Import Libraries

In [40]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Activation Function and it's Derivative

In [41]:
```
# WE need to use same sigmoid function whenever needed

def sigmoid(x):
    return 1/ (1 + np.exp(-x))

def sigmoid_derivative(x):
    s = sigmoid(x)
    return s * (1 -s)
```

Load X and y

In [42]:
```
X = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])
```

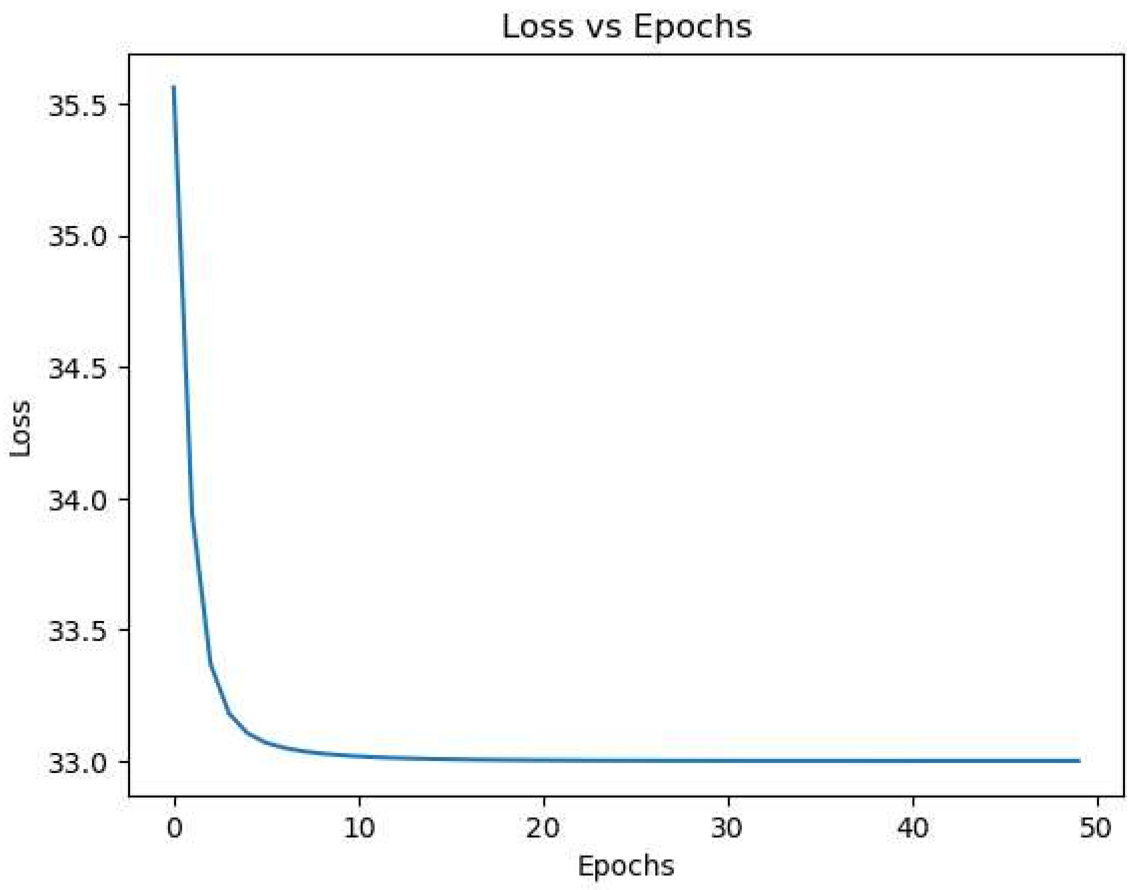
Weight and Bias Initialisation

In [43]:
```
W1 = np.random.rand()
B1 = np.random.rand()

W2 = np.random.rand()
B2 = np.random.rand()
```

In [44]:
```
epoch = 50
learningRate = 0.1
loss_arr = []

for i in range(epoch):
    # Forward pass

    z1 = X*W1 + B1
    a = sigmoid(z1)

    z2 = a*W2 + B2
    y_pred = sigmoid(z2)

    # Calculate loss (Mean Squared Error)
    loss = np.mean((y - y_pred) ** 2)
    print(loss)
    loss_arr.append(loss)

    # Backward pass

    output_error = y_pred - y
    error_output = output_error * sigmoid_derivative(z2)

    hidden_error = error_output * W2
    error_hidden = hidden_error * sigmoid_derivative(z1)

    # Update weights and biases
    W2 = W2 + learningRate * (y - y_pred) * a
    B2 = B2 + learningRate * (y