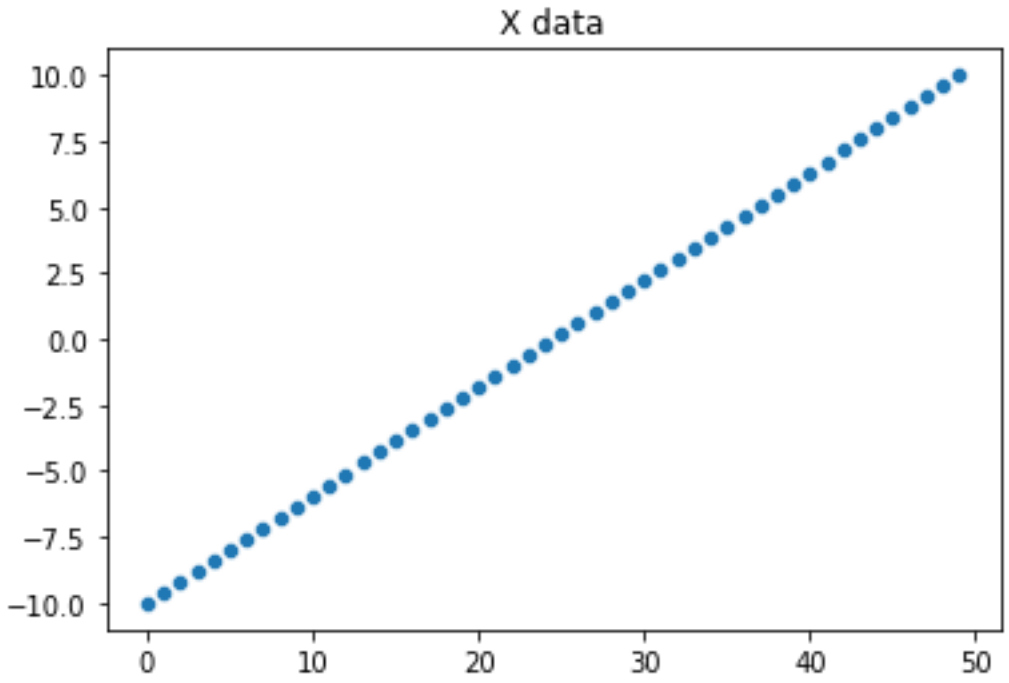
Lab Assignment No. 1

#Sigmoid Function

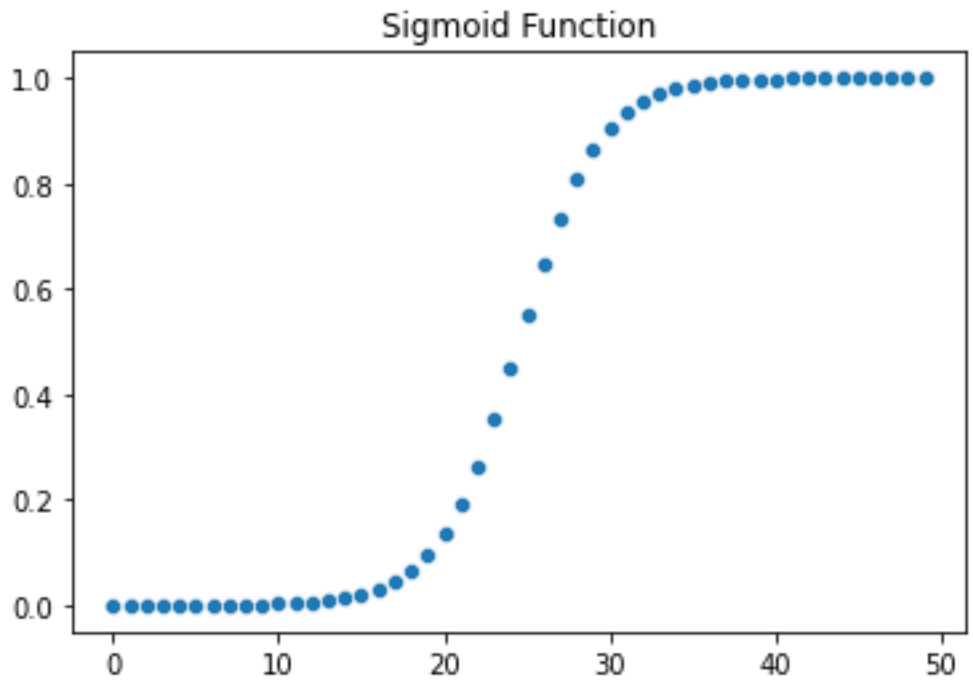
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

def sigmoid_function(x):
  return 1/(1+np.exp(-x))

x = np.linspace(-10, 10)

sns.scatterplot(x)
plt.title("X data")
```

```
Text(0.5, 1.0, 'X data')
```



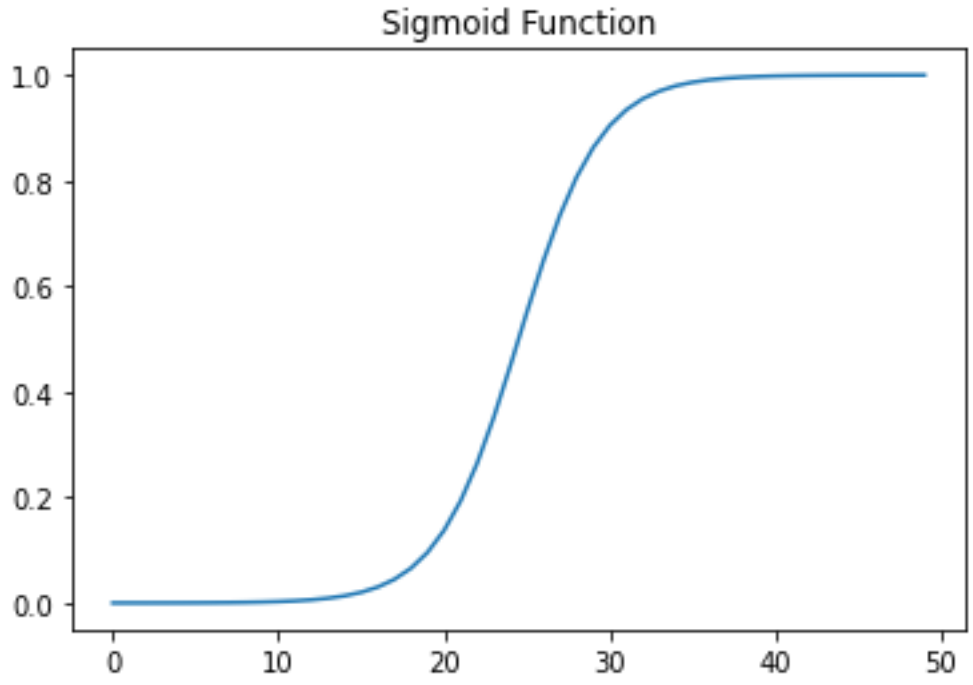

```
y = sigmoid_function(x)

sns.scatterplot(y)
plt.title("Sigmoid Function")
```

```
Text(0.5, 1.0, 'Sigmoid Function')
```




```
plt.plot(y)
plt.title("Sigmoid Function")
```

```
Text(0.5, 1.0, 'Sigmoid Function')
```



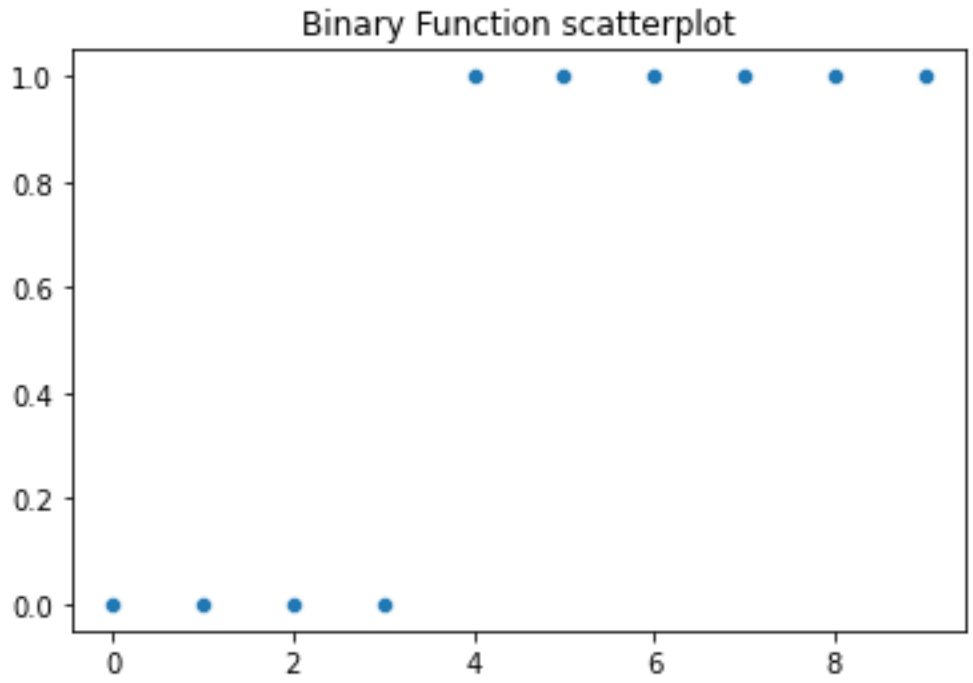
#Binary function

```
def binary_function(x):
  return 0 if x<0 else 1

binary_output = []
for i in [-10, -8, -5, -2, 0, 1, 2, 3, 8, 10]:
  binary_output.append(binary_function(i))

binary_output
```

```
[0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
```

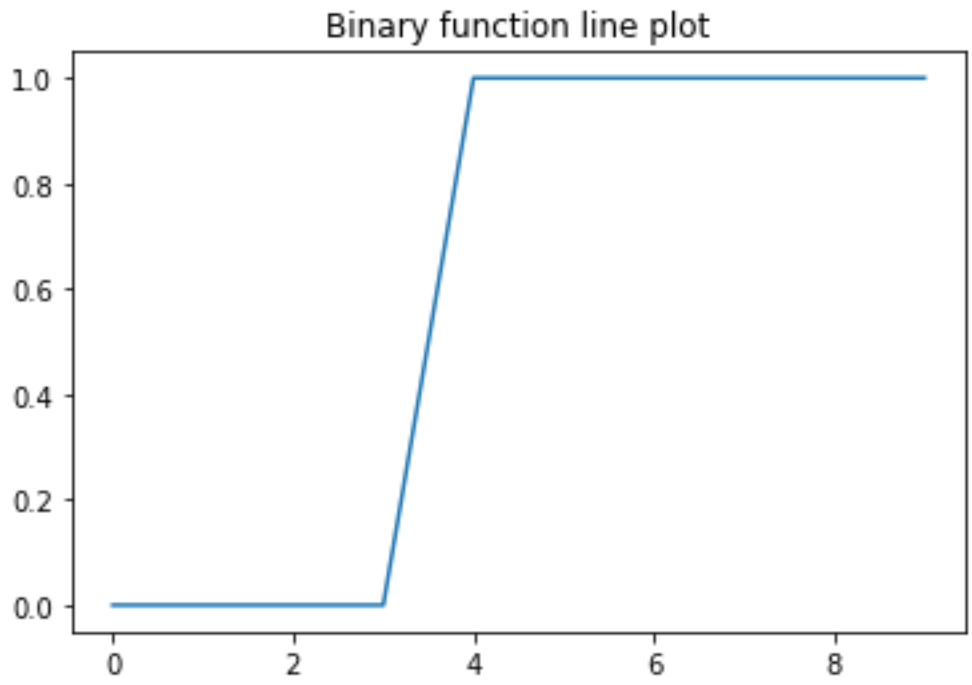
```
sns.scatterplot(binary_output)
plt.title("Binary Function scatterplot")
```

```
Text(0.5, 1.0, 'Binary Function scatterplot')
```



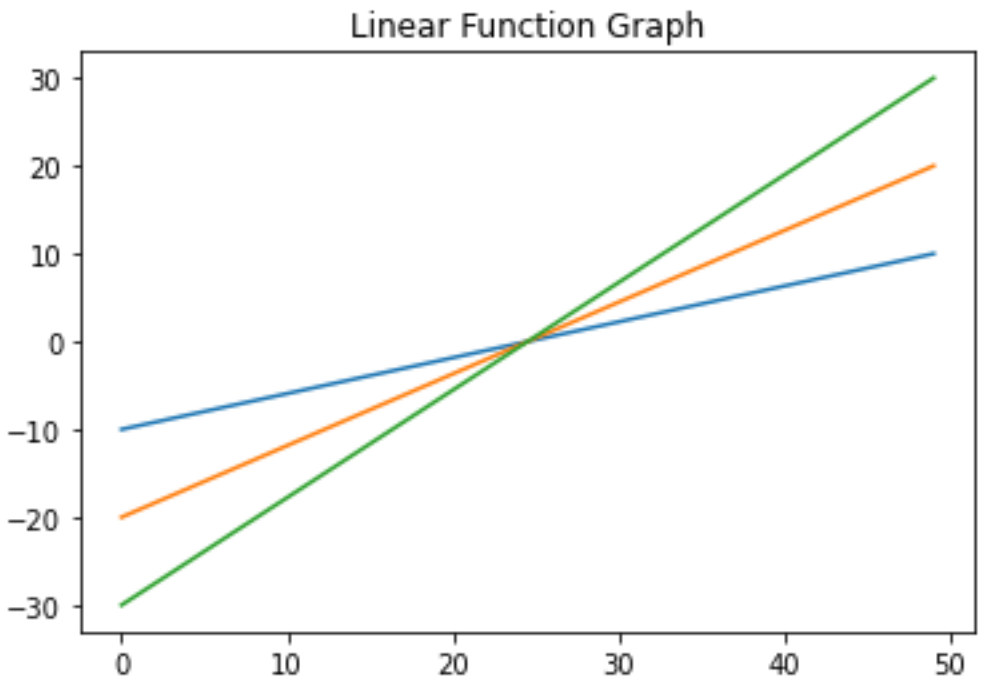

```
plt.plot(binary_output)
plt.title("Binary function line plot")
```

```
Text(0.5, 1.0, 'Binary function line plot')
```



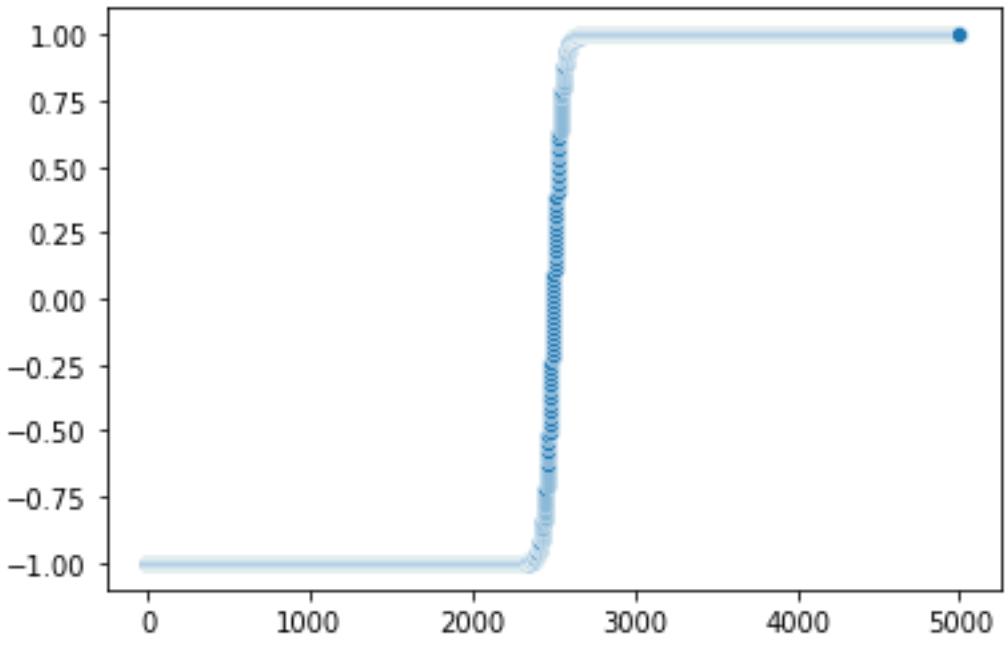
#Linear Function

```
def linear_function(a, x):
  return a*x;

x = np.linspace(-10, 10)

y1 = [linear_function(1, i) for i in x]

y2 = [linear_function(2, i) for i in x]

y3 = [linear_function(3, i) for i in x]

plt.plot(y1)
plt.plot(y2)
plt.plot(y3)
plt.title("Linear Function Graph")

Text(0.5, 1.0, 'Linear Function Graph')
```



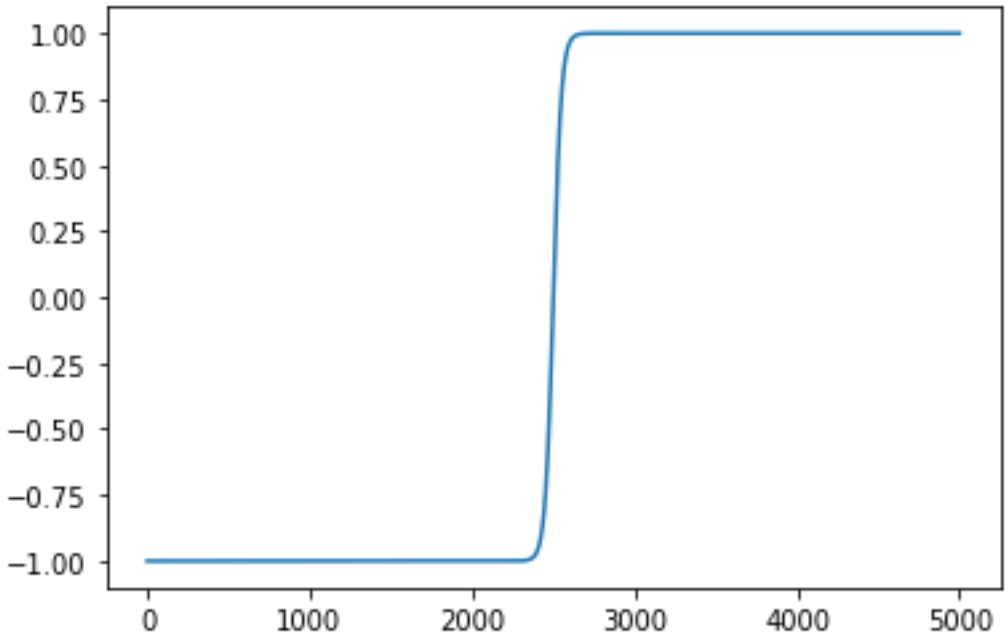
#Tanh

```
def tanh_function(x):
  return (np.exp(x) - np.exp(-x) / np.exp(x) + np.exp(-x))

def tanh_function2(x):
  return 2*sigmoid_function(2*x)-1

x = np.linspace(-50, 50, 5000)

y = [tanh_function(i) for i in x]

y_tanh = [tanh_function2(i) for i in x]

sns.scatterplot(y_tanh)

<Axes: >
```



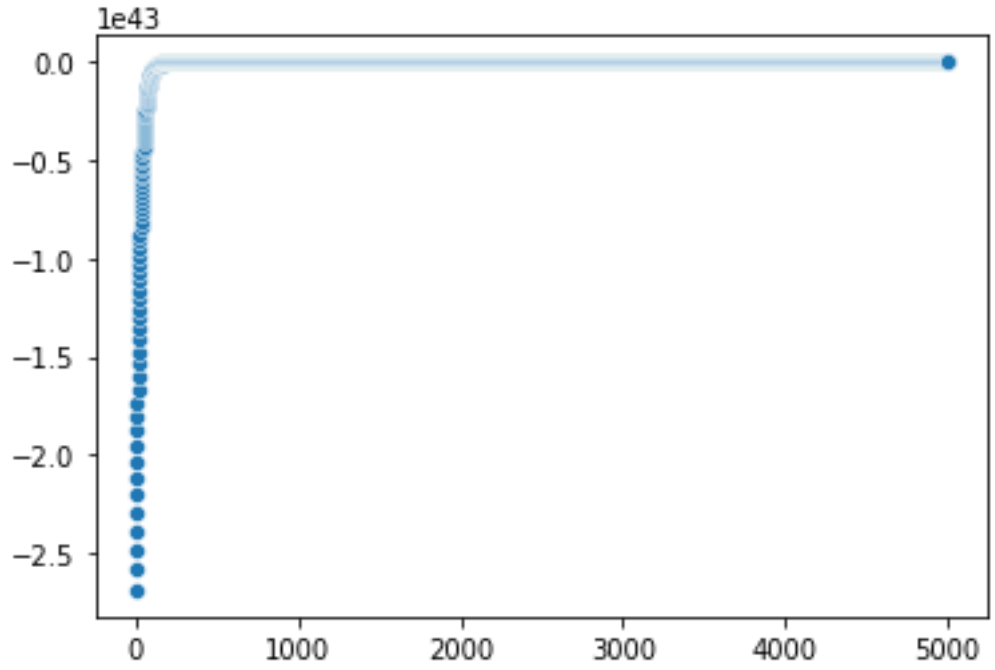

```
plt.plot(y_tanh)
```

```
[<matplotlib.lines.Line2D at 0x7f9a52d413a0>]
```



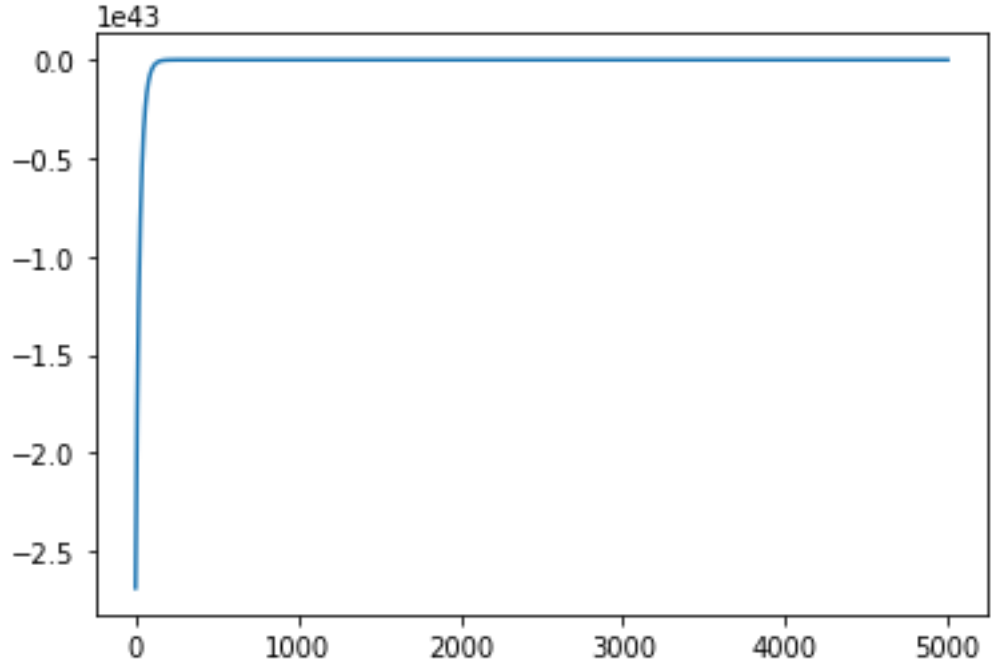

```
sns.scatterplot(y)
```

```
<Axes: >
```




```
plt.plot(y)
```

```
[<matplotlib.lines.Line2D at 0x7f9a52c26790>]
```

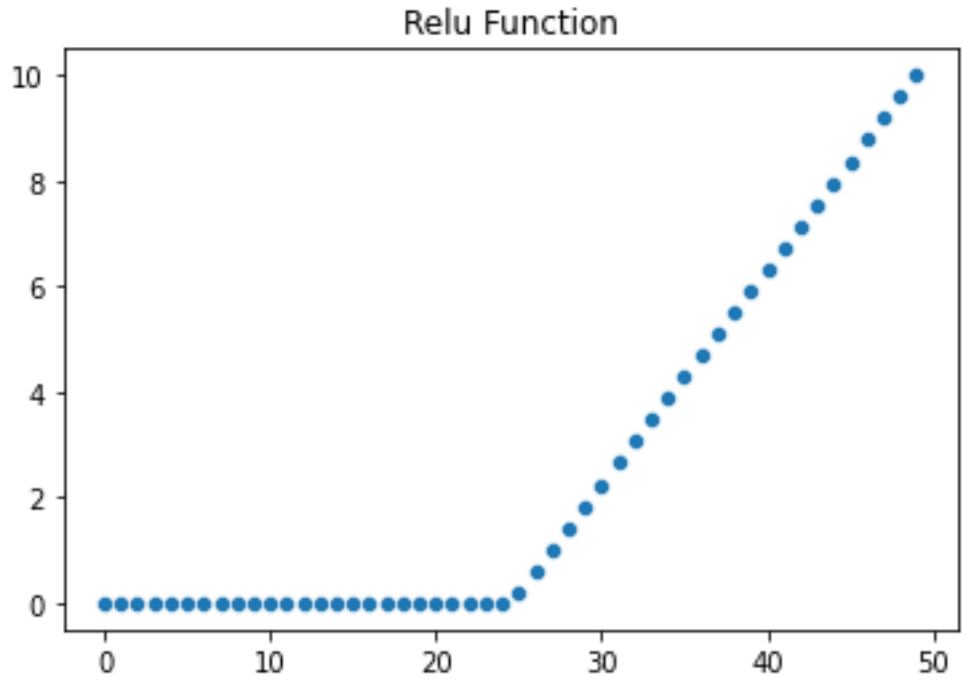


#Relu

```
def relu_function(x):
  return np.array([0, x]).max()
```

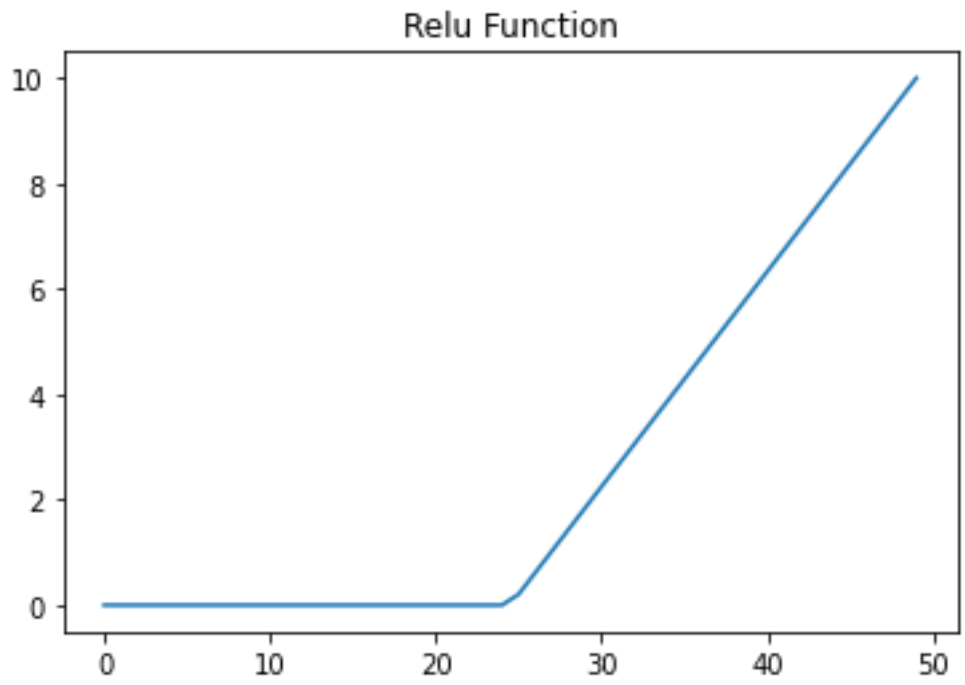
```
x = np.linspace(-10, 10)

y = [relu_function(i) for i in x]

sns.scatterplot(y)
plt.title("Relu Function")
```

```
Text(0.5, 1.0, 'Relu Function')
```



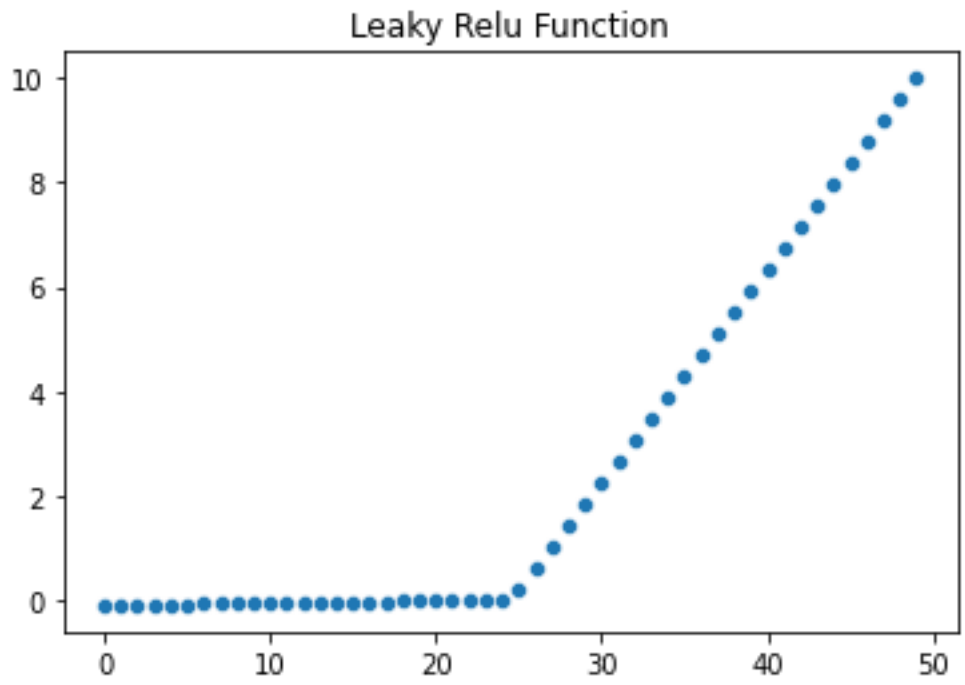

```
plt.plot(y)
plt.title("Relu Function")
```

```
Text(0.5, 1.0, 'Relu Function')
```



#leaky Relu

```
def leaky_relu_function(x):
  return  0.01*x if x < 0 else x

x = np.linspace(-10, 10)

y = [leaky_relu_function(i) for i in x]

sns.scatterplot(y)
plt.title('Leaky Relu Function')

Text(0.5, 1.0, 'Leaky Relu Function')
```




```
plt.plot(y)
plt.title("Leaky Relu Function")
```

```
Text(0.5, 1.0, 'Leaky Relu Function')
```