Name : Shivam Indrabhan Borse

Roll No : 21119

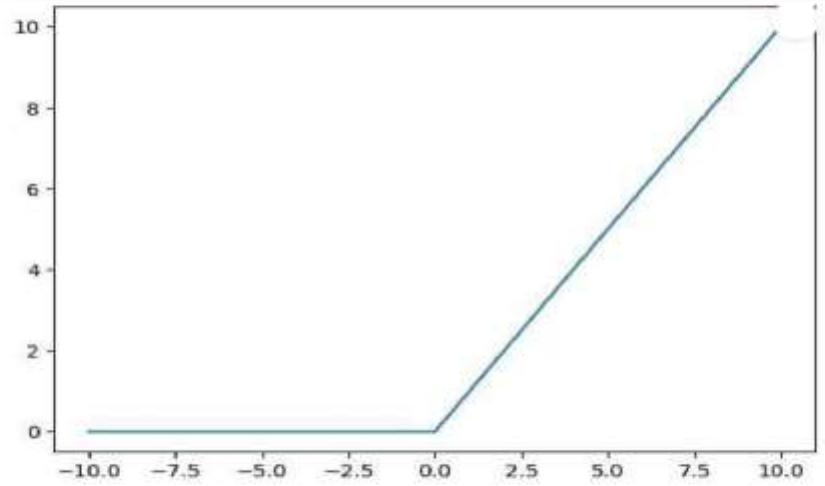
Subject: Software Laboratory II (ANN)

# Assignment No : 01

**Problem statement**: Write a Python program to plot a few activation functions that are being used in neural networks.

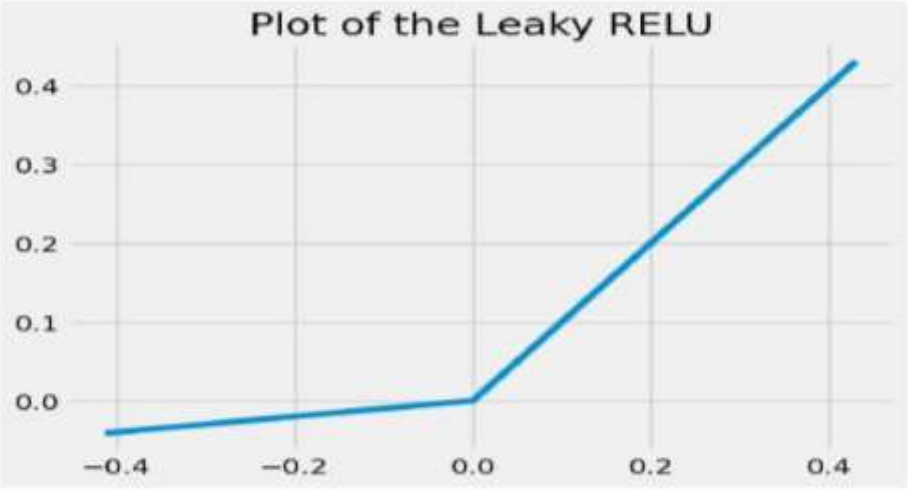
---

In [1]:
```
#Relu
from matplotlib import pyplot

# rectified linear function
def rectified(x):
        return max(0.0, x)

# define a series of inputs
series_in = [x for x in range(-10, 11)]
# calculate outputs for our inputs
series_out = [rectified(x) for x in series_in]
# line plot of raw inputs to rectified outputs
pyplot.plot(series_in, series_out)
pyplot.show()
```




In [3]:
```
def leaky_relu(x):
  if x>0 :
    return x
  else :
    return 0.01*x

x = 1.0
print('Applying Leaky Relu on (%.1f) gives %.1f' % (x, leaky_relu(x)))
x = -10.0
print('Applying Leaky Relu on (%.1f) gives %.1f' % (x, leaky_relu(x)))
x = 0.0
print('Applying Leaky Relu on (%.1f) gives %.1f' % (x, leaky_relu(x)))
x = 15.0
print('Applying Leaky Relu on (%.1f) gives %.1f' % (x, leaky_relu(x)))
x = -20.0
print('Applying Leaky Relu on (%.1f) gives %.1f' % (x, leaky_relu(x)))
```

In [7]:
```
#Leaky relu
from matplotlib import pyplot as plt
import numpy as np

def lrelu_forward(x):
    return np.where(x > 0, x, x * 0.1)


x = np.random.rand(10) - 0.5
x = np.append(x, 0.0)
x = np.sort(x)
y = lrelu_forward(x)

plt.style.use('fivethirtyeight')
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title("Plot of the Leaky RELU")
plt.show()
```

