In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
```

In [4]:
```
#Numpy Heaviside
x = np.array([-1.5,2,0, 1])
Out= np.heaviside(x, 0.5)
print(Out)
```

```
[0.  1.  0.5 1. ]
```

In [ ]:
```
```

In [ ]:
```
```

In [5]:
```
#Numpy Linspace
x=np.linspace(0,1,5)
print(x)
```

```
[0.   0.25 0.5  0.75 1.  ]
```

In [ ]:
```
```

In [ ]:
```
```

In [6]:
```
#Function in Python
def ADDER(a,b):
 return a+b

ADDER(4,5)
```

Out[6]: 9

In [ ]:
```
```

In [ ]:
```
```

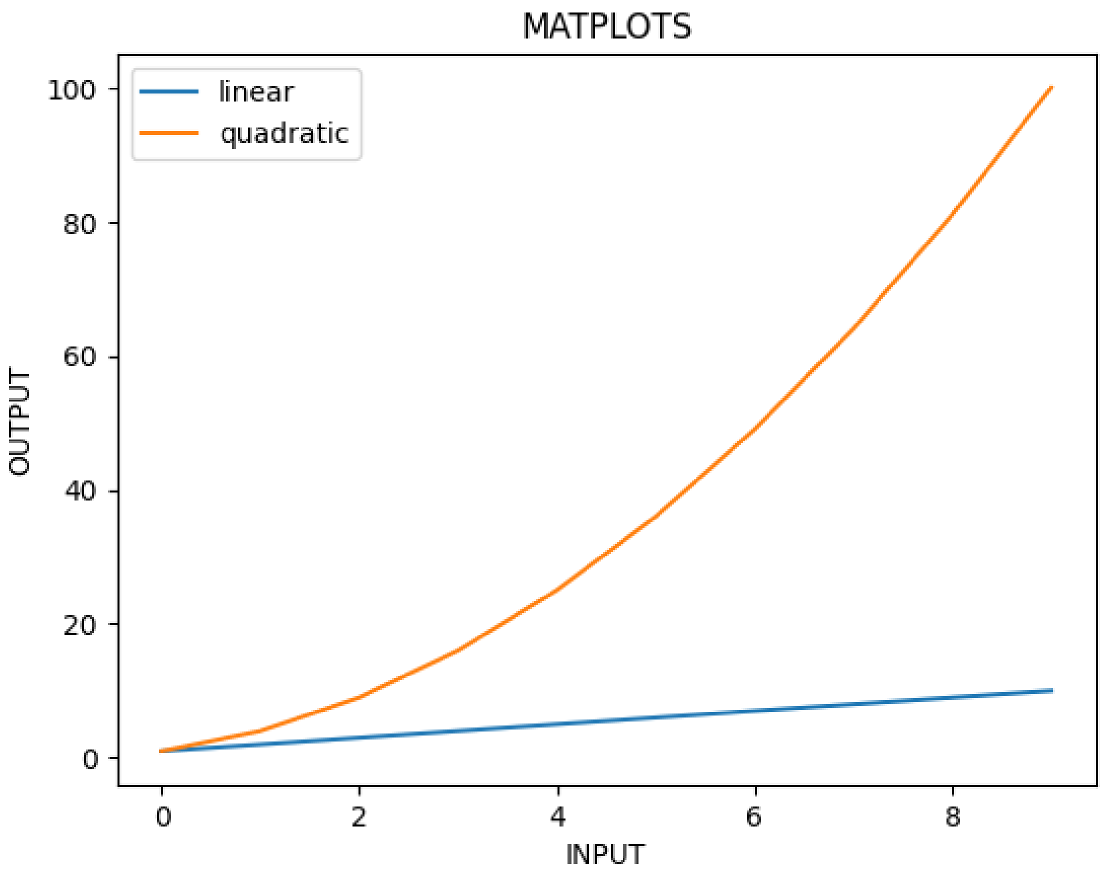
In [8]:
```
#Append to an empty list
list1=[]
list2=[1,2,3,4,5,7,9]
for i in list2:
 if(i%2==0):
     list1.append(i)
print(list1)
```

