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

In [ ]:
```
#Numpy heaviside
#Function in Python
#List append function
#Numpy linspace function
#Matplot Line plot
#Matplot Subplots
```

Numpy Heaviside

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

In [ ]:
```
print(Out)
```

Numpy Linspace

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

Function in Python

In [ ]:
```
def ADDER(a,b):
    return a+b
```

In [ ]:
```
ADDER(4,5)
```

Append to an empty list

In [ ]:
```
list1=[]
list2=[1,2,3,4,5,7,9]
for i in list2:
    if(i%2==0):
        list1.append(i)
print(list1)
```

In [ ]:
```
#MATPLOT
```

In [ ]:
```
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 [ ]:
```
#MATPLOT SUBLPOTS
```

In [ ]:
```
x1=[1,2,3,4,5,6,7,8,9,10]
y1=[1,2,3,4,5,6,7,8,9,10]
y2=[1,4,9,16,25,36,49,64,81,100]
```

In [ ]:
```
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)
```

1 Binary Step Activation function

In [ ]:
```
def binaryStep(x):
    return np.heaviside(x,1)
```

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

2 Linear Activation function

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

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

3 Sigmoid Activation

In [ ]:
```
def sigmoidAct(x):
    return 1/(1+np.exp(-x))
```

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

In [ ]:
```
#4 TanH Activation
```

In [ ]:
```
def tanhAct(x):
    return np.tanh(x)
```

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

5 Rectified Linear Unit (ReLU) Activation

In [ ]:
```
def reluAct(x):
    x1=[]
    for i in x:
        if(i<0):
            x1.append(0)
        else:
            x1.append(i)
    return x1
```

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

6 Softmax Activation

In [ ]:
```
def softmaxAct(x):
    return np.exp(x)/np.sum(np.exp(x))
```

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

In [ ]:
```

```