Atividade - Neurônio Artificial

Instituto Federal de Minas Gerais

Engenharia de Computação

Aluno: Gabriel Henrique Silva Duque

In [1]:
```
#bibliotecas
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

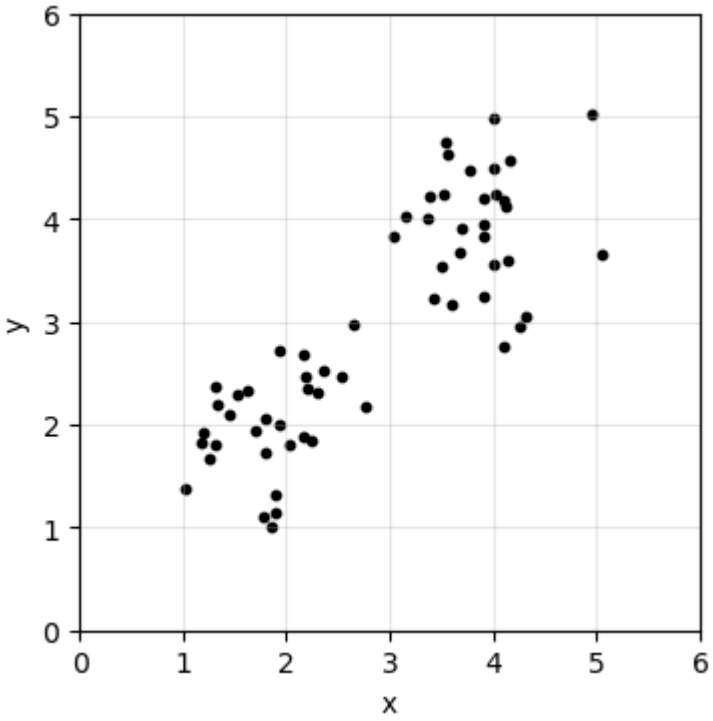
In [2]:
```
#função do neurônio
def meuNeuronio(x1, x2,w0, w1, w2, bias):
    soma = (x1 * w1) + (x2 * w2) - (bias * w0)
    return 1 if soma > 0 else 0
```

In [3]:
```
#leitura de dados
dados = pd.read_csv('amostrabivariada.csv', sep=';',decimal=',')

#convertendo para float se necessário
dados['x'] = pd.to_numeric(dados['x'],errors='coerce')
dados['y'] = pd.to_numeric(dados['y'],errors='coerce')

dados.head()
```

Out[3]:

	x	y
0	1.183988	1.832880
1	1.523565	2.293337
2	2.199241	2.342880
3	2.768052	2.179136
4	2.165374	1.888445

In [4]:
```
#criando o gráfico de dispersão
plt.figure(figsize=(4,4))
plt.scatter(dados['x'], dados['y'],c='black',s=10)
plt.xlim(0, 6)
plt.ylim(0, 6)
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, alpha=0.3)
plt.show()
```



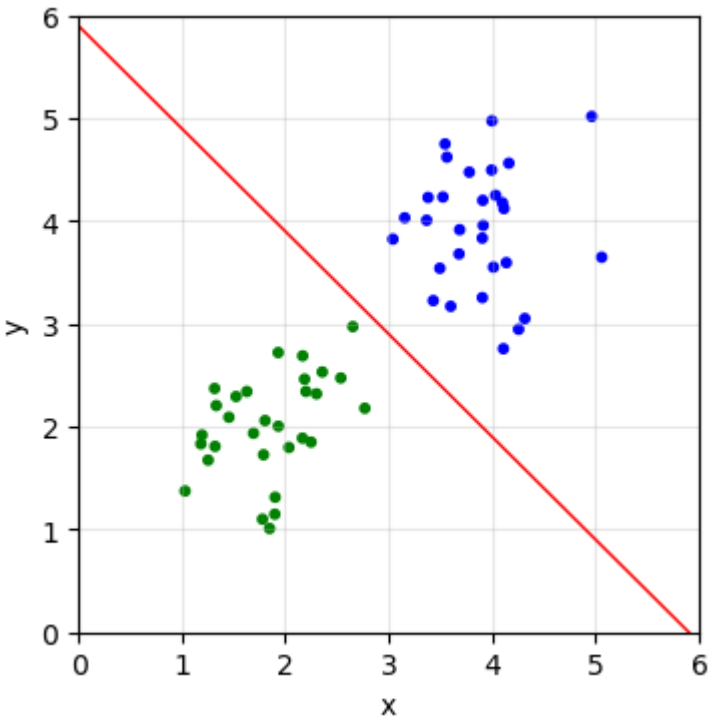
In [5]:
```
w0 = 5.9 #peso do bias
w1 = -1 #peso da entrada x1
w2 = -1 #peso da entrada x2
bias = -1  #valor do bias

#aplicando a função do neurônio aos dados
classificacoes = []
for i, linha in dados.iterrows():
    classe =meuNeuronio(linha['x'], linha['y'], w0, w1, w2, bias)
    classificacoes.append(classe)

#adicionando as classificações ao dataframe
dados['classe'] = classificacoes
dados.head()
```

Out[5]:

	x	y	classe
0	1.183988	1.832880	1
1	1.523565	2.293337	1
2	2.199241	2.342880	1
3	2.768052	2.179136	1
4	2.165374	1.888445	1

In [6]:
```
plt.figure(figsize=(4,4))
cores = ['blue' if c == 0 else 'green' for c in dados['classe']]
plt.scatter(dados['x'], dados['y'], c=cores, s=10)

#gerando a reta de separação
eixox = np.linspace(0, 6, 100)
#eixoy = (w0 * -1) + eixox
eixoy = w0 - eixox
plt.plot(eixox, eixoy, 'r-', linewidth=1)
```

```
plt.xlim(0, 6)
plt.ylim(0, 6)
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True, alpha=0.3)
plt.show()
```



Atividade - Funções de Ativação

Instituto Federal de Minas Gerais

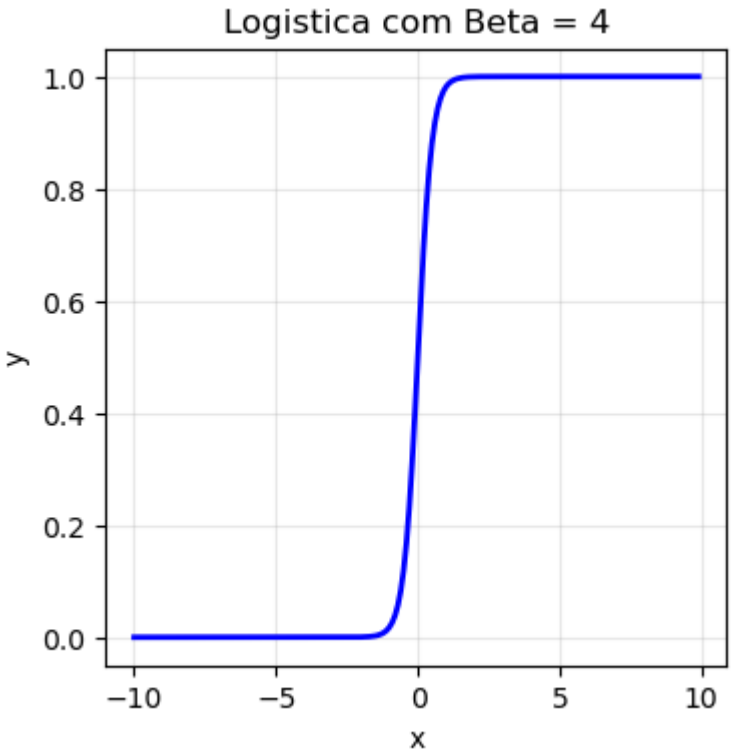
Engenharia de Computação

Aluno: Gabriel Henrique Silva Duque

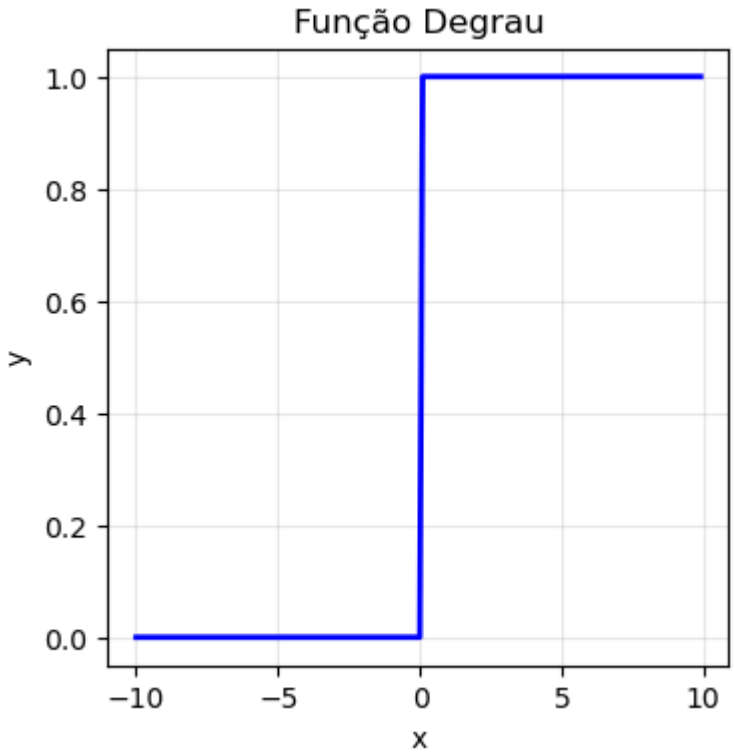
In [7]:
```
#função plota grafico
def plota_grafico(x,y,titulo):
    plt.figure(figsize=(4,4))
    plt.plot(x,y,'b-', linewidth=2)
    plt.title(titulo)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(True,alpha=0.3)
    plt.show()