```
[2, 4]
```

In [ ]:

In [ ]:

In [9]:
```
#Matplot
y1=[1,2,3,4,5,6,7,8,9,10]
y2=[1,4,9,16,25,36,49,64,81,100]
plt.plot(y1)
plt.plot(y2)
plt.legend(["linear", "quadratic"], loc ="upper left")
plt.title("MATPLOTS")
plt.xlabel("INPUT")
plt.ylabel("OUTPUT")
plt.show()
```



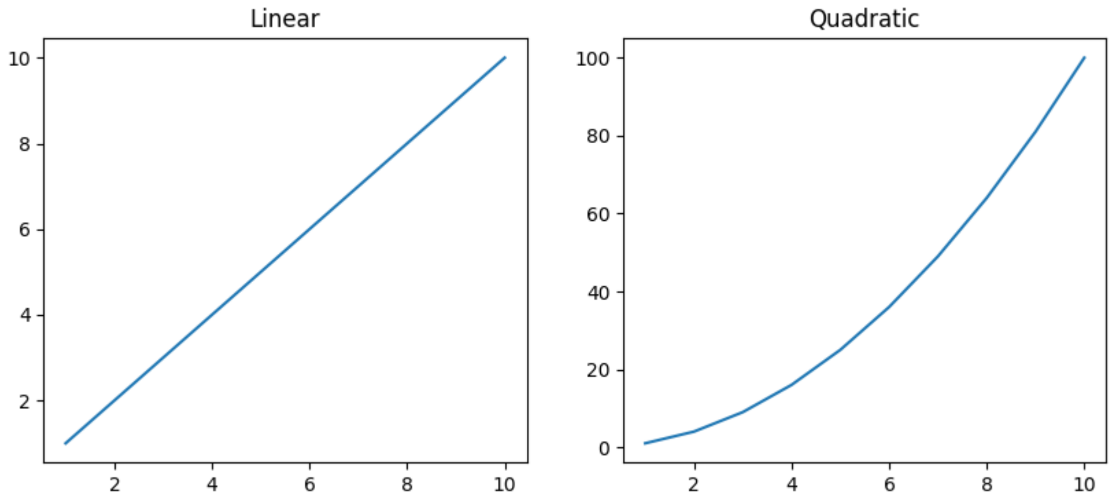
In [ ]:

In [ ]:

In [10]:
```
#MATPLOT SUBLPOTS
x1=[1,2,3,4,5,6,7,8,9,10]

