj=extobj)
    552     return wrap(ainv.astype(result_t, copy=False))
    553

d:\Users\Tim\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in _raise_li
nalgerror_singular(err, flag)
     95
     96 def _raise_linalgerror_singular(err, flag):
---> 97     raise LinAlgError("Singular matrix")
     98
     99 def _raise_linalgerror_nonposdef(err, flag):

LinAlgError: Singular matrix
```

In [41]:

```
#Инициализируйте генератор случайных числе с помощью объекта seed, равного 42.
np.random.seed(42)
```

In [42]:

```
#Создайте одномерный массив c, составленный из последовательности 16-ти случайных целых чис
c=np.random.randint(0, 16,16)
c
```

Out[42]:

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

In [43]:

```
#Поменяйте его форму так, чтобы получилась квадратная матрица C.
C=c.reshape(4,4)
C
```

Out[43]:

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

In [44]:

```
#Получите матрицу D, поэлементно прибавив матрицу B из предыдущего вопроса к матрице C, умн
D=B+C*10
D
```

Out[44]:

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

In [45]:

```
#Вычислите определитель для D.
d=np.linalg.det(D)
d
```

Out[45]:

```
-28511999.999999944
```

In [46]:

```
#Вычислите ранг для D.
r=np.linalg.matrix_rank(D)
r
```

Out[46]:

```
4
```

In [47]:

```
#Вычислите обратную матрицу D_inv для D.
D_inv=np.linalg.inv(D)
D_inv
```

Out[47]:

```
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]])
```

In [48]:

```
#Приравняйте к нулю отрицательные числа в матрице D_inv, а положительные — к единице.
#Убедитесь, что в матрице D_inv остались только нули и единицы.
D_inv=np.where(D_inv<0, 0,1)
D_inv
```

Out[48]:

```
array([[1, 1, 1, 0],
       [0, 0, 0, 1],
       [0, 0, 0, 1],
       [1, 0, 1, 0]])
```

In [49]:

```
#С помощью функции numpy.where, используя матрицу D_inv в качестве маски, а матрицы B и C -
#источников данных, получите матрицу E размером 4x4. Элементы матрицы E, для которых соотве
#элемент матрицы D_inv равен 1, должны быть равны соответствующему элементу матрицы B, а эл
#матрицы E, для которых соответствующий элемент матрицы D_inv равен 0, должны быть равны
#соответствующему элементу матрицы C.
print(B)
print(C)
E=np.where(D_inv==1, B, C)
E
```

```
[[  5  14  23  32]
 [ 14  50  86 122]
 [ 23  86 149 212]
 [ 32 122 212 302]]
[[ 6  3 12 14]
 [10  7 12  4]
 [ 6  9  2  6]
 [10 10  7  4]]
```

Out[49]:

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

In [ ]:

```

```