In [1]:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import scipy.optimize
from tabulate import tabulate

from matplotlib import pyplot as plt
from IPython.display import display
```

In [2]:

```
mpl.rcParams['axes.grid'] = True
mpl.rcParams['axes.axisbelow'] = True
N = 1000000
```

Задание 1

In [3]:

```
X = np.array([69, 74, 68, 70, 72, 67, 66, 70, 76, 68, 72, 79, 74, 67, 66, 71, 74
, 75, 75, 76])
Y = np.array([153, 175, 155, 135, 172, 150, 115, 137, 200, 130, 140, 265, 185, 1
12, 140,  150, 165, 185, 210, 220])
```

A. Найти среднее, медиану и моду величины X

In [4]:

```
print('Среднее X -', np.mean(X))
print('Медиана X -', np.median(X))
_, indices, counts = np.unique(X, return_counts=True, return_index=True)
print('Мода X -', X[indices[np.argmax(counts)]])
```

```
Среднее X - 71.45
Медиана X - 71.5
Мода X - 74
```

B. Найти дисперсию Y

In [5]:

```
print('Дисперсия Y -', np.var(Y))
```

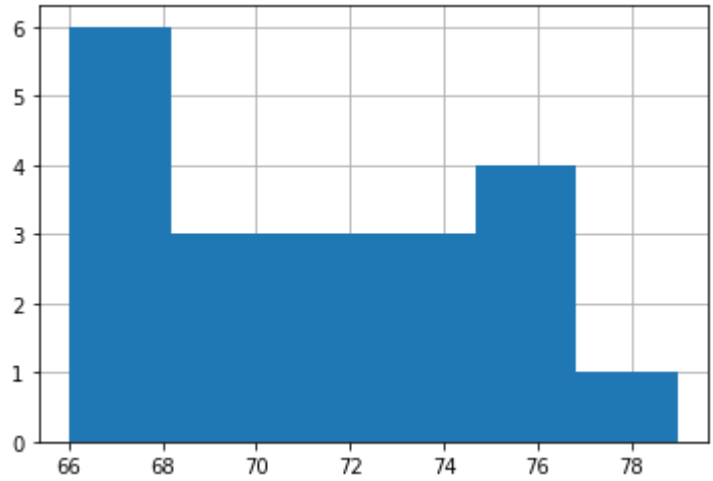
```
Дисперсия Y - 1369.2099999999998
```

C. Построить график нормального распределения для X

Данные в X распределены не нормально:

In [6]:

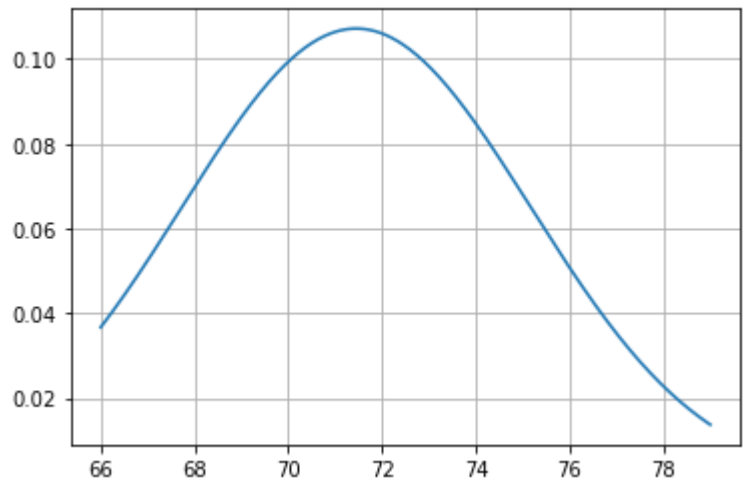
```
plt.hist(X, bins=6)
pass
```



Можно построить нормальное распределение с тем же мат. ожиданием и дисперсией

In [7]:

```
F = lambda s, m: (lambda x: 1 / (s * np.sqrt(2 * np.pi)) * np.exp(-(x - m)**2 /
(2 * s**2)))
F_n = F(np.sqrt(np.var(X)), np.mean(X))


x = np.linspace(min(X), max(X), N)
f_n = F_n(x)
plt.plot(x, f_n)
pass
```



D. Найти вероятность того, что возраст больше 80

In [8]:

```
np.count_nonzero(X > 80) / len(X)
```

Out[8]:

```
0.0
```

E. Найти двумерное мат. ожидания и ковариационную матрицу для этих двух величин

In [9]:

```
print('Мат. ожидание -', np.mean([X, Y], axis=1))
print('Матрица ковариции -\n', np.cov([X, Y]))
```

```
Мат. ожидание - [ 71.45 164.7 ]
Матрица ковариции -
 [[  14.57631579  128.87894737]
 [ 128.87894737 1441.27368421]]
```

F. Определить корреляцию между X и Y

In [10]:

```
np.corrcoef(X, Y)[0,1]
```

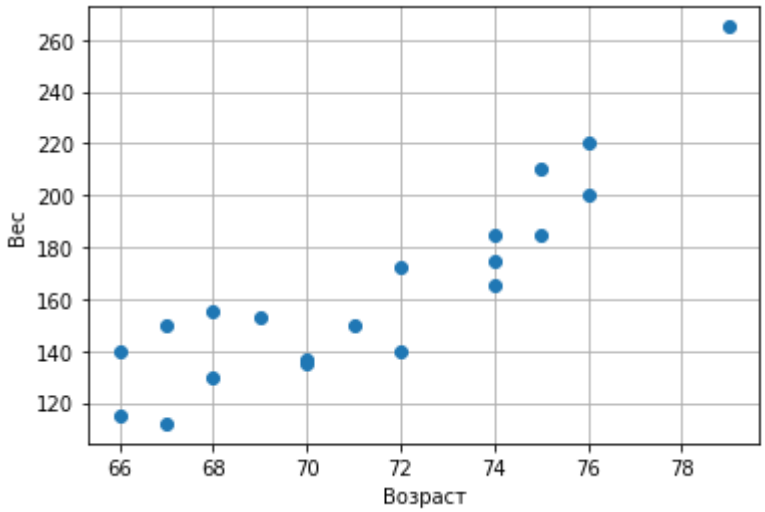
Out[10]:

```
0.8891701351748048
```

G. Построить диаграмму рассеяния, отображающая зависимость между возрастом и весом

In [11]:

```
plt.scatter(X, Y)
plt.xlabel('Возраст')
plt.ylabel('Вес')
pass
```



Задание 2

In [12]:

```
Z = np.array([[17, 17, 12], [11, 9, 13], [11, 8, 19]])
```

In [13]:

```
m = np.cov(Z.T)
display(tabulate(m, headers=['X_1', 'X_2', 'X_3'], showindex=['X_1', 'X_2', 'X_
3'], tablefmt='html'))
print('Обобщенная дисперсия -', np.linalg.det(m))
```

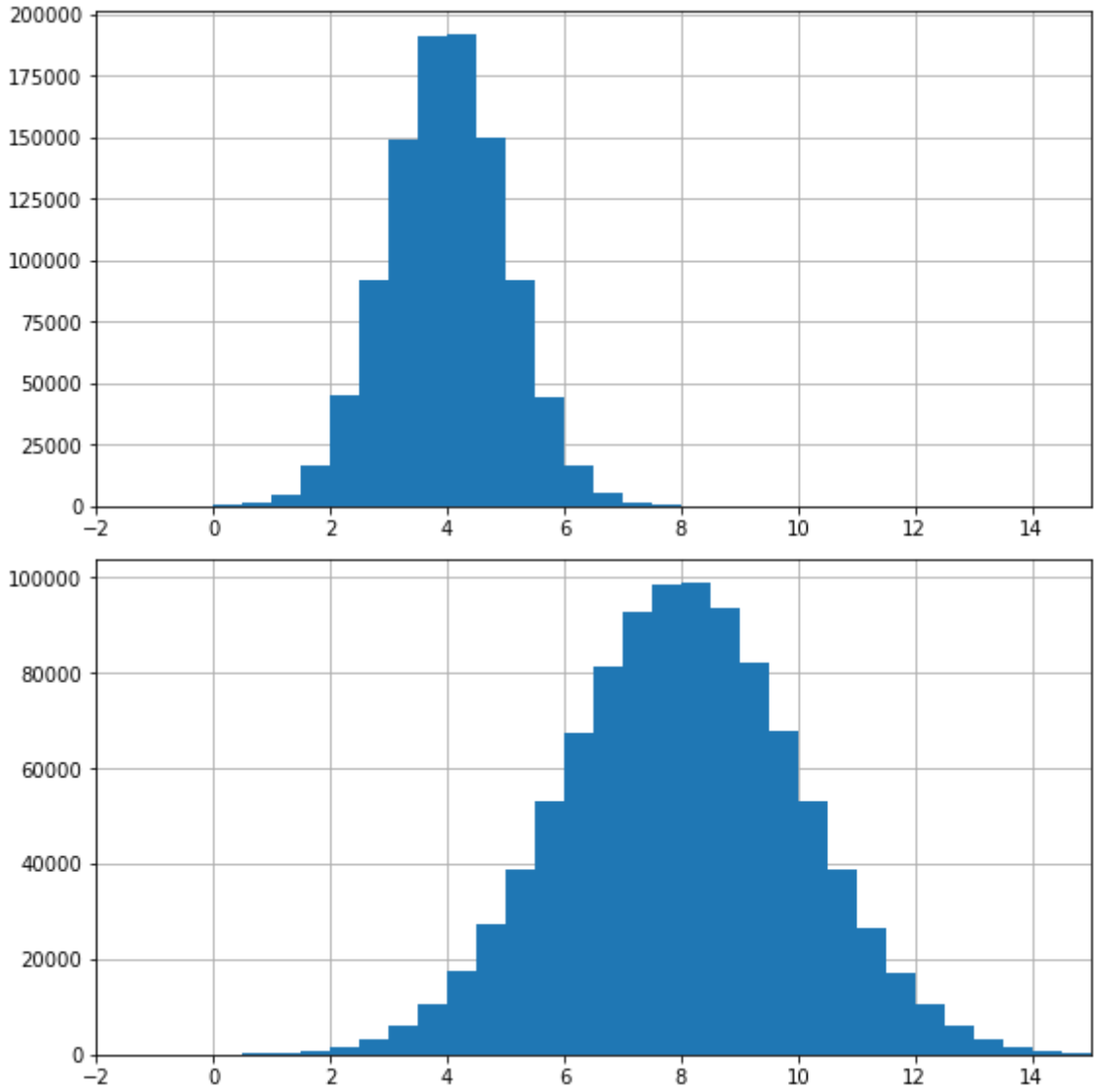
	X_1	X_2	X_3
X_1	12	17	-8
X_2	17	24.3333	-12.8333
X_3	-8	-12.8333	14.3333