figure, axis = plt.subplots(1,2,figsize=(10, 4))
axis[0].plot(x1,y1)
axis[0].set_title('Linear', fontsize = 12)
axis[1].plot(x1,y2)
axis[1].set_title('Quadratic', fontsize = 12)
```

Out[10]:
```
Text(0.5, 1.0, 'Quadratic')
```

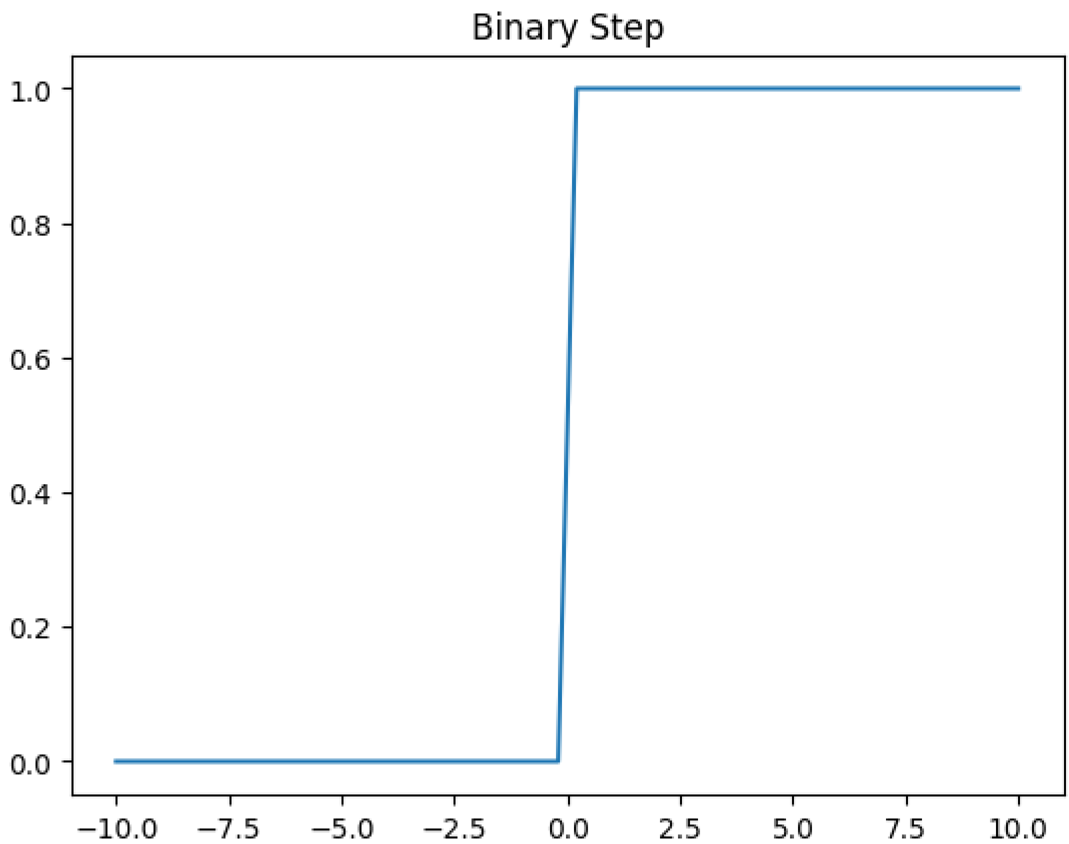


In [ ]:

In [ ]:

In [11]:
```
#1 Binary Step Activation function
def binaryStep(x):
 return np.heaviside(x,1)
```

In [12]:
```
x=np.linspace(-10,10)
plt.plot(x,binaryStep(x))
plt.title("Binary Step")
plt.show()
```

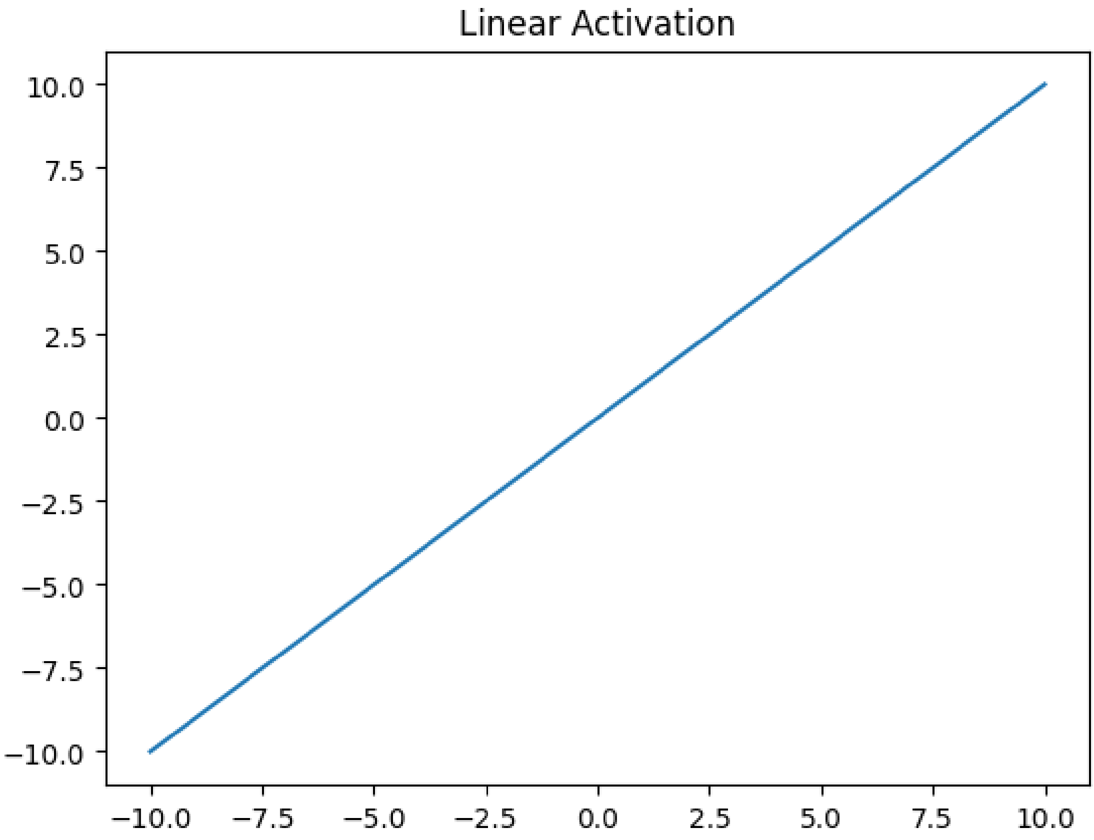


In [ ]:

In [ ]:

In [14]:
```
#2 Linear Activation function
def linearAct(x):
 return x

