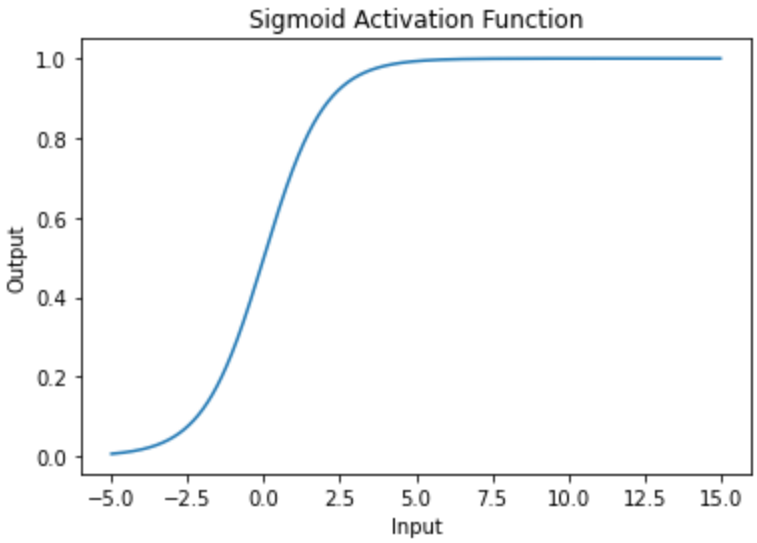
Activation Functions in Deep neural Networks.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-5,15,0.01)
y = 1/(1 + np.exp(-x))
plt.plot(x,y)
plt.title("Sigmoid Activation Function")
plt.xlabel('Input')
plt.ylabel('Output')
```

Out[ ]:
```
Text(0, 0.5, 'Output')
```

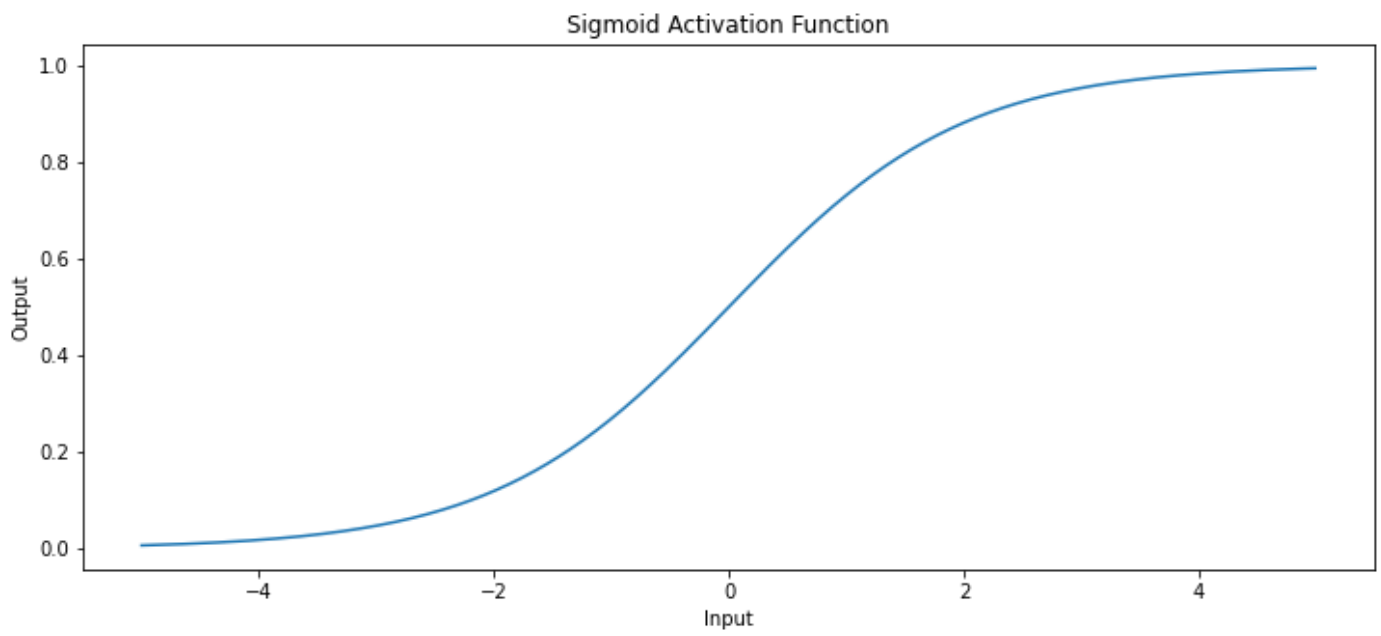


sigmoid Activation Function

In [ ]:
```
x = np.arange(-5,5,0.01)
y = 1/(1 + np.exp(-x))
```

In [ ]:
```
plt.figure(figsize=(12,5))
plt.plot(x,y)
plt.title("Sigmoid Activation Function")
plt.xlabel('Input')
plt.ylabel('Output')
```

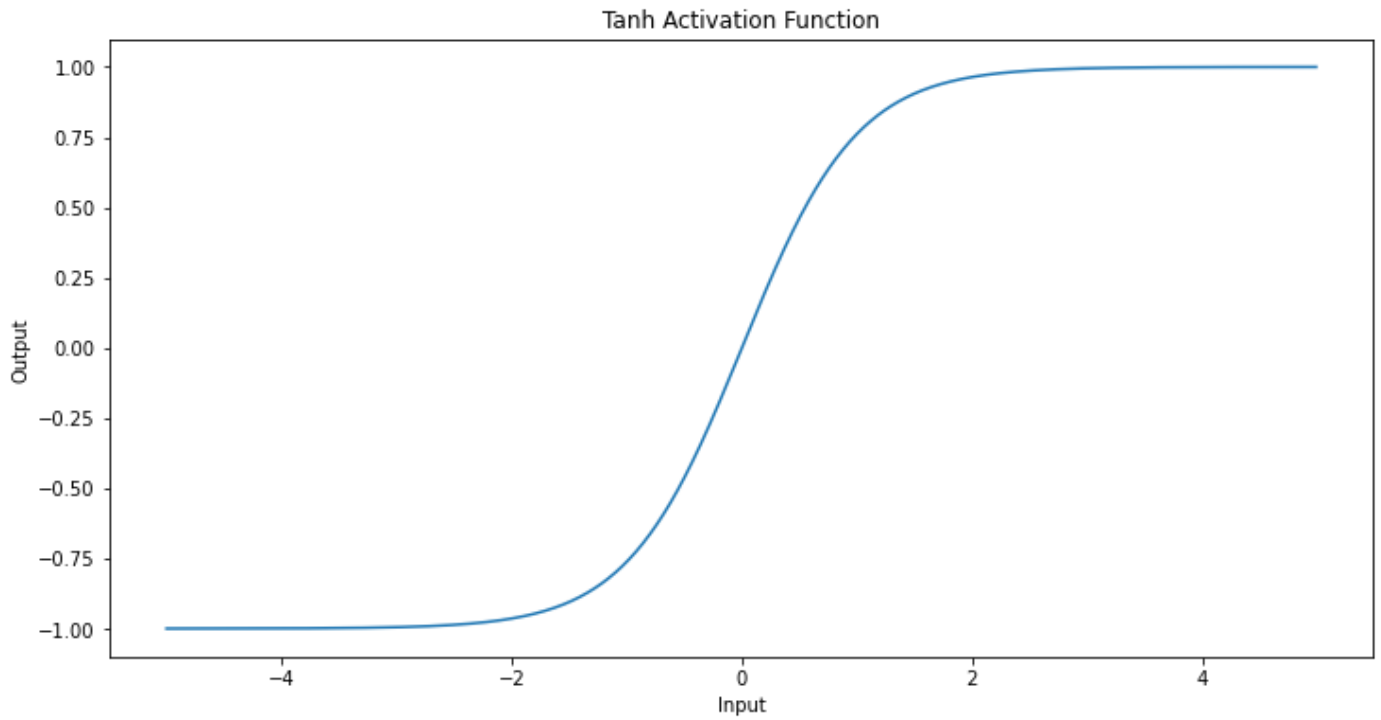
Out[ ]:
```
Text(0, 0.5, 'Output')
```



Tanh Activation Function

In [ ]:
```
x = np.arange(-5, 5, 0.01)
y = (2 / (1 + np.exp(-2*x)))-1
plt.figure(figsize=(12,6))
plt.plot(x,y)
plt.title('Tanh Activation Function')
plt.xlabel('Input')
plt.ylabel('Output')
```

Out[ ]:
```
Text(0, 0.5, 'Output')
```

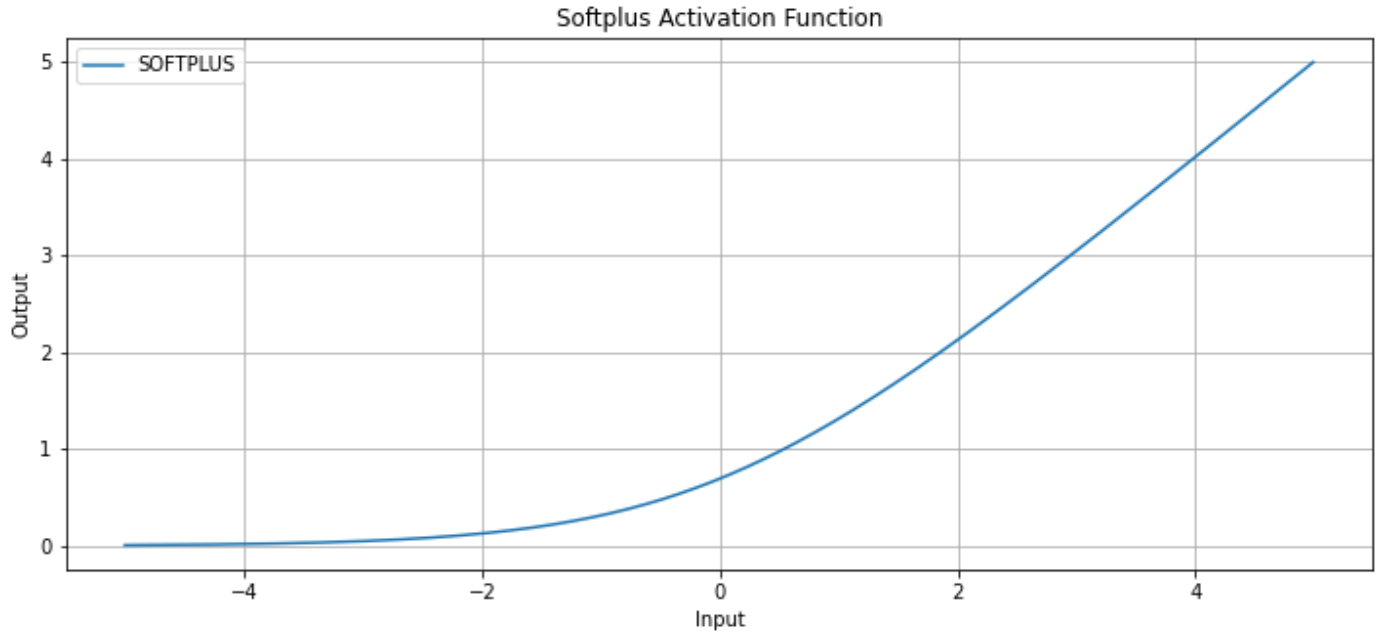


Softplus Activation Function

In [ ]:
```
x = np.arange(-5, 5, 0.01)
y = np.log(1+np.exp(x))
plt.figure(figsize=(12,5))
```

```
plt.plot(x,y)
plt.title('Softplus Activation Function')
plt.legend(['SOFTPLUS'])
plt.grid()
plt.xlabel('Input')
plt.ylabel('Output')
```

Out[ ]:
```
Text(0, 0.5, 'Output')
```



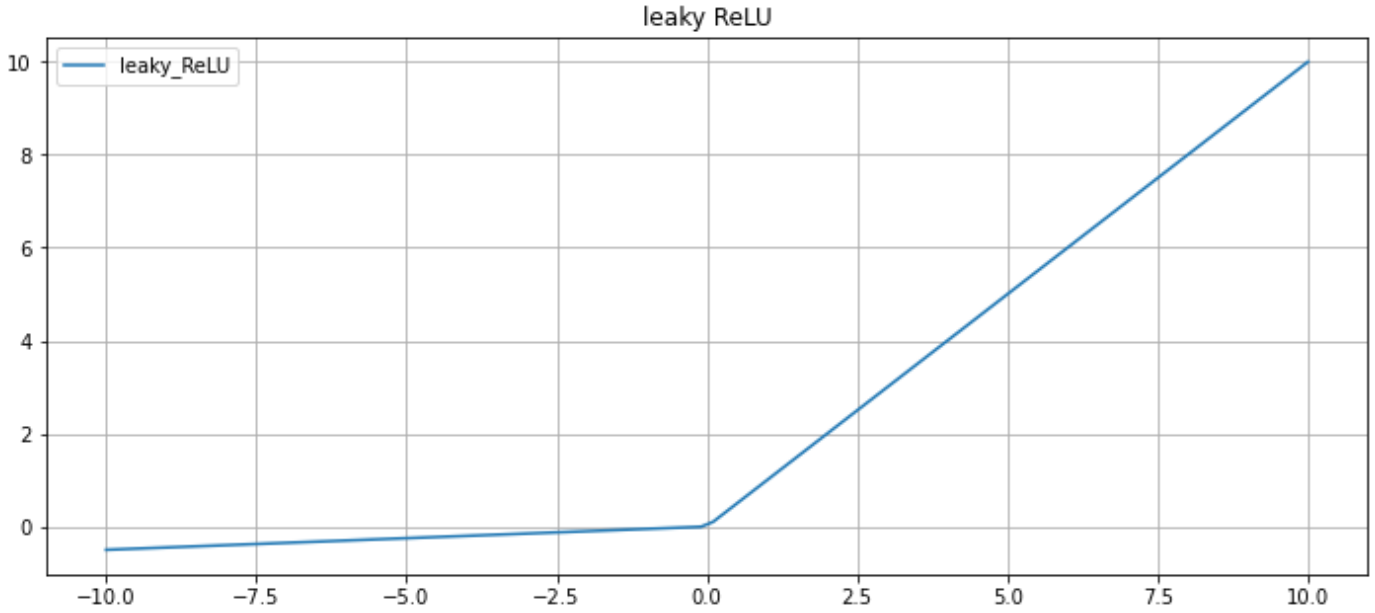
Leaky_ReLU activation Function

In [ ]:
```
# Leaky Rectified Linear Unit (leaky ReLU) Activation Function
def leaky_ReLU(x):
  data = [max(0.05*value,value) for value in x]
  return np.array(data, dtype=float)

# Derivative for leaky ReLU
def der_leaky_ReLU(x):
  data = [1 if value>0 else 0.05 for value in x]
  return np.array(data, dtype=float)

# Generating data For Graph
x_data = np.linspace(-10,10,100)
y_data = leaky_ReLU(x_data)
dy_data = der_leaky_ReLU(x_data)

# Graph
plt.figure(figsize=(12,5))
plt.plot(x_data, y_data)
plt.title('leaky ReLU')
plt.legend(['leaky_ReLU'])
plt.grid()
plt.show()
```

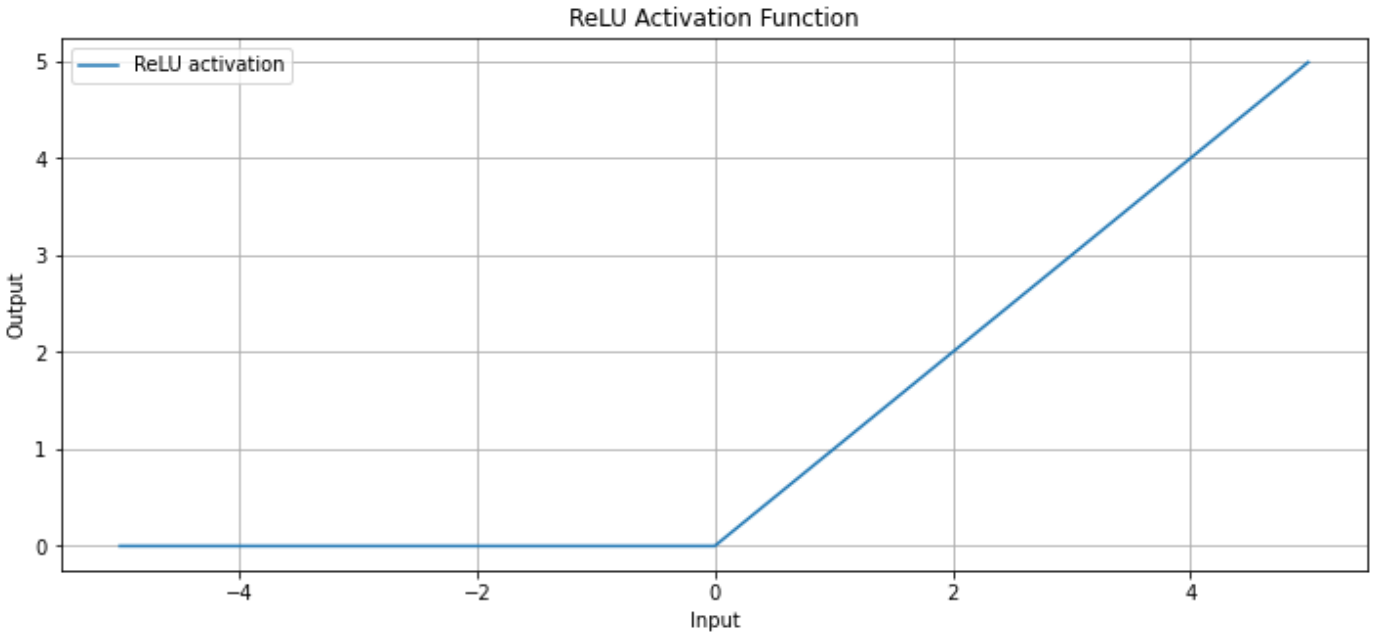


ReLU activation Function

In [ ]:
```
x = np.arange(-5, 5, 0.01)
z = np.zeros(len(x))
y = np.maximum(z,x)
plt.figure(figsize=(12,5))
plt.plot(x,y)

plt.title('ReLU Activation Function')
plt.legend(["ReLU activation"])
plt.grid()
plt.xlabel('Input')
plt.ylabel('Output')
```

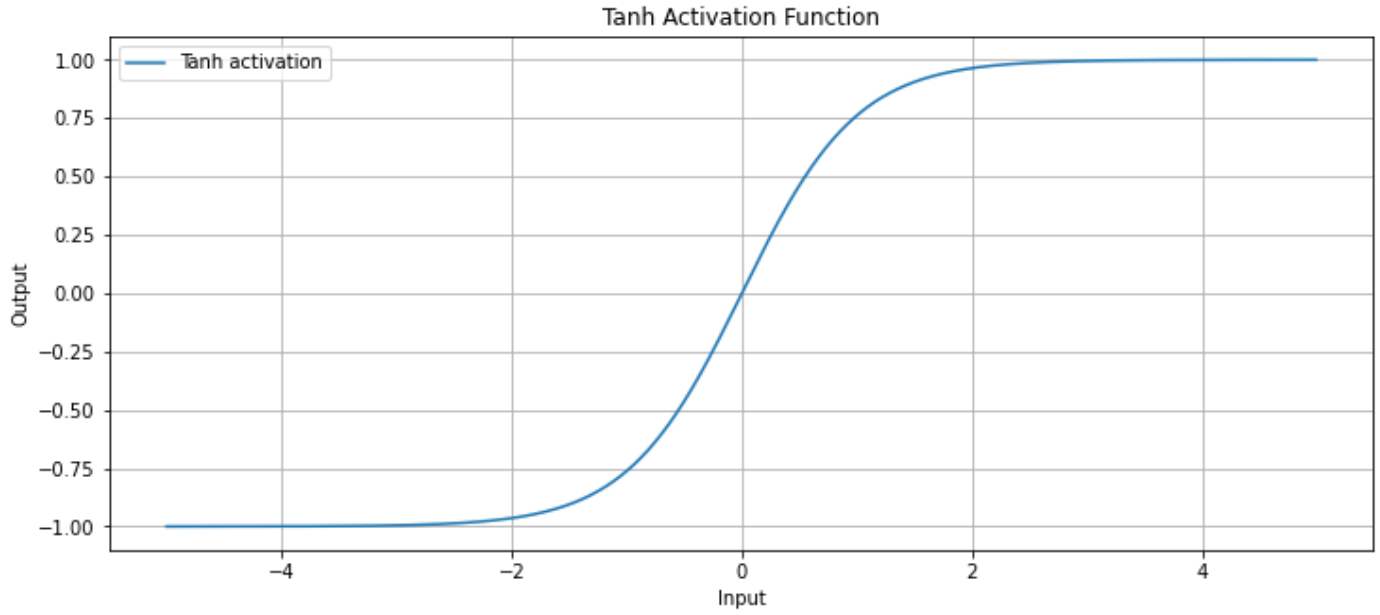
Out[ ]:
```
Text(0, 0.5, 'Output')
```



In [ ]:
```
x = np.arange(-5, 5, 0.01)
y = (2 / (1 + np.exp(-2*x)))-1
plt.figure(figsize=(12,5))
plt.plot(x,y)
plt.title('Tanh Activation Function')
```

```
plt.legend(['Tanh activation'])
plt.xlabel('Input')
plt.ylabel('Output')
```

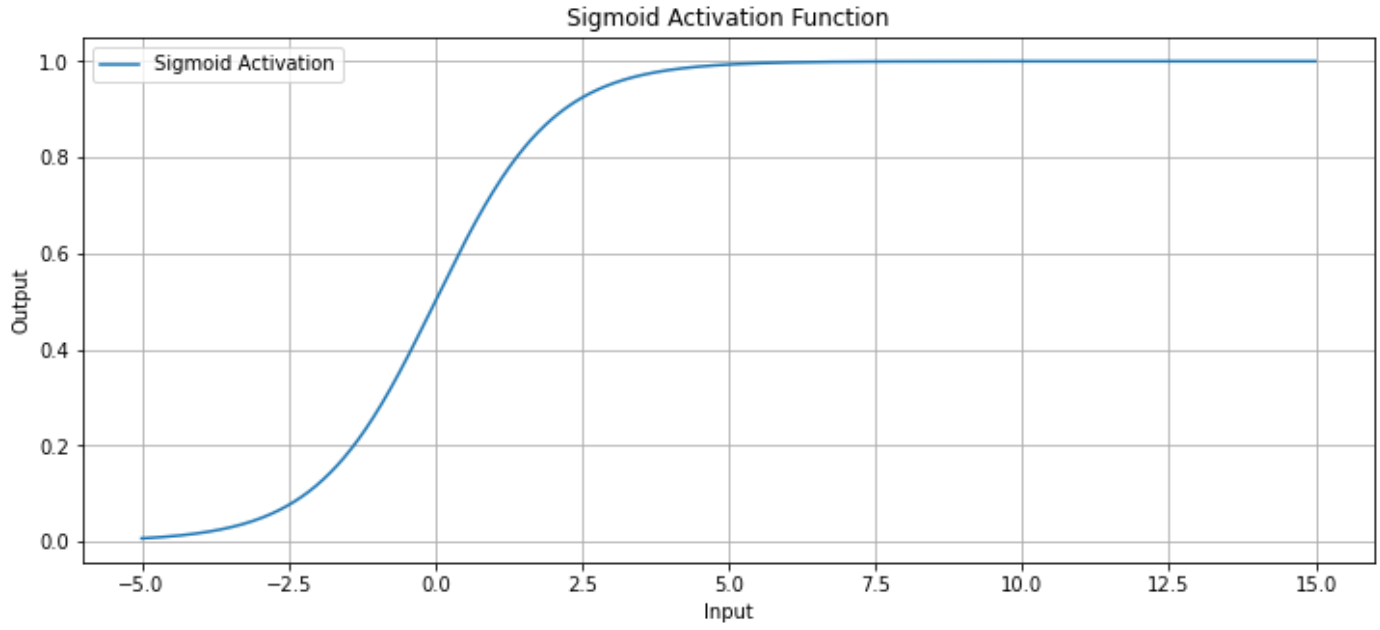
Out[ ]:
```
Text(0, 0.5, 'Output')
```



Sigmoid Activation Fucntions

In [ ]:
```
x = np.arange(-5,15,0.01)
y = 1/(1 + np.exp(-x))
plt.figure(figsize=(12,5))
plt.plot(x,y)
plt.title("Sigmoid Activation Function")
plt.grid()
plt.legend(['Sigmoid Activation'])
plt.xlabel('Input')
plt.ylabel('Output')
```

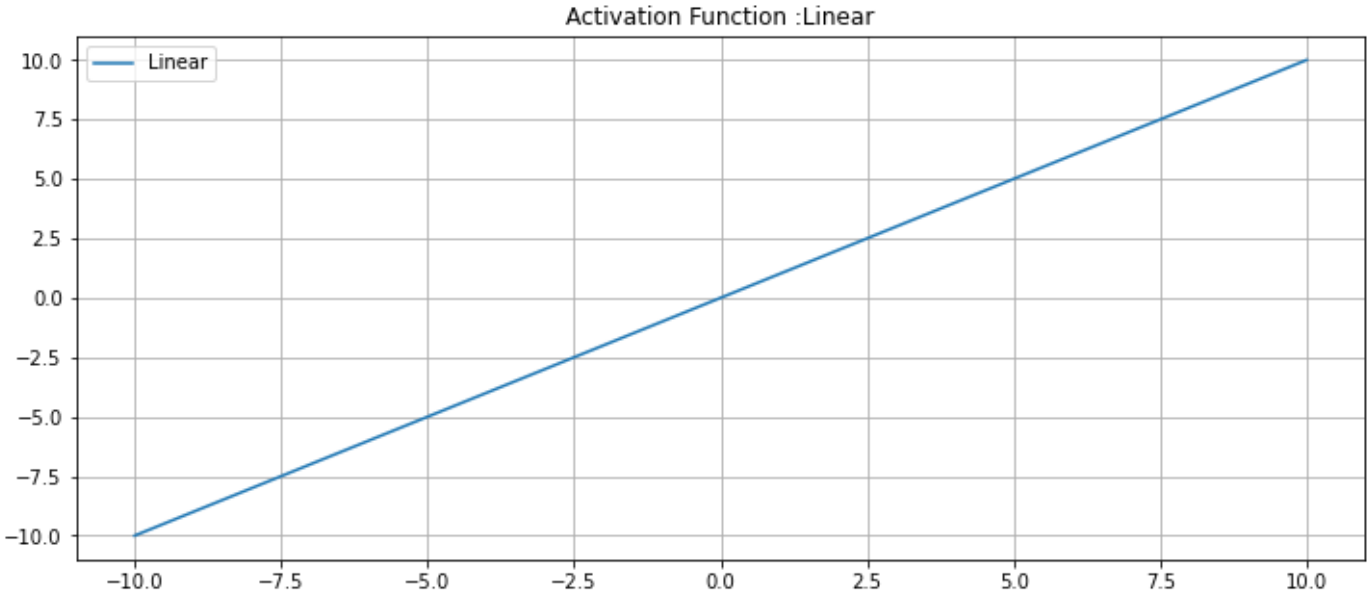
Out[ ]:
```
Text(0, 0.5, 'Output')
```



Activation Fucntion : Linear

In [ ]:
```
def linear(x):
    return x
```

In [ ]:
```
x = np.linspace(-10, 10)
plt.figure(figsize=(12,5))
plt.plot(x, linear(x))
plt.axis('tight')
plt.title('Activation Function :Linear')
plt.grid()
plt.legend(['Linear'])
plt.show()
```



Activation Function : Step Fucntion

In [ ]:
```
def step_function(x):
    return np.heaviside(x,1)
x = np.linspace(-10, 10)
plt.figure(figsize=(12,5))
plt.plot(x, step_function(x))
plt.axis('tight')
plt.title('Activation Function :Step function')
plt.grid()
plt.legend(['Step function'])
plt.show()
```