```
Обобщенная дисперсия - -1.565414464721469e-14
```

Задание 3

In [14]:

```
N_a = np.random.normal(4, 1, N)
N_b = np.random.normal(8, 2, N)

fig, [ax1, ax2] = plt.subplots(2, 1, figsize=(8, 8))
ax1.hist(N_a, bins=34, range=(-2, 15))
ax2.hist(N_b, bins=34, range=(-2, 15))
ax1.set_xlim(-2, 15)
ax2.set_xlim(-2, 15)
fig.tight_layout()
```



Вероятность выпадения какого-то конкретного числа (в настоящем непрерывном распределении) равна 0

In [15]:

```
indices = 5, 6, 7
t = [[np.count_nonzero(N_a == x) / N, np.count_nonzero(N_b == x) / N] for x in i
ndices]
display(tabulate(t, tablefmt='html', headers=['N_a', 'N_b'], showindex=indices))
```

	N_a	N_b
5	0	0
6	0	0
7	0	0

Но можно дискретизировать с шагом 1:

In [16]:

```
t = [[np.count_nonzero(np.abs(N_a - x) < 1) / N, np.count_nonzero(np.abs(N_b - x
) < 1) / N] for x in indices]
display(tabulate(t, tablefmt='html', headers=['N_a', 'N_b'], showindex=indices))
```

	N_a	N_b
5	0.477953	0.137033
6	0.157381	0.240907
7	0.022452	0.339898

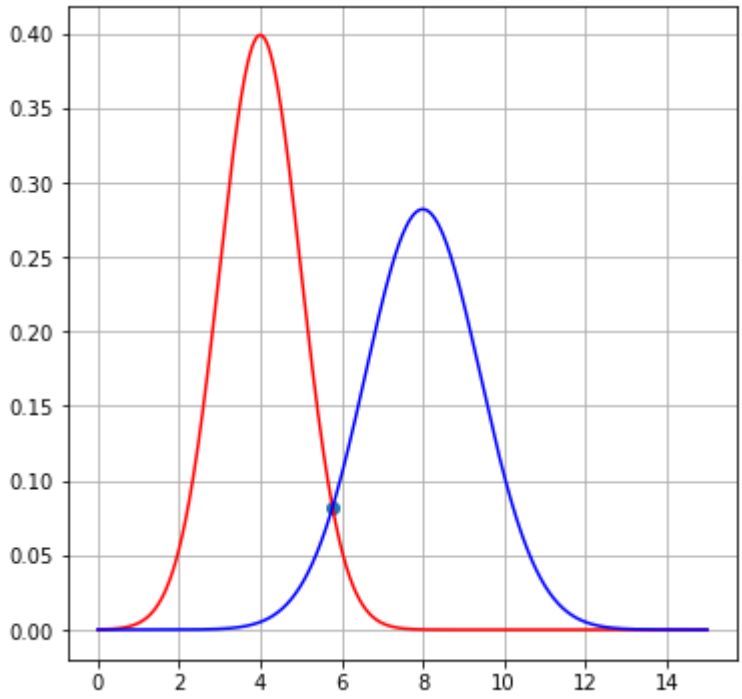
Вычислить значение, которое может быть сгенерировано обеими распределениями с равной вероятностью можно, приравняв функции плотности

In [17]:

```
x = np.linspace(0, 15, N)
s_1 = np.sqrt(1)
s_2 = np.sqrt(2)
m_1 = 4
m_2 = 8

F_1 = F(s_1, m_1)
F_2 = F(s_2, m_2)

f_1 = F_1(x)
f_2 = F_2(x)

plt.figure(figsize=(6, 6))
plt.plot(x, f_1, 'r')
plt.plot(x, f_2, 'b')

res = scipy.optimize.root(lambda x: F_1(x) - F_2(x), 5)
display(res)
plt.scatter(res.x, [F_1(res.x)])
print('Результат:', res.x[0])
```

```
    fjac: array([[-1.]])
     fun: array([6.9388939e-17])
 message: 'The solution converged.'
    nfev: 8
     qtf: array([-1.32810429e-13])
       r: array([0.23720977])
  status: 1
 success: True
       x: array([5.77808743])
```

Результат: 5.778087431072663



In [ ]:

```

```