x=np.linspace(-10,10)
plt.plot(x,linearAct(x))
plt.title('Linear Activation')
plt.show()
```

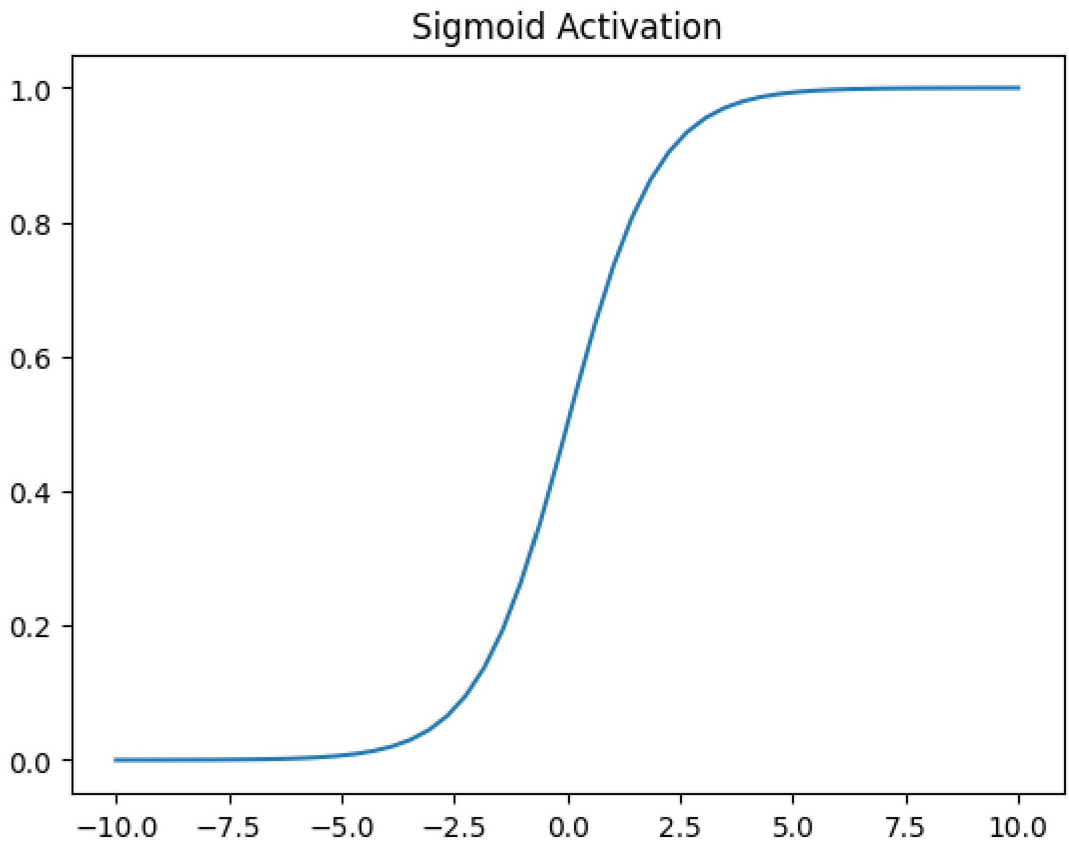


In [ ]:

In [ ]:

In [15]:
```
#3 Sigmoid Activation
def sigmoidAct(x):
 return 1/(1+np.exp(-x))

x=np.linspace(-10,10)
plt.plot(x,sigmoidAct(x))
plt.title('Sigmoid Activation')
plt.show()
```

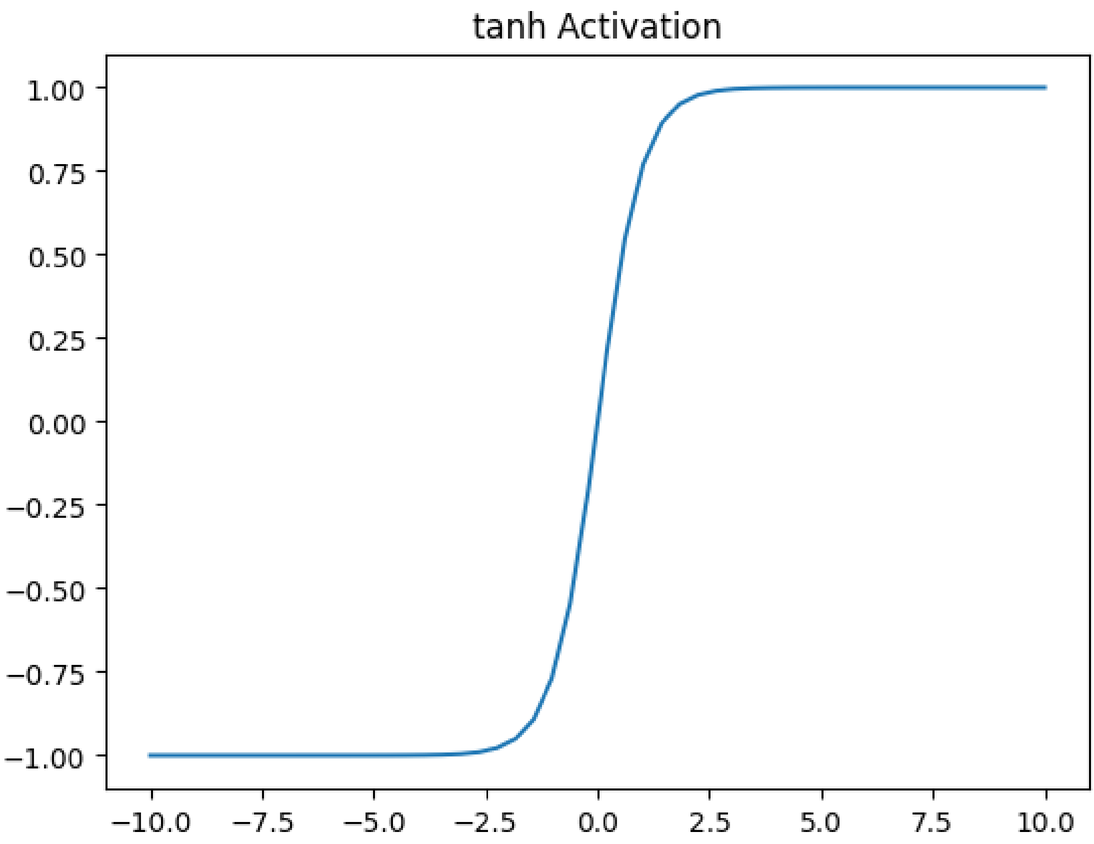


In [ ]:

In [ ]:

In [16]:
```
#4 TanH Activation
def tanhAct(x):
 return np.tanh(x)

x=np.linspace(-10,10)
plt.plot(x,tanhAct(x))
plt.title('tanh Activation')
plt.show()
```



In [ ]:

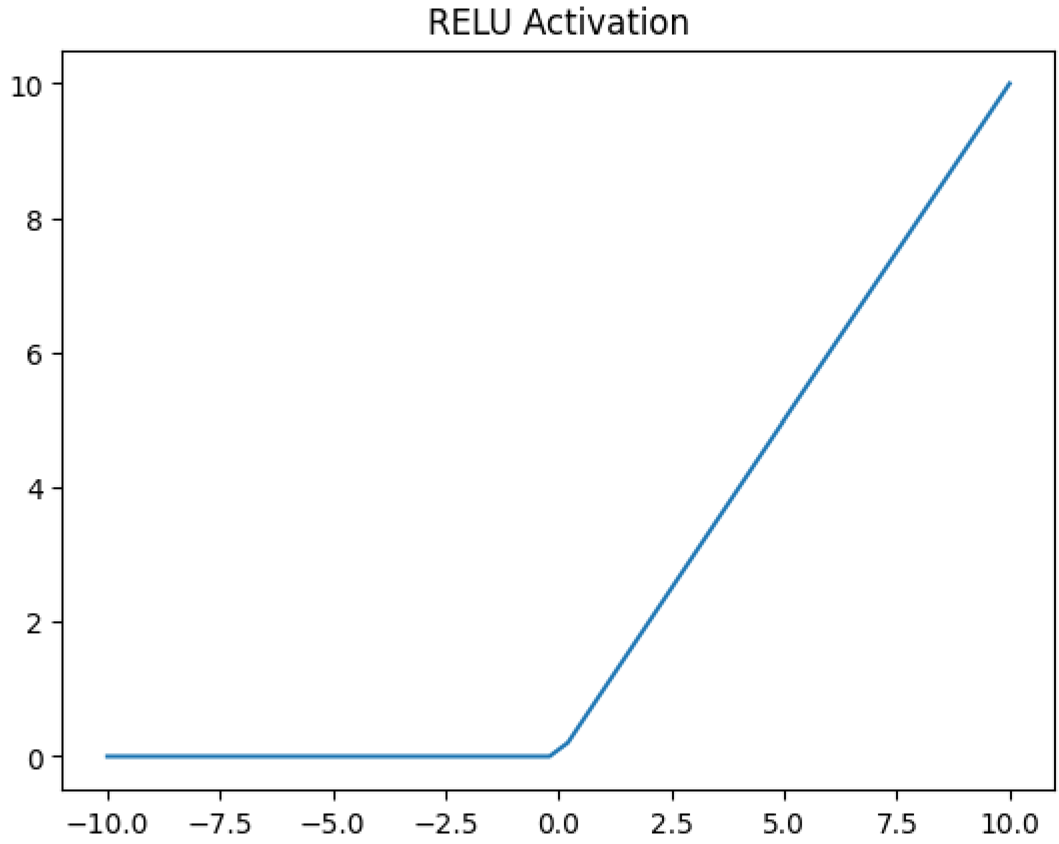
In [ ]:

In [21]:
```
#5 Rectified Linear Unit (ReLU) Activation
def reluAct(x):
    x1=[]
    for i in x:
        if(i<0):
            x1.append(0)
        else:
            x1.append(i)

    return x1
```

In [23]:

```
x=np.linspace(-10,10)
plt.plot(x,reluAct(x))
plt.title('RELU Activation')
plt.show()
```

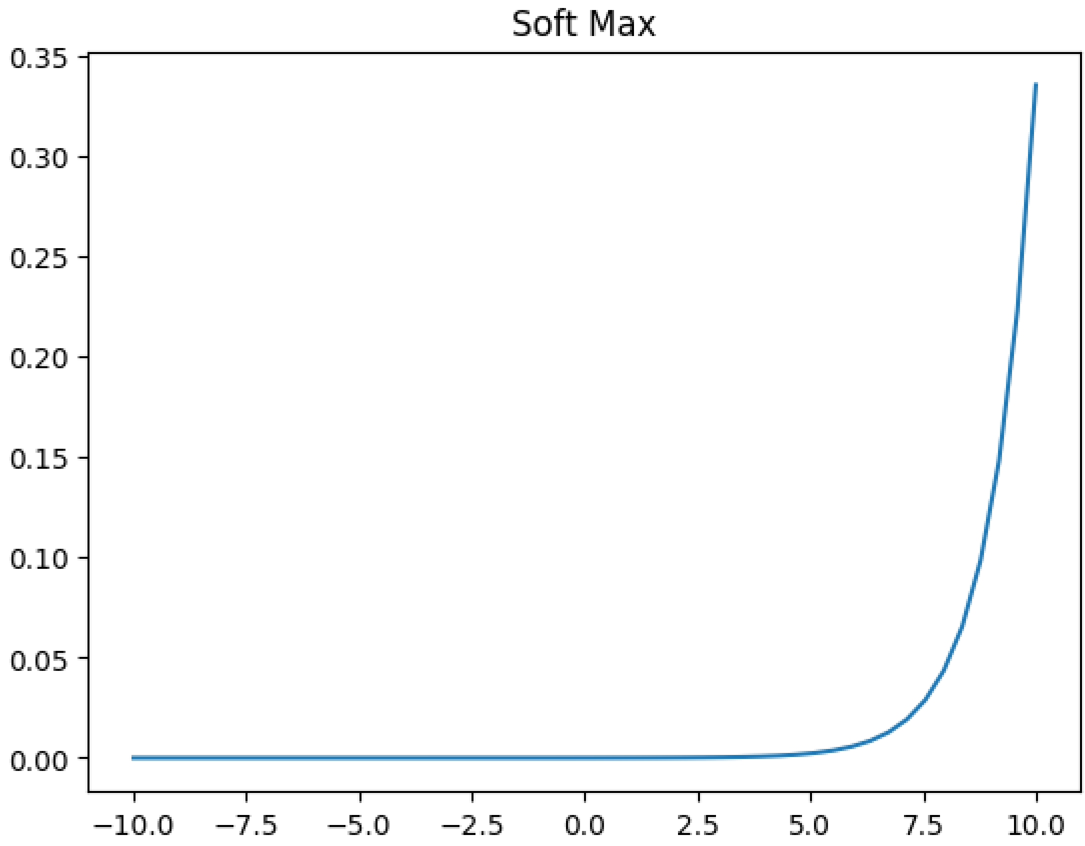


In [ ]:

In [ ]:

In [24]:
```
#6 Softmax Activation
def softmaxAct(x):
 return np.exp(x)/np.sum(np.exp(x))

x=np.linspace(-10,10)
plt.plot(x,softmaxAct(x))
plt.title('Soft Max')
plt.show()
```



In [ ]: