Задача 1.

Даны два вектора в трехмерном пространстве: (10,10,10) и (0,0,-10) Найдите их сумму. (на листочке) Напишите код на Python, реализующий расчет длины вектора, заданного его координатами. (в программе)

Решение

$\vec{c} = (10+0, 10+0, 10-10) = (10, 10, 0)$

In [1]:
```
from math import sqrt
def vec_length(a:())->float:
    """Calculate vector length
    param: a - tuple of vector components
    """
    a_square = 0
    for i in range(len(a)):
        a_square += a[i]**2
    return sqrt(a_square)

a = (10, 10, 0)
vec_length(a)
```

Out[1]: 14.142135623730951

Задача 2.

Почему прямые не кажутся перпендикулярными? (см.ролик)

Решение

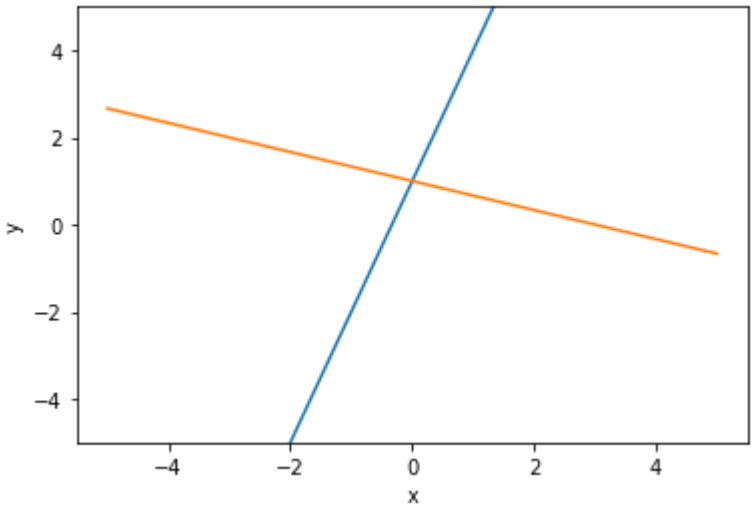
По причине разного масштаба делений осей, нужно выровнять длины

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5,5,21)
y = 3*x+1
y2 = (-1/3)*x+1
plt.plot(x,y)
plt.plot(x,y2)
plt.xlabel("x")
plt.ylabel("y")

plt.ylim(-5, 5)
```

Out[2]: (-5.0, 5.0)

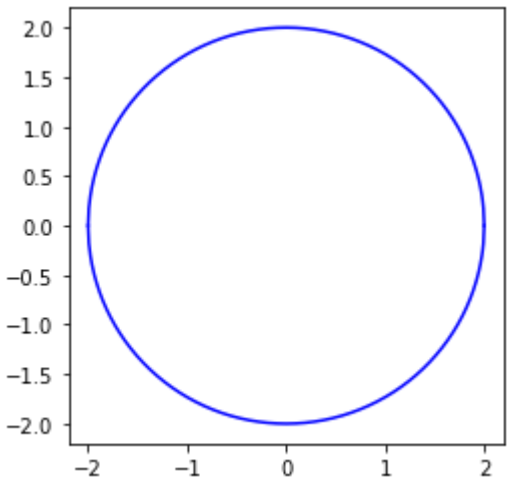


Задача 3.

Напишите код на Python, реализующий построение графиков: окружности, эллипса, гиперболы.

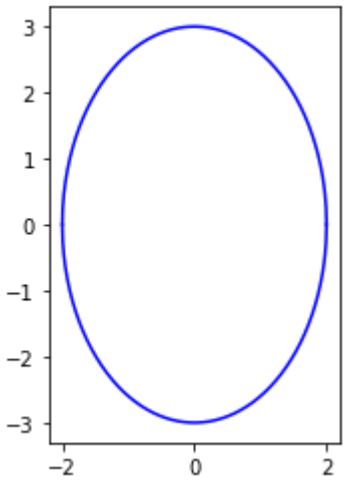
In [20]:

