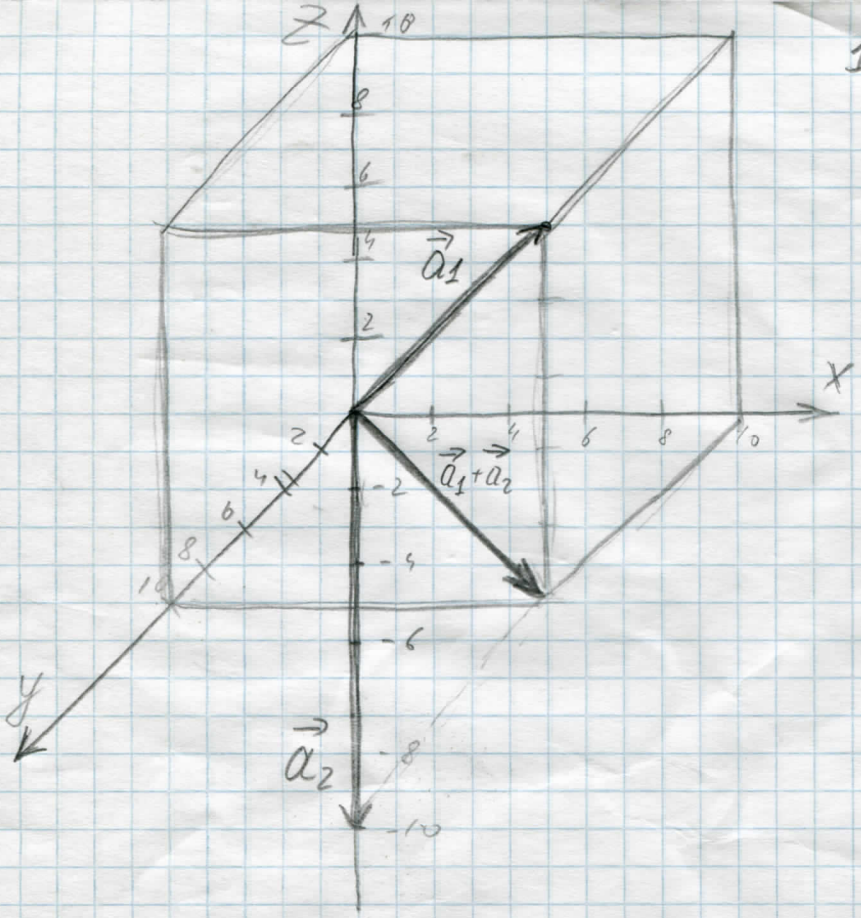
1 задание



$\vec{a}_1 = (10; 10, 10)$; $\vec{a}_2 = (0, 0, -10)$

$\vec{a}_1 + \vec{a}_2 = (10+0, 10+0, 10+(-10)) =$

$\vec{a}_1 + \vec{a}_2 = (10, 10, 0)$

In [1]:

```
#
#1. Задание
#Даны два вектора в трехмерном пространстве: (10,10,10) и (0,0,-10)
#Напишите код на Python, реализующий расчет длины вектора, заданного его координатами. (в п
```

In [2]:

```
# 1. импортируем библиотеки

%matplotlib inline
import numpy as np
```

In [3]:

```
# определим функцию
def sum_vect (a, b):
    sum_ab=[]
    if np.shape(a) != np.shape(b):
        return f'Ошибка - векторы {a} и {b} имеют разную длину'
    else:
        for i,x in enumerate(a):
            sum_ab.append(a[i] + b[i])
    return f'Сумма векторов {a} и {b} равна {sum_ab}'

print (sum_vect([10,10,10,10],[0,0,-10]))
print (sum_vect([10,10,10],[0,0,-10]))
```

```
Ошибка - векторы [10, 10, 10, 10] и [0, 0, -10] имеют разную длину
Сумма векторов [10, 10, 10] и [0, 0, -10] равна [10, 10, 0]
```

In [1]:

```
#2. Задание (на листочке)
# Почему прямые не кажутся перпендикулярными? (см.ролик)

# На листочке делать не стал:
# в блокноте нагляднее!
```

In [2]:

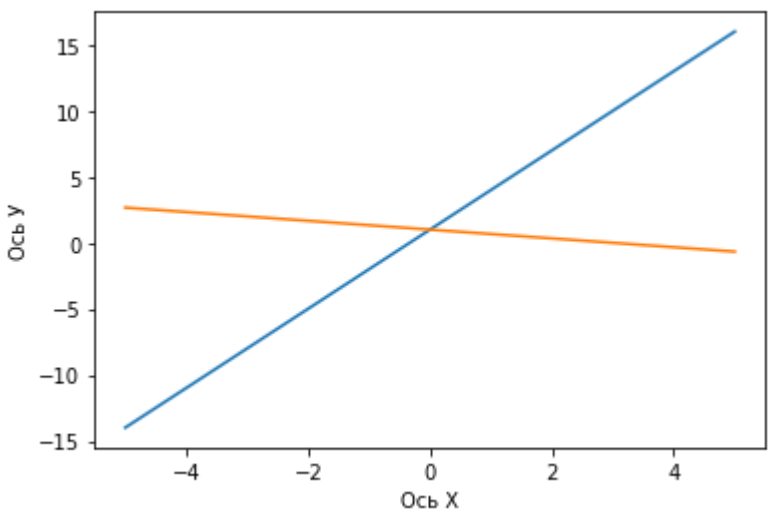
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:

```
# в примере линии не перпендикулярны
x=np.linspace(-5, 5, 21)
y=3*x+1
y2=(-1/3)*x+1

plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.plot(x, y, x, y2)
```

Out[3]:

```
[<matplotlib.lines.Line2D at 0x82cb400>,
 <matplotlib.lines.Line2D at 0x82cb430>]
```

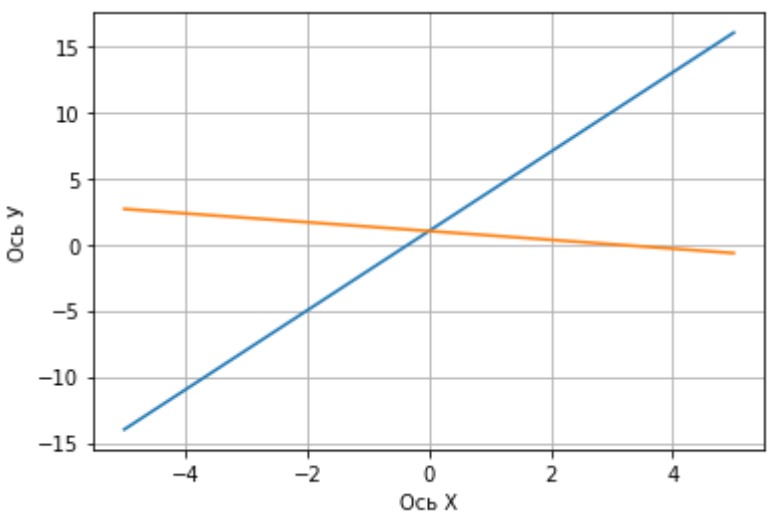


In [4]:

```
# если нарисовать сетку, то всё станет понятно:
# они не перпендикулярны, потому, что масштабы
# осей x и y не одинаковые:

x=np.linspace(-5, 5, 21)
y=3*x+1
y2=(-1/3)*x+1

plt.xlabel("Ось X")
plt.ylabel("Ось Y")
plt.grid(True)
plt.plot(x, y, x, y2)
```

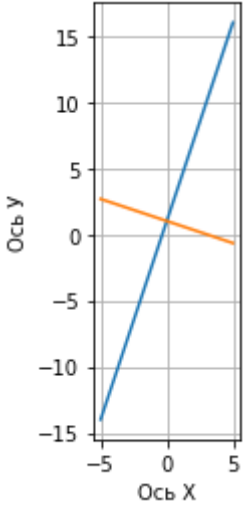
Out[4]:

```
[<matplotlib.lines.Line2D at 0x8361ca0>,
 <matplotlib.lines.Line2D at 0x8361d90>]
```