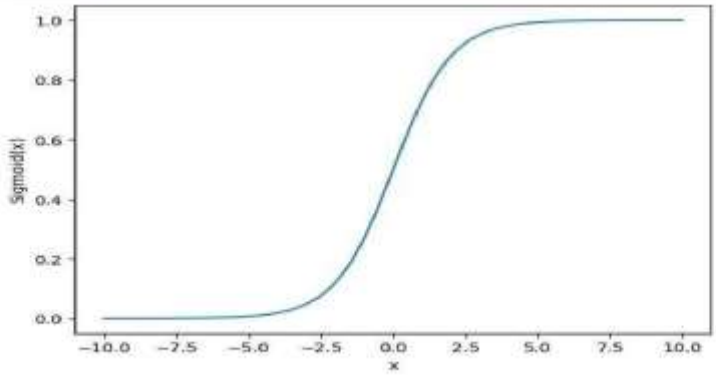



In [5]:
```
import numpy as np
def sig(x):
 return 1/(1 + np.exp(-x))
```

```
x = 1.0
print('Applying Sigmoid Activation on (%.1f) gives %.1f' % (x, sig(x)))

x = -10.0
print('Applying Sigmoid Activation on (%.1f) gives %.1f' % (x, sig(x)))

x = 0.0
print('Applying Sigmoid Activation on (%.1f) gives %.1f' % (x, sig(x)))

x = 15.0
print('Applying Sigmoid Activation on (%.1f) gives %.1f' % (x, sig(x)))

x = -2.0
print('Applying Sigmoid Activation on (%.1f) gives %.1f' % (x, sig(x)))
```

```
Applying Sigmoid Activation on (1.0) gives 0.7
Applying Sigmoid Activation on (-10.0) gives 0.0
Applying Sigmoid Activation on (0.0) gives 0.5
Applying Sigmoid Activation on (15.0) gives 1.0
Applying Sigmoid Activation on (-2.0) gives 0.1
```

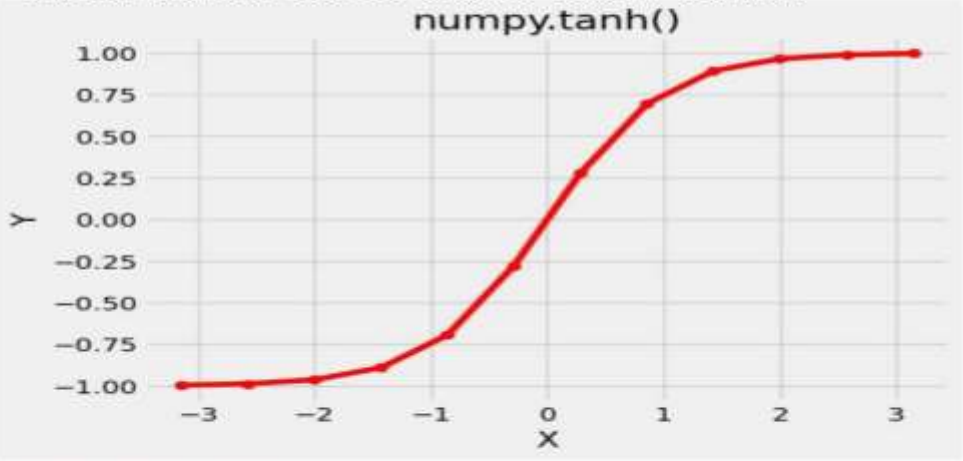
In [6]:
```
#sigmoid
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-10, 10, 50)
p = sig(x)
plt.xlabel("x")
plt.ylabel("Sigmoid(x)")
plt.plot(x, p)
plt.show()
```