#funcao logistica
def logistica(x,beta):
    return 1/ (1 + np.exp(-beta * x))
```

In [8]:
```
#gera os valores de x
x_val = np.arange(-10,10,0.1)

#calcula a funcao logistica
beta = 4
```

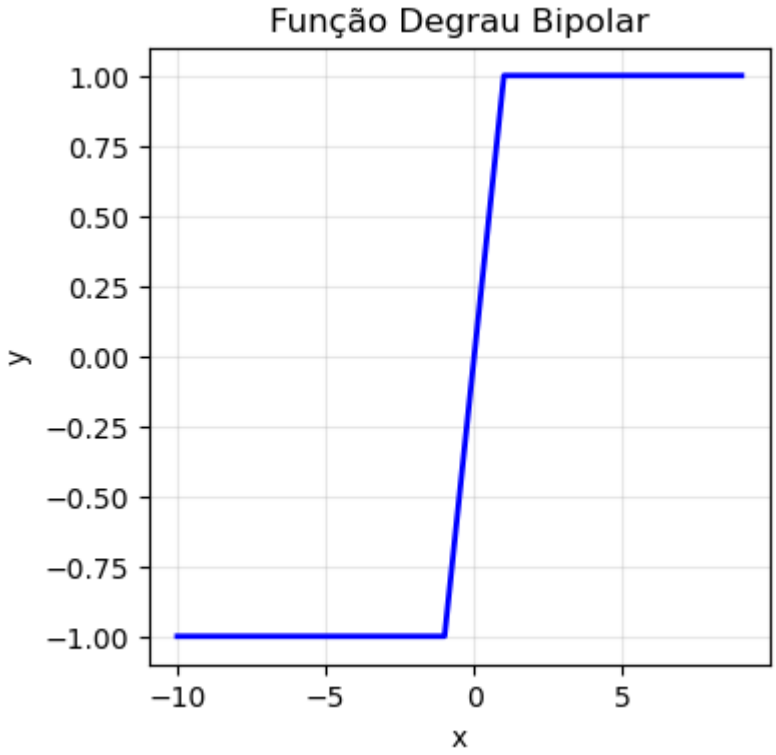
```
y_val = logistica(x_val,beta)
plota_grafico(x_val, y_val, 'Logistica com Beta = ' + str(beta))
```



In [9]:
```
#funcao degrau
def funcao_degrau(x):
    # Suporta arrays numpy
    return np.where(x >= 0, 1, 0)

#gera os valores de x
x_valor = np.arange(-10, 10, 0.1)
y_valor = funcao_degrau(x_valor)

plota_grafico(x_valor, y_valor, 'Função Degrau')
```

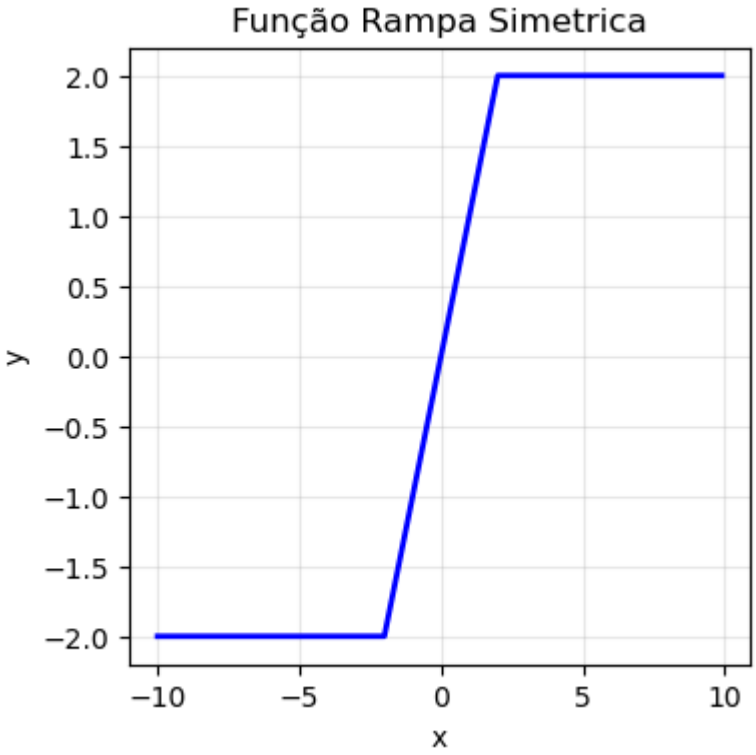


In [22]:
```
#degrau bipolar
def degrauBipolar(u):
 return np.where(u>0,1, np.where(u<0,-1,0))
#gera os valores de x
x_valor = np.arange(-10, 10, 1)
y_valor = degrauBipolar(x_valor)

plota_grafico(x_valor, y_valor, 'Função Degrau Bipolar')
```

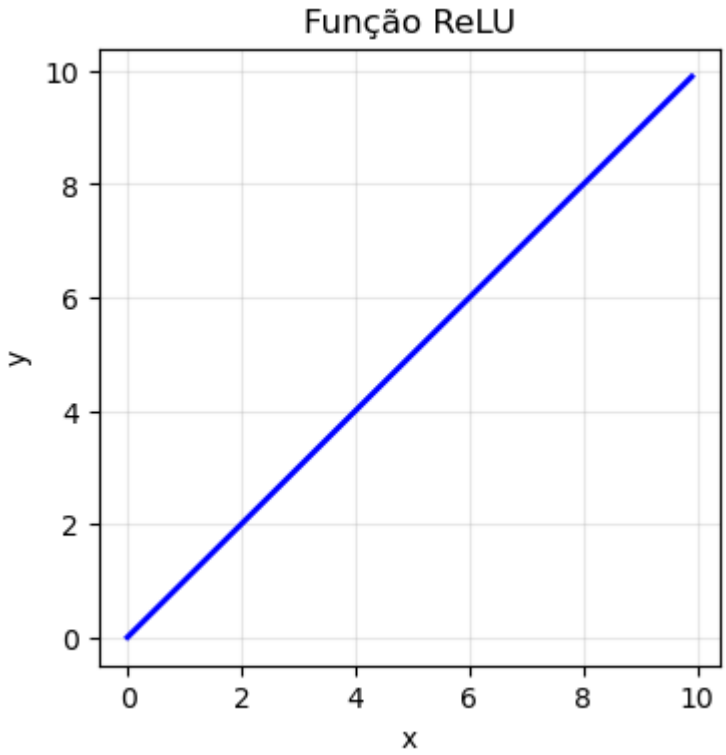


In [11]:
```
#funcao rampa simetrica
def rampa_simetrica(u,a):
    return np.maximum(-a, np.minimum(a, u))
```

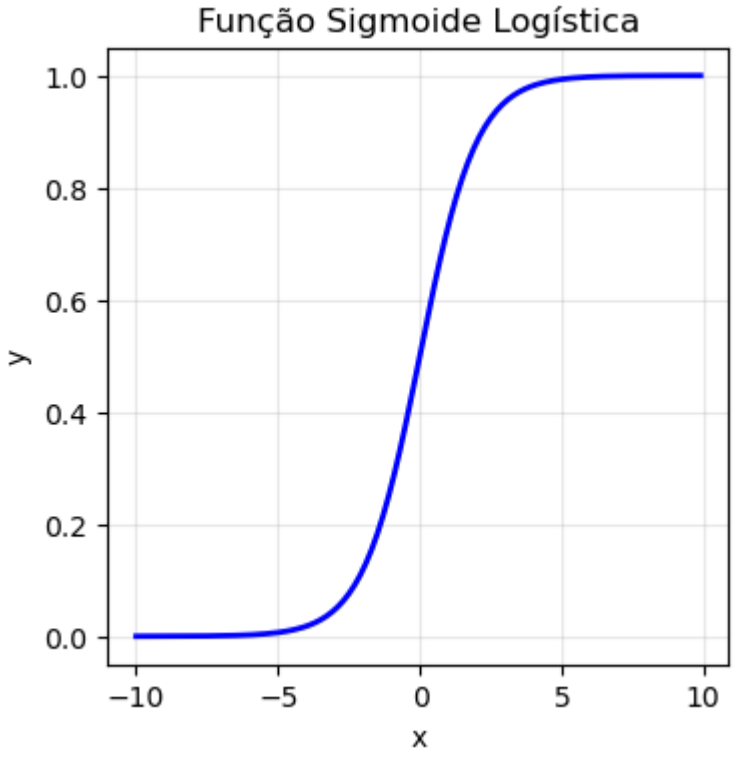
```
a = 2

#gera os valores de x
x_valor = np.arange(-10, 10, 0.1)
y_valor = rampa_simetrica(x_valor,a)

plota_grafico(x_valor, y_valor, 'Função Rampa Simetrica')
```



In [12]:
```
#funcao ReLU
def relu(u):
    return np.maximum(0, u)

#gera os valores de x
x_valor = np.arange(0, 10, 0.1)
y_valor = relu(x_valor)

plota_grafico(x_valor, y_valor, 'Função ReLU')
```

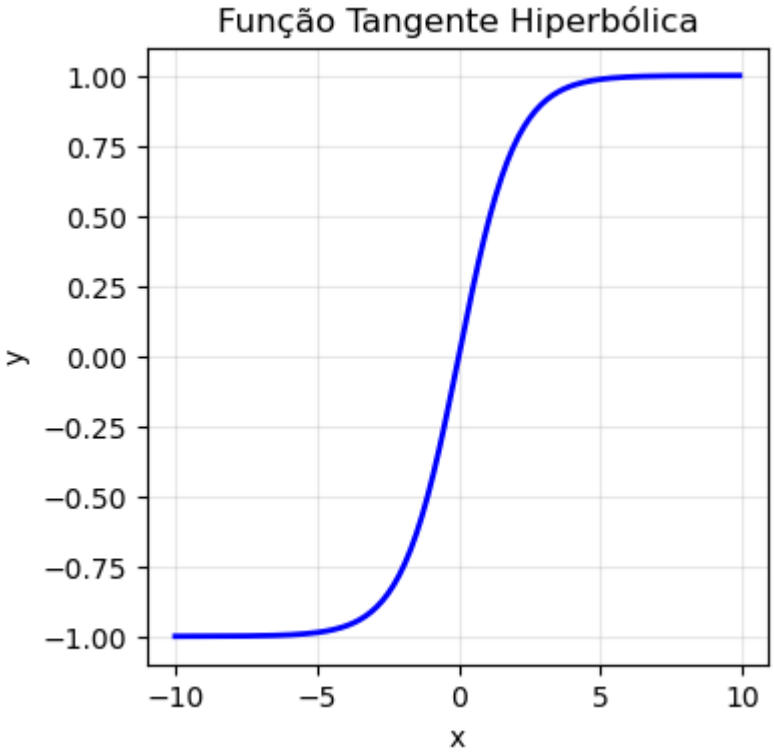


In [13]:
```python
#funcao sigmoide logistica
def sigmoide_logistica(u, beta=1):
    return 1/ (1 + np.exp(-beta * u))

#gera os valores de x
x_valor = np.arange(-10, 10, 0.1)
y_valor = sigmoide_logistica(x_valor)

plota_grafico(x_valor, y_valor, 'Função Sigmoide Logística')
```

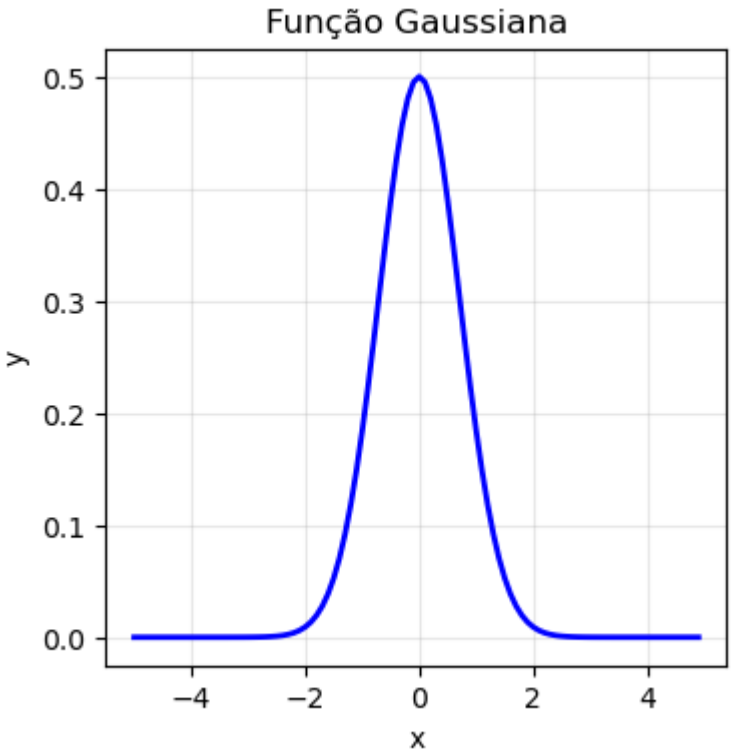


In [14]:
```python
#funcao tangente hiperbolica
def tan_hiperbolica(u, beta=1):
```

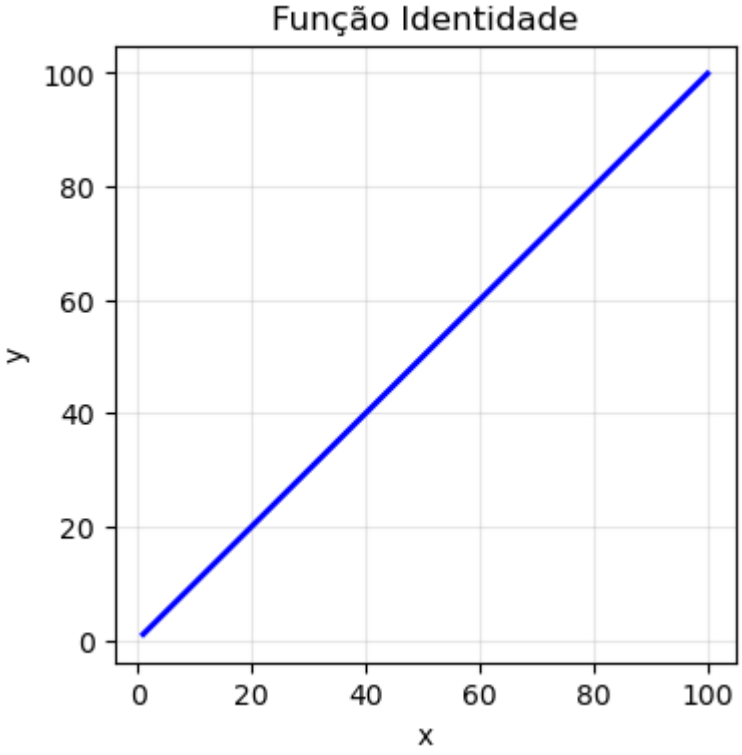
```
    return (1 - np.exp(-beta * u))/(1 + np.exp(-beta * u))

#gera os valores de x
x_valor = np.arange(-10, 10, 0.1)
y_valor = tan_hiperbolica(x_valor)

plota_grafico(x_valor, y_valor, 'Função Tangente Hiperbólica')
```



In [25]:
```
#funcao gaussiana
def gaussiana(u,c = 0, sigma = 1):
    return np.exp(-((u - c))**2)/ (2 * sigma**2)

#gera os valores de x
x_valor = np.arange(-5, 5, 0.1)
y_valor = gaussiana(x_valor)

plota_grafico(x_valor, y_valor, 'Função Gaussiana')
```



In [23]:
```
#funcao identidade
def funcao_identidade(u):
    return u

#gera os valores de x
x_valor = np.arange(1, 100, 0.1)
y_valor = funcao_identidade(x_valor)

plota_grafico(x_valor, y_valor, 'Função Identidade')
```



In [ ]: