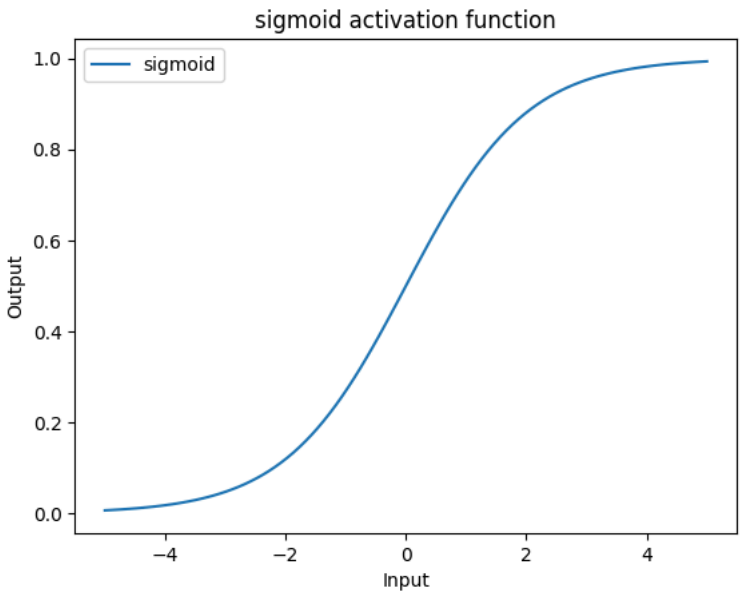
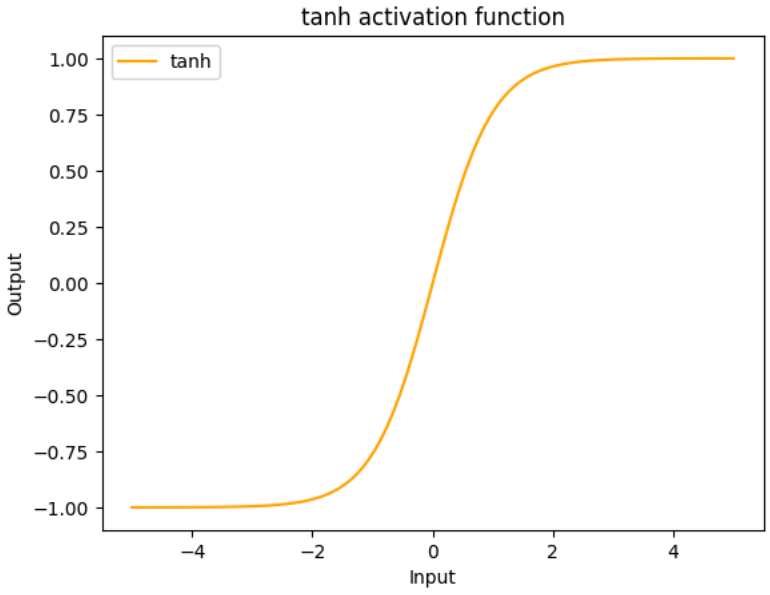
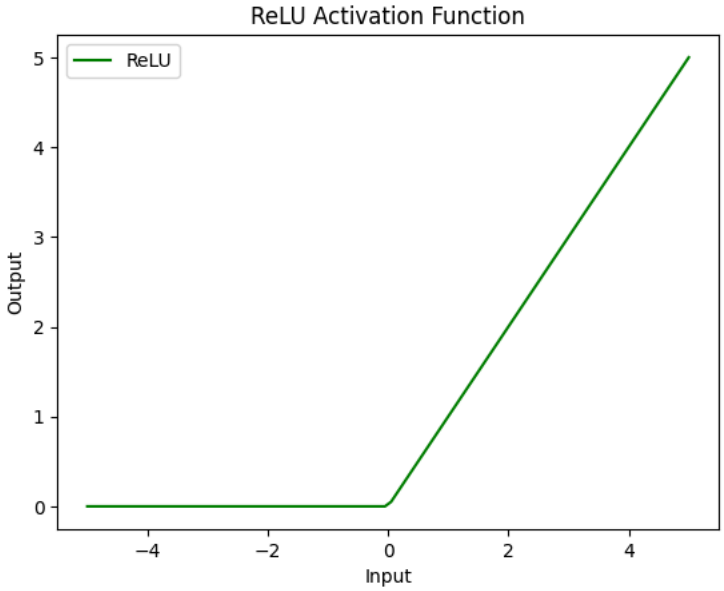
```
#sigmoid Activation function
import numpy as np
def sigmoid(x):
  return 1/(1+np.exp(-x))
x_values=np.linspace(-5,5,100)
y_values=sigmoid(x_values)
import matplotlib.pyplot as plt
plt.plot(x_values,y_values,label='sigmoid')
plt.title('sigmoid activation function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```



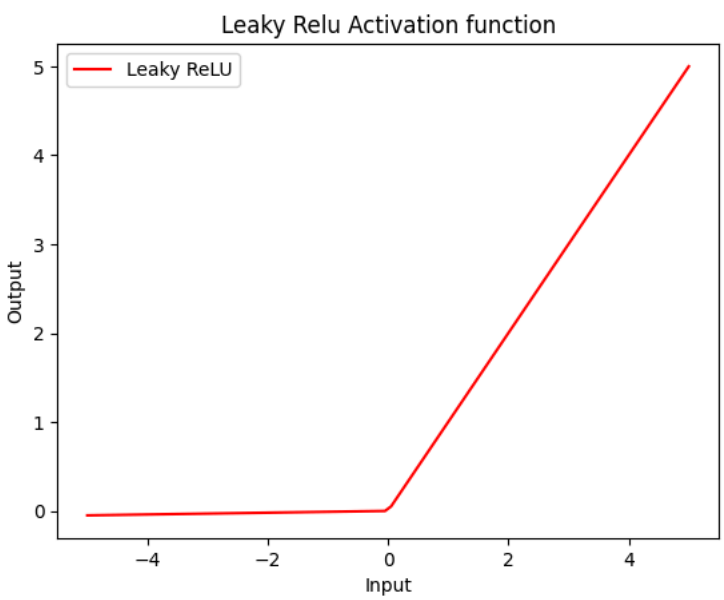

```
#Hyperbolic tangent function
def tanh(x):
  return np.tanh(x)
y_values_tanh=tanh(x_values)
plt.plot(x_values,y_values_tanh,label='tanh',color='orange')
plt.title('tanh activation function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```



```
#Rectified Linear Unit(RELU) activation function
def relu(x):
  return np.maximum(0,x)
y_values_relu=relu(x_values)
plt.plot(x_values,y_values_relu,label='ReLU',color='green')
plt.title('ReLU Activation Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```




```
#Leaky Rectified Linear Unit(Leaky ReLU) Activation fuction
def leaky_relu(x,alpha=0.01):
  return np.where(x>0,x,alpha*x)
y_values_leaky_relu=leaky_relu(x_values)
plt.plot(x_values,y_values_leaky_relu,label='Leaky ReLU',color='red')
plt.title('Leaky Relu Activation function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.legend()
plt.show()
```




```
#Softmax activation function
def softmax(x):
  exp_x=np.exp(x-np.max(x,axis=-1,keepdims=True))
  return exp_x/np.sum(exp_x,axis=-1,keepdims=True)
scores=np.array([2.0,1.0,0.1])
probs=softmax(scores)
print('Softmax Probabilities:',probs)
```

```
Softmax Probabilities: [0.65900114 0.24243297 0.09856589]
```