```
R = 2
x = np.linspace(-R, R, 512, endpoint=True)
y = np.sqrt(-(x*x) + R*R)
plt.plot(x, y, 'b')
plt.plot(x, -y,'b')
plt.gca().set_aspect('equal')
plt.show()
```



In [21]:

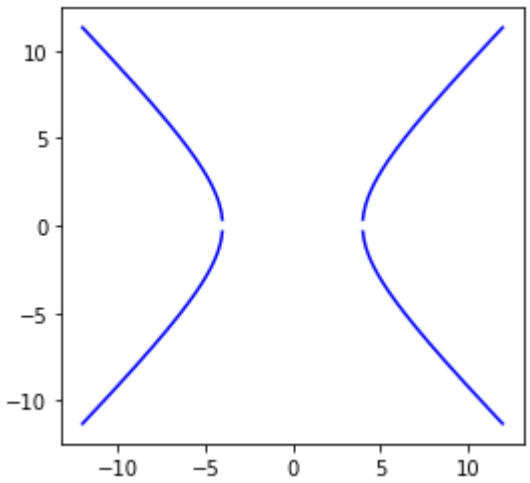
```
a = 2
b = 3
x = np.linspace(-a, a, 512, endpoint=True)
y = np.sqrt((1-x**2/a**2)*b**2)
plt.plot(x, y, 'b')
plt.plot(x, -y, 'b')
plt.gca().set_aspect('equal')
plt.show()
```



In [28]:

```
a = 4
b = 4
x = np.linspace(-a*3, a*3, 512, endpoint=True)
y = np.sqrt((x**2/a**2 - 1)*b**2)
plt.plot(x, y, 'b')
plt.plot(x, -y, 'b')
plt.gca().set_aspect('equal')
plt.show()
```

```
<ipython-input-28-b3965661a73d>:4: RuntimeWarning: invalid value encountered i
n sqrt
  y = np.sqrt((x**2/a**2 - 1)*b**2)
```



Задача 4.

1) Пусть задана плоскость: $A \cdot x + B \cdot y + C \cdot z + D = 0$ Напишите уравнение плоскости, параллельной данной и проходящей через начало координат.

Уравнение параллельной плоскости имеет теже коэффициенты при x, y, z и $D = 0$
$A \cdot x + B \cdot y + C \cdot z = 0$

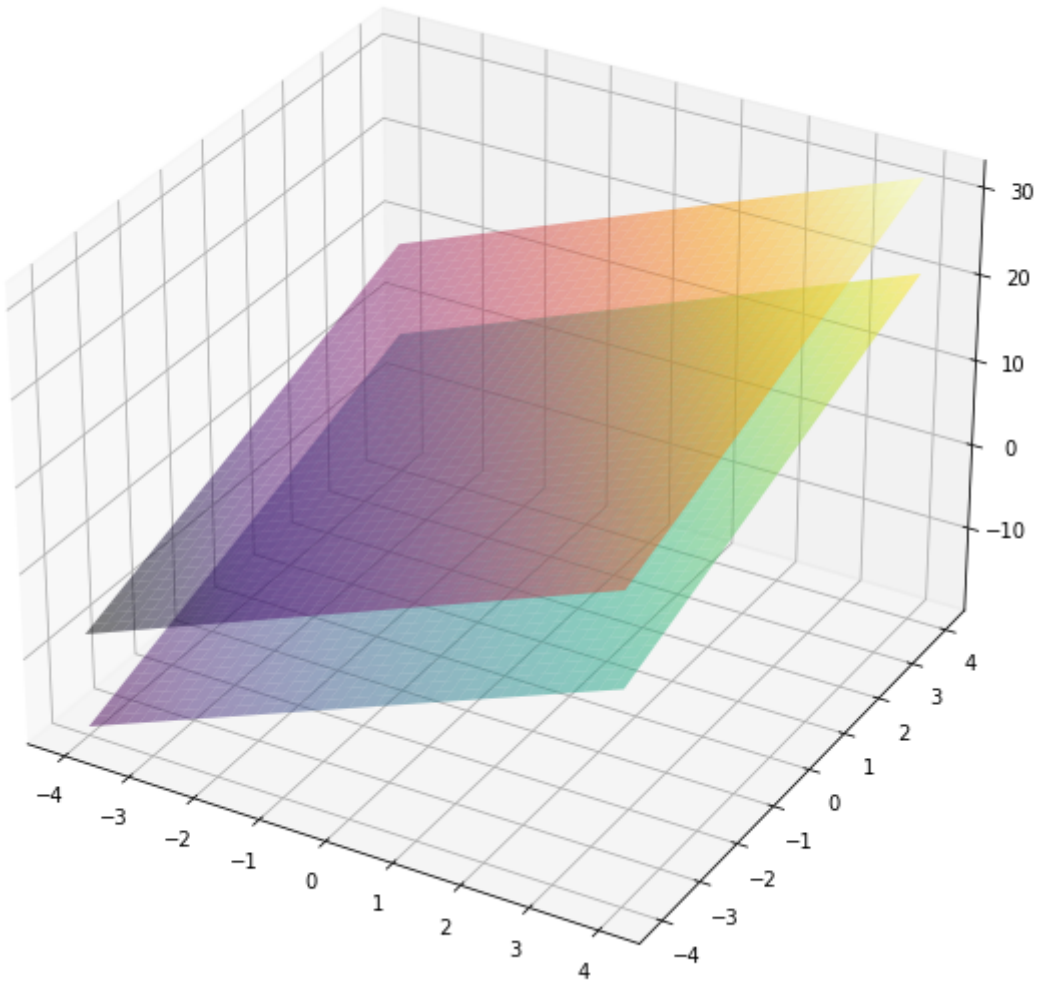
2) Пусть задана плоскость: $A_1x + B_1y + C_1z + D_1 = 0$ и прямая:
$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1}$ Как узнать, принадлежит прямая плоскости или нет?

Прямая принадлежит плоскости, если две её точки принадлежат этой плоскости
$A_1(x - x_1) + B_1(y - y_1) + C_1(z - z_1) + D_1 = 0$
$A_1(x - x_2) + B_1(y - y_2) + C_1(z - z_2) + D_1 = 0$

Задача 5.

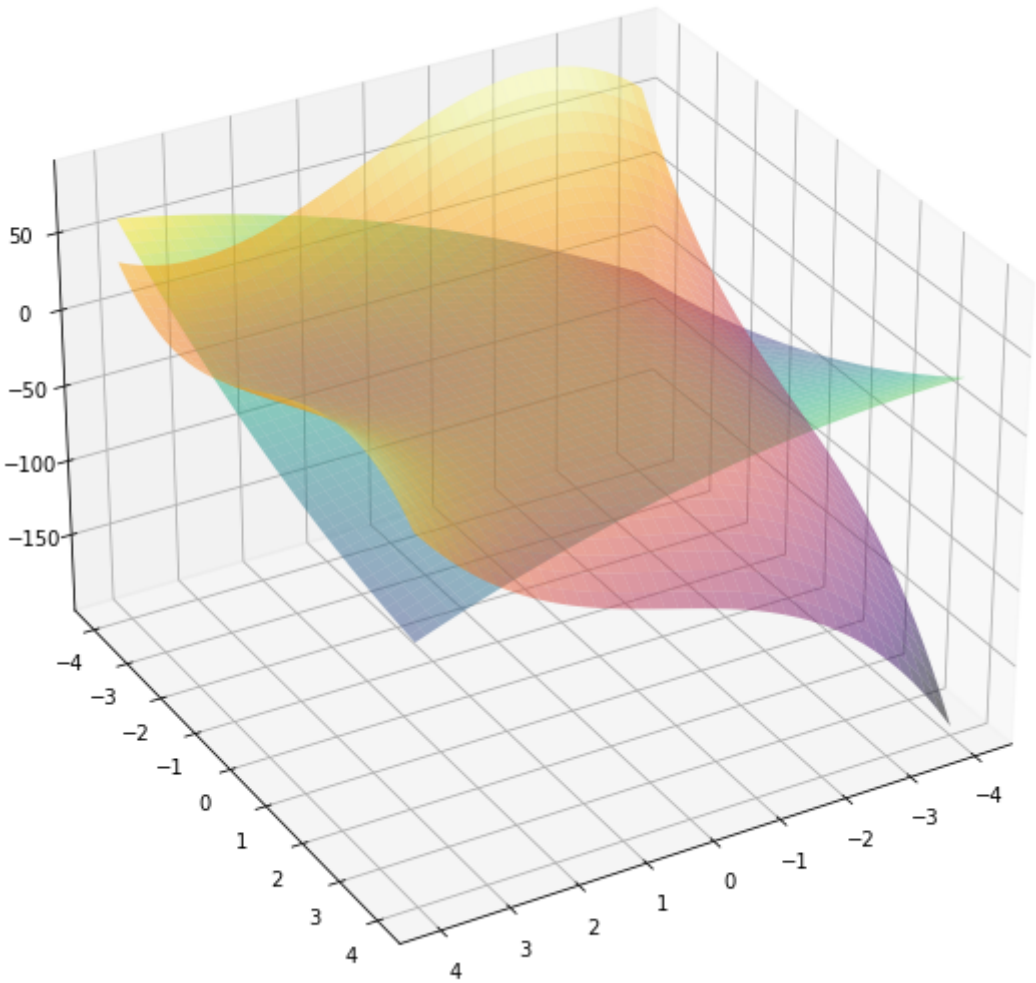
1) Нарисуйте трехмерный график двух параллельных плоскостей.

In [29]:
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
f = lambda x, y: 3*x + 2*y + 12
f1 = lambda x, y: 3*x + 2*y + 1
fig = plt.figure(figsize = (10, 10))
ax = fig.add_subplot(1, 1, 1, projection = '3d')
xval = np.linspace(-4, 4, 200)
yval = np.linspace(-4, 4, 200)
x, y = np.meshgrid(xval, yval)
z = f(x, y)
z1 = f1(x, y)
surf = ax.plot_surface(
x, y, z, alpha=0.5,
rstride = 5,
cstride = 5,
cmap = cm.inferno)
surf = ax.plot_surface(
x, y, z1, alpha=0.5,
rstride = 5,
cstride = 5,
cmap = cm.viridis)
```



2) Нарисуйте трехмерный график двух любых поверхностей второго порядка.

In [30]:
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
f = lambda x, y: x ** 3 - 3*x**2 - y ** 3 + 2*x**2 - 3*x + 4*x*y
f1 = lambda x, y: -x ** 2 + y ** 2 - 3*x*y + 4*x
fig = plt.figure(figsize = (10, 10))
ax = fig.add_subplot(1, 1, 1, projection = '3d')
ax.view_init(azim=60)
xval = np.linspace(-4, 4, 200)
yval = np.linspace(-4, 4, 200)
x, y = np.meshgrid(xval, yval)
z = f(x, y)
z1 = f1(x, y)
surf = ax.plot_surface(
x, y, z, alpha=0.5,
rstride = 5,
cstride = 5,
cmap = cm.inferno)
surf = ax.plot_surface(
x, y, z1, alpha=0.5,
rstride = 5,
cstride = 5,
cmap = cm.viridis)
```



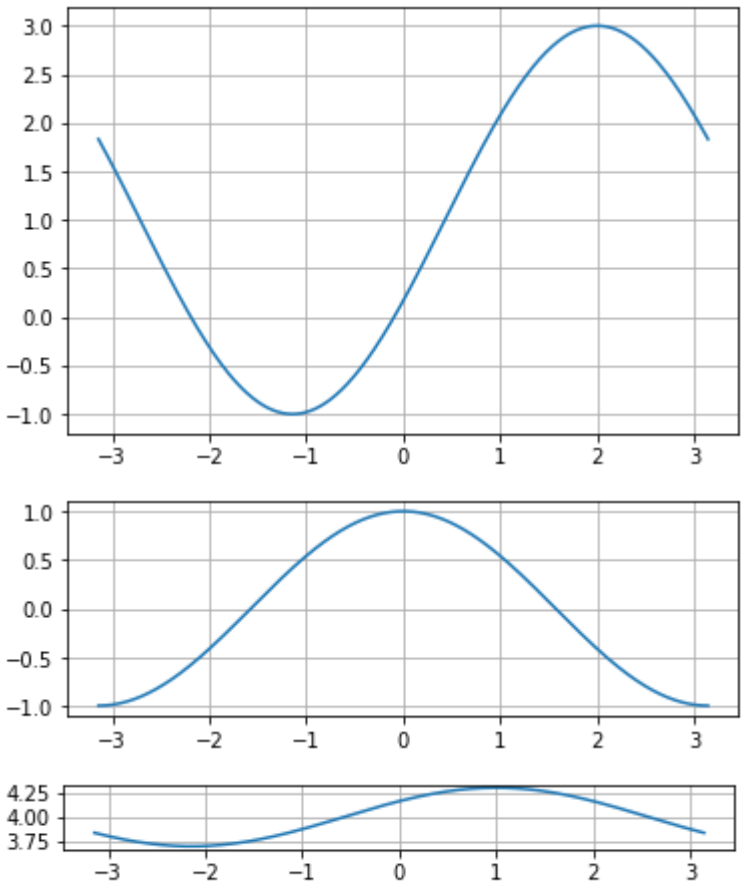
Задание 33

Задача 1.

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

In [36]:
```
def graf_f(k,a,b):
    plt.plot(x,k*np.cos(x-a)+b)
    plt.grid(True)
    plt.gca().set_aspect('equal')
    plt.show()

x = np.linspace(-np.pi,np.pi,512)
graf_f(2,2,1)
graf_f(1,0,0)
graf_f(0.3,1,4)
```



Задача 2.

Докажите, что при ортогональном преобразовании сохраняется расстояние между точками.

Для точек (x_1, y_1) и (x_2, y_2) растояние равно

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ортогональное преобразование:

$$\begin{cases} X = a_{11}x + a_{12}y + a_{13} \\ Y = a_{21}x + a_{22}y + a_{23} \end{cases}$$

При этом:

$$\begin{cases} a_{11}^2 + a_{21}^2 = 1 \\ a_{21}^2 + a_{22}^2 = 1 \\ a_{11}a_{12} + a_{21}a_{22} = 0 \end{cases}$$

Тогда:

$$L = \sqrt{a_{11}^2(x_2 - x_1)^2 + a_{12}^2(y_2 - y_1)^2 - 2a_{11}a_{12}(x_2 - x_1)(y_2 - y_1) + a_{21}^2(x_2 - x_1) + a_{22}^2(y$$

Задача 3.

1) Напишите код, который будет переводить полярные координаты в декартовы. 2) Напишите код, который будет рисовать график окружности в полярных координатах. 3) Напишите код, который будет рисовать график отрезка прямой линии в полярных координатах.

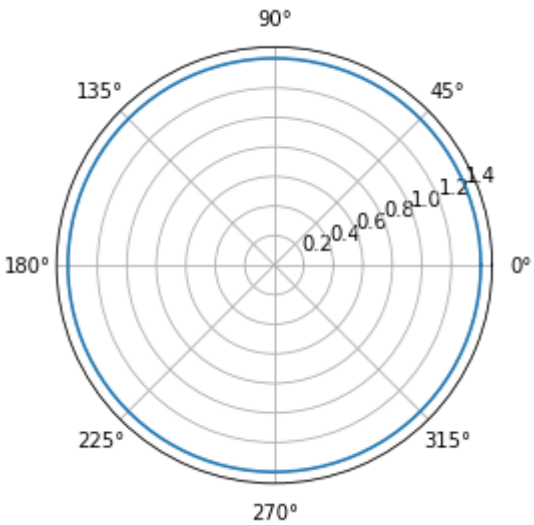
In [38]:
```
import math
def pol_to_dec(r:float, f:float)->(float, float):
    return (r*math.cos(f), r*math.sin(f))

pol_to_dec(1, math.pi/4)
```

Out[38]: (0.7071067811865476, 0.7071067811865475)

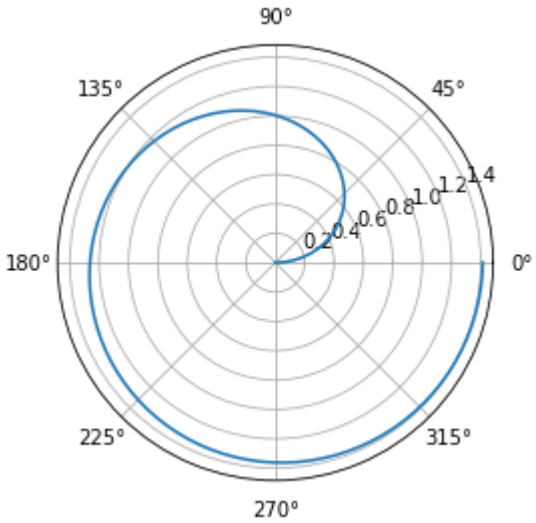
In [41]:
```
theta = np.linspace(0, 2*np.pi, 100)

plt.polar(theta, [1.4]*len(theta))  # окружность
plt.gca().set_aspect('equal')
plt.show()
```



In [42]:
```
theta = np.linspace(0, 2*np.pi, 100)

plt.polar(theta, np.arctan(theta))
plt.gca().set_aspect('equal')
plt.show()
```

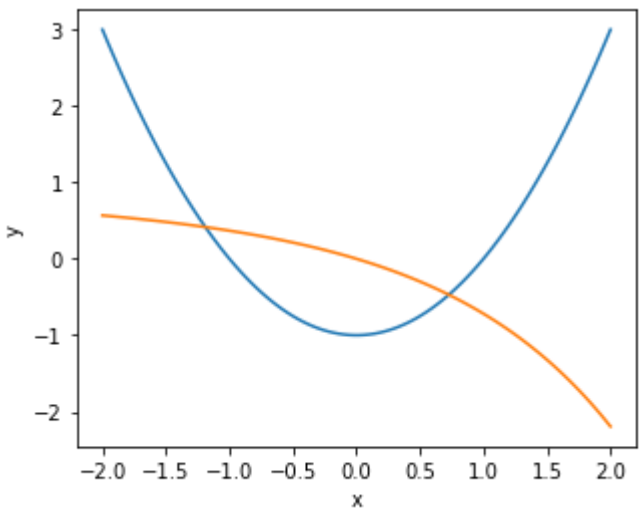


Задача 4.

Решите систему уравнений

$$\exp(x) + x \cdot (1-y) = 1$$
$$y = x^2 - 1$$

In [51]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-2, 2, 100)
y1 = x**2-1
y2 = (1-np.exp(x))/x +1
plt.figure(figsize=(5,4))
plt.plot(x, y1, x, y2)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



In [48]:
```
from scipy.optimize import fsolve
def equations(p):
    x, y = p
    return (y-x**2+1, np.exp(x)+x*(1-y)-1)
fsolve(equations, (1, 0))
```

Out[48]:
```
array([ 6.51825754e-15, -1.00000000e+00])
```

Решите систему уравнений и неравенств:

$$y = x^2 - 1$$
$$\exp(x) + x \cdot (1 - y) - 1 > 0$$

In [47]:
```
from scipy.optimize import fsolve
def equations(p):
    x, y = p
    return (y-x**2+1, np.exp(x)+x*(1-y)>1)
fsolve(equations, (0, 0))
```

Out[47]:
```
array([-7.21680063e-36, -1.00000000e+00])
```

In [ ]:
```

```