In [5]:

```
# Исправим масштаб осей, и получим перпендикуляр:

x=np.linspace(-5, 5, 21)
y=3*x+1
y2=(-1/3)*x+1

plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.grid(True)
plt.plot(x, y, x, y2)
plt.gca().set_aspect('equal')
```

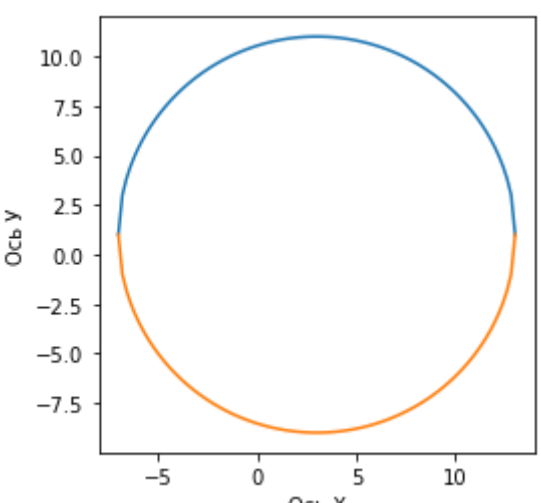


In [1]:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

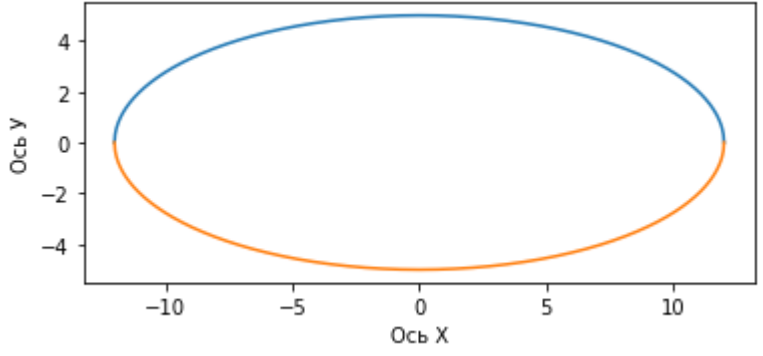
In [2]:

```
#строим график окружности
r=10
x0= 3
y0= 1

x=np.linspace(-r+x0, r+x0, 100)
y1=(r**2 - (x-x0)**2)**0.5 + y0
y2=-(r**2 - (x-x0)**2)**0.5 + y0
plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.plot(x, y1, x, y2)
plt.gca().set_aspect('equal')
```



In [3]:

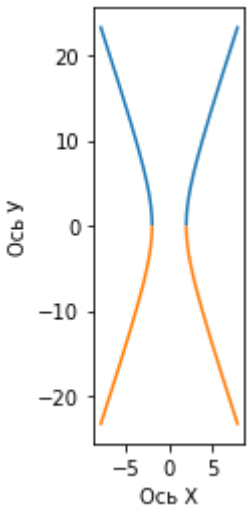
```
#строим график эллипса

a= 12
b= 5

x=np.linspace(-a, a, 1000)
y1 = ((1 - (x**2/a**2))*b**2)**0.5
y2 = -((1 - (x**2/a**2))*b**2)**0.5
plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.plot(x, y1, x, y2)
plt.gca().set_aspect('equal')
```



In [4]:

```
#строим график гиперболы

a= 2
b= 6

x=np.linspace(-a*5, -a, 10000) + np.linspace(a, a*5, 10000)
y1 = (((x**2)/(a**2)-1)*(b**2))**0.5
y2 = -(((x**2/a**2)-1)*(b**2))**0.5
plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.plot(x, y1, x, y2)
plt.gca().set_aspect('equal')
```

```
<ipython-input-4-f1b36598bbfc>:7: RuntimeWarning: invalid value encountered
in sqrt
  y1 = (((x**2)/(a**2)-1)*(b**2))**0.5
<ipython-input-4-f1b36598bbfc>:8: RuntimeWarning: invalid value encountered
in sqrt
  y2 = -(((x**2/a**2)-1)*(b**2))**0.5
```



4 Задание

① Плоскость $Ax + By + Cz + D = 0$

Угол наклона определяется коэффициентами

A, B, C.

Параллельная плоскость будет проходить через начало координат, когда

$$A \cdot 0 + B \cdot 0 + C \cdot 0 + D = 0 \Rightarrow D = 0$$

т.е ответ: $Ax + By + Cz = 0$

② Прямая: $\frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1} = \frac{z-z_1}{z_2-z_1}$

принадлежит плоскости: $A_1x+B_1y+C_1z+D_1=0$

тогда, и только тогда когда выполняются условия:

$$\begin{cases} A_1x_1 + B_1y_1 + C_1z_1 + D_1 = 0 \\ A_1x_2 + B_1y_2 + C_1z_2 + D_1 = 0 \end{cases}$$

т.е. когда верно равенство:

$$A_1(x_1-x_2) + B_1(y_1-y_2) + C_1(y_1-y_2) = 0$$

In [1]:

```
#Нарисуйте трехмерный график двух параллельных плоскостей.
```

In [2]:

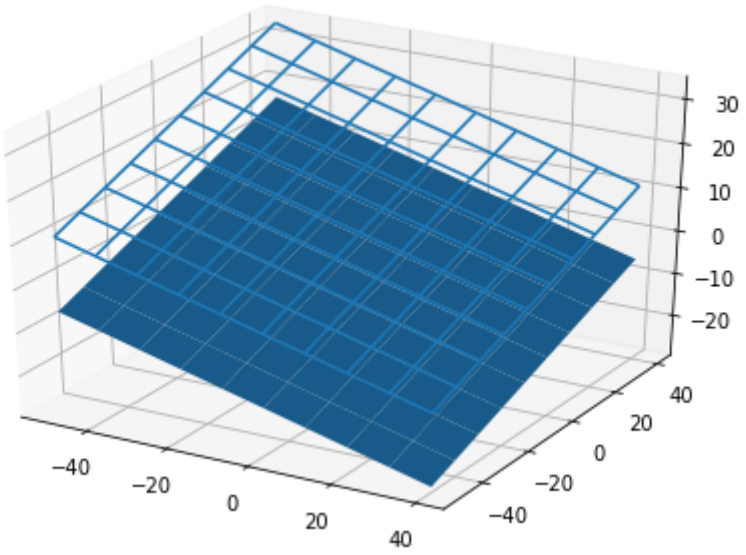
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:

```
def f_plane(a, b, c, d):
    return (d-a*X-b*Y)/c
```

In [24]:

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
fig = figure()
ax = Axes3D(fig)
X = np.arange(-50, 50, 10)
Y = np.arange(-50, 50, 10)
X, Y = np.meshgrid(X, Y)


Z2 = f_plane(1, -1, 4, 45)
Z1 = f_plane(1, -1, 4, -23)
ax.plot_wireframe(X, Y, Z2)
ax.plot_surface(X, Y, Z1)
show()
```



In [ ]:

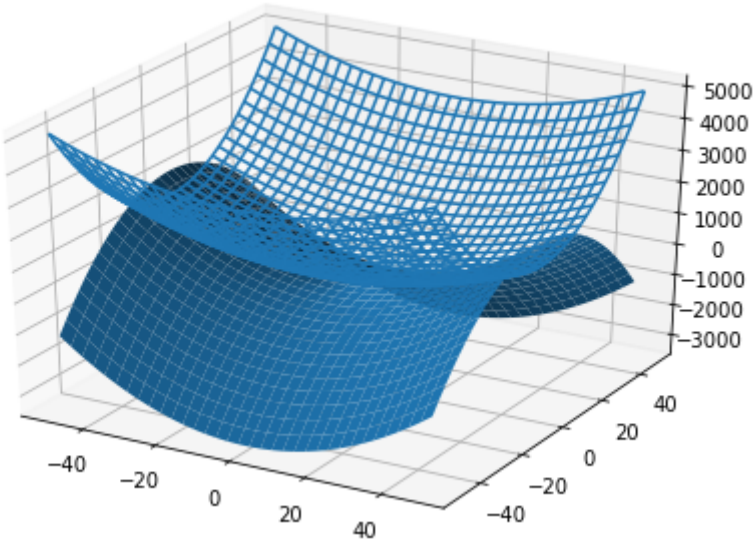
```
# Нарисуйте трехмерный график двух любых поверхностей второго порядка.
```

In [26]:

```
def f_parab(a, b, c, d):
    return (a*X**2+b*Y**2-d)/c
def f_giperb(a, b, c, d):
    return (X**2/a**2-Y**2/b**2-d)/c
```

In [54]:

```
fig = figure()
ax = Axes3D(fig)
X = np.arange(-50, 50, 3)
Y = np.arange(-50, 50, 3)
X, Y = np.meshgrid(X, Y)


Z2 = f_parab(1, 3, 2, -300)
Z1 = f_giperb(0.6, 0.5,3,400)
ax.plot_wireframe(X, Y, Z2)
ax.plot_surface(X, Y, Z1)
show()
```



In [ ]:

```
```

In [1]:

```
#Нарисуйте график функции:
#для некоторых (2-3 различных) значений параметров k, a, b

# импортируем библиотеки
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [6]:

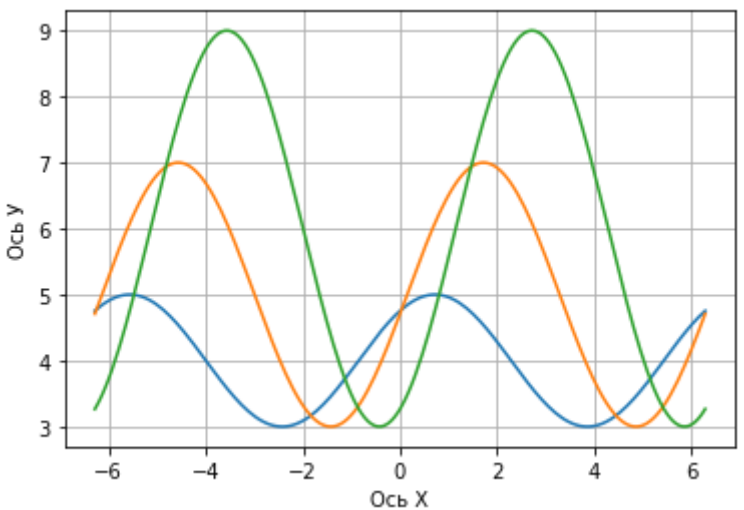
```
# зададим массив значений переменной x
x=np.linspace(-2*np.pi, 2*np.pi, 180)

# определим функцию
def f(k,a,b):
    return k*np.cos(x-a)+b
```

In [7]:

```
# Зададим коэффициенты:
k1 = 1
k2 = 2
k3 = 3
a1 = 7
a2 = 8
a3 = 9
b1 = 4
b2 = 5
b3 = 6


# рисуем график:
plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.grid(True)
plt.plot(x, f(k1,a1,b1), x, f(k2,a2,b2), x, f(k3,a3,b3))
#plt.show
```

Out[7]:

```
[<matplotlib.lines.Line2D at 0x572f400>,
 <matplotlib.lines.Line2D at 0x572f4f0>,
 <matplotlib.lines.Line2D at 0x572f5b0>]
```



In [1]:

```
# Напишите код, который будет переводить полярные координаты в декартовы.

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

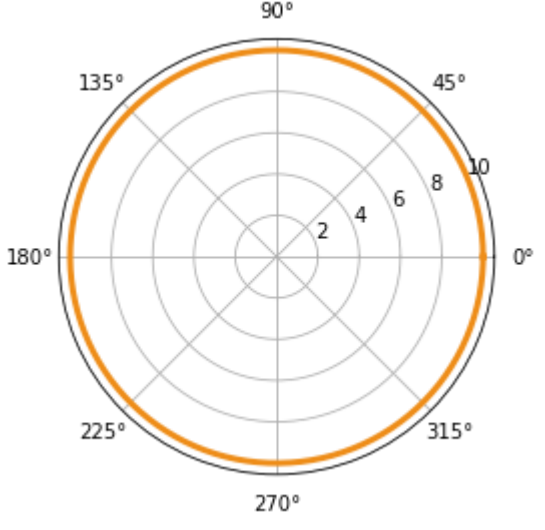
```
# определим функцию перевода полярных координат в декартовы
def f(r, alpha):
    xy=[0,0]
    xy[0] = r * np.cos(alpha)
    xy[1] = r * np.sin(alpha)
    return xy

print ("Переводим полярные координаты в декартовы")
r=float(input("введите длину вектора 'R' для координаты в полярной системе: "))
alph = np.radians(int(input("Введите угол в градусах для координаты в полярной системе: ")))
print
print(f"Данная точка в Декартовой системе будет иметь координаты {f(r,alph)}")
```

```
Переводим полярные координаты в декартовы
введите длину вектора 'R' для координаты в полярной системе: 8
Введите угол в градусах для координаты в полярной системе: 35
Данная точка в Декартовой системе будет иметь координаты [6.553216354311934,
4.588611490808368]
```

In [3]:

```
# Напишите код, который будет рисовать график окружности в полярных координатах.

print ("рисуем окружность в полярных координатах")
fig = plt.figure()
ax = fig.add_subplot(projection='polar')

radius=int(input("введите радиус окружности: "))
r = np.linspace(radius, radius, 360)
alpha = np.linspace(0, 2*np.pi, 360)

line, = ax.plot(alpha, r, color='#ee8d18', lw=3)

plt.show()
```

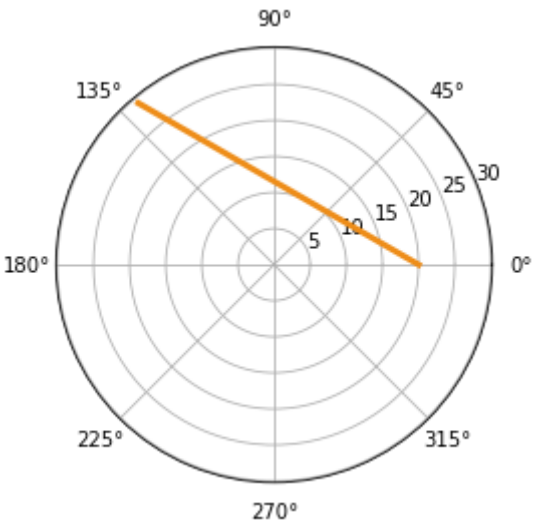
```
рисуем окружность в полярных координатах
введите радиус окружности: 10
```



In [4]:

```
# Напишите код, который будет рисовать график отрезка прямой линии в полярных координатах.
print ("Рисуем отрезок в полярных координатах")
fig = plt.figure()
ax = fig.add_subplot(projection='polar')

al = np.linspace(0, np.radians(130), 115)
r = 10/ np.cos(al-np.radians(60))
line, = ax.plot(al, r, color='#ee8d18', lw=3)

plt.show()
```

```
Рисуем отрезок в полярных координатах
```



In [1]:

```
# 4. Задание (в программе)
# exp(x) + x·(1 – y) = 1  => exp(x) + x – x·y - 1 = 0
# y = x2 – 1              => x2 - 1 - y = 0
```

In [2]:

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
```

In [3]:

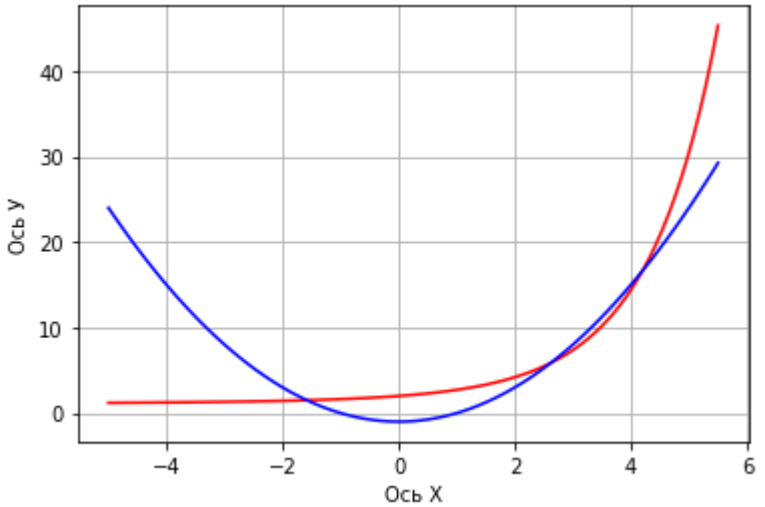
```
# определим функции
def f_exp(x):
    return (np.exp(x)+x-1)/x
def f_kv(x):
    return x*x-1
def f_sist(xy):
    x,y = xy
    return (np.exp(x) + x - x*y - 1, x*x - 1 - y)
```

In [4]:

```
x=np.linspace(-5, 5.5, 100)
# рисуем график:
plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.grid(True)
plt.plot(x, f_exp(x),'red', x, f_kv(x), 'blue')
```

Out[4]:

```
[<matplotlib.lines.Line2D at 0x8cf2d60>,
 <matplotlib.lines.Line2D at 0x8cf2d90>]
```



In [5]:

```
# решим систему уравнений в районе x=-1:
print (fsolve(f_sist, (-2, 5)))

# решим систему уравнений в районе x=2:
print (fsolve(f_sist, (2, 7)))

# решим систему уравнений в районе x=4:
print (fsolve(f_sist, (4, 20)))
```

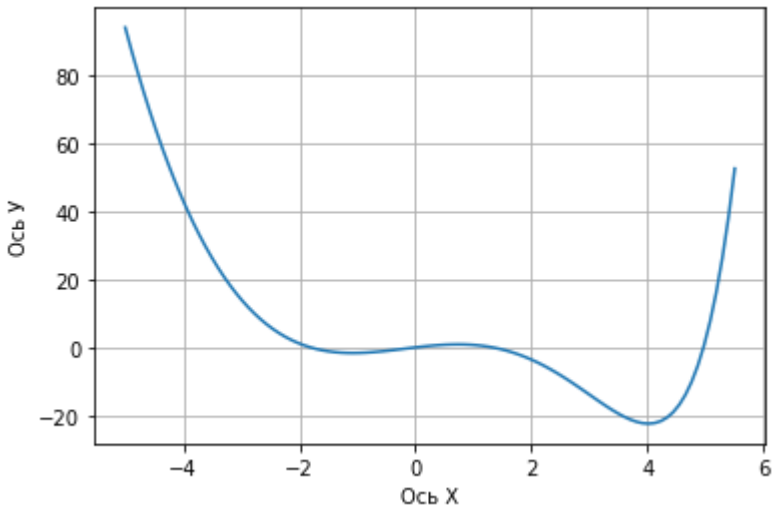
```
[-1.58183535  1.50220308]
[2.61814557 5.85468624]
[ 4.20010584 16.64088908]
```

In [6]:

```
def f_nerav(x):
    return (np.exp(x) - x**3 - x**2 + x - 1)
plt.xlabel("Ось X")
plt.ylabel("Ось y")
plt.grid(True)
plt.plot(x, f_nerav(x))
```

Out[6]:

```
[<matplotlib.lines.Line2D at 0x8dab940>]
```



In [7]:

```
print (fsolve(f_nerav, -2))
print (fsolve(f_nerav, -0.5))
print (fsolve(f_nerav, 1))
print (fsolve(f_nerav, 5))
```

```
[-1.80855201]
[9.43438893e-17]
[1.32669536]
[4.96119696]
```

In [8]:

```
# итого, неравенство выполняется при:
# x < -1.8 или 0 < x < 1.33 или x > 4.96
# ранее мы нашли корни системы уравнений:
# [-1.58,  1.50]
# [ 2.62,  5.85]
# [ 4.20, 16.64]
# Ни один из корней не входит в диапазон области определения системы уравнения и неравенств
# таким образом система не имеет решений
```