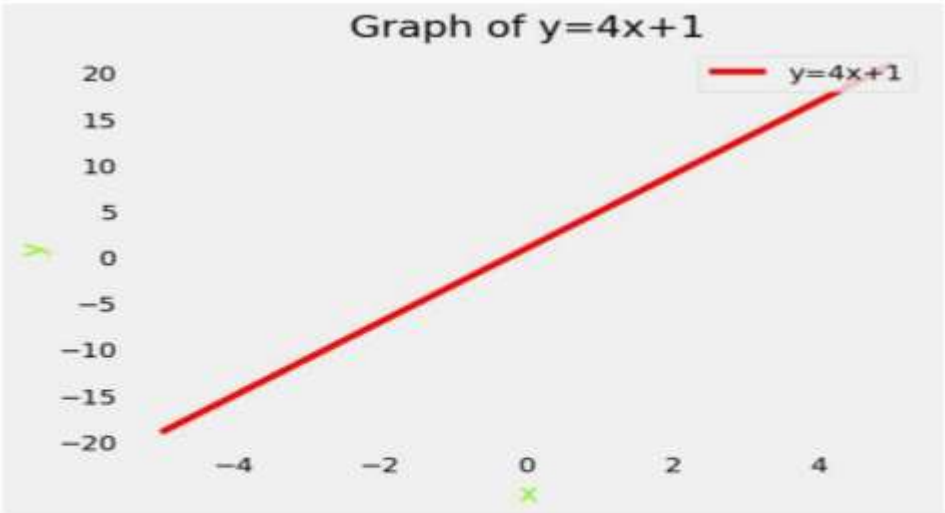

In [8]:
```
# Python program showing Graphical
# representation of tanh() function
```

```
import numpy as np
import matplotlib.pyplot as plt

in_array = np.linspace(-np.pi, np.pi, 12)
out_array = np.tanh(in_array)

print("in_array : ", in_array)
print("\nout_array : ", out_array)

# red for numpy.tanh()
plt.plot(in_array, out_array, color = 'red', marker = "o")
plt.title("numpy.tanh()")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

```
in_array :  [-3.14159265 -2.57039399 -1.99919533 -1.42799666 -0.856798   -0.28559933
  0.28559933  0.856798    1.42799666  1.99919533  2.57039399  3.14159265]

out_array :  [-0.99627208 -0.98836197 -0.96397069 -0.89125532 -0.69460424 -0.27807943
  0.27807943  0.69460424  0.89125532  0.96397069  0.98836197  0.99627208]
```

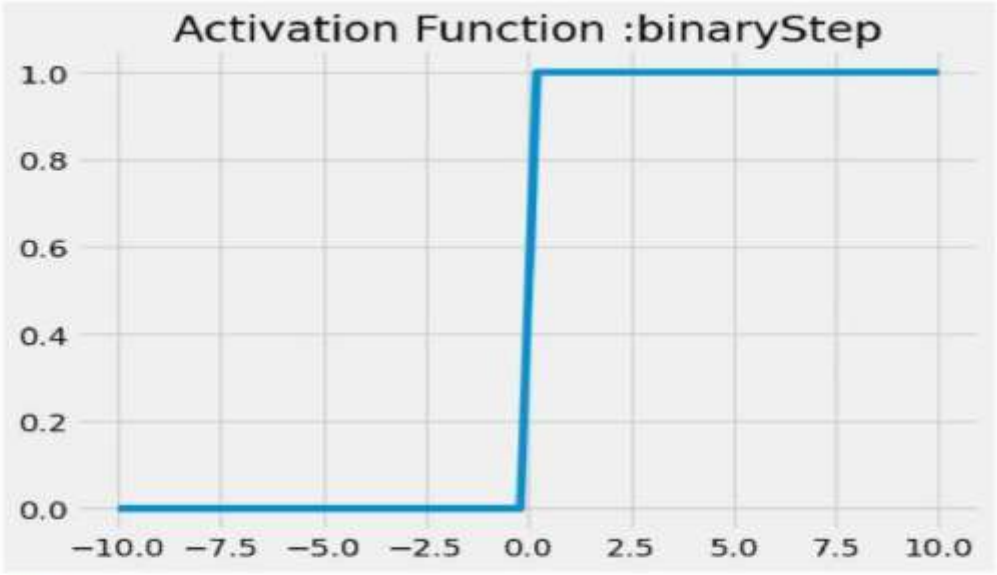



In [24]:
```
#linear activation function
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5,5,150)
y = 4*x+1
plt.plot(x, y, '-r', label='y=4x+1')
plt.title('Graph of y=4x+1')
plt.xlabel('x', color='#7FFF00')
plt.ylabel('y', color='#7FFF00')
plt.legend(loc='upper right')
plt.grid()
plt.show()
```




In [26]:
```
def binaryStep(x):
    ''' It returns '0' is the input is less then zero otherwise it returns one '''
    return np.heaviside(x,1)
```

In [27]:
```
x = np.linspace(-10, 10)
plt.plot(x, binaryStep(x))
plt.axis('tight')
plt.title('Activation Function :binaryStep')
plt.show()
```




In [28]:
```
def softmax(x):
    ''' Compute softmax values for each sets of scores in x. '''
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

In [29]:
```
x = np.linspace(-10, 10)
plt.plot(x, softmax(x))
plt.axis('tight')
plt.title('Activation Function :Softmax')
plt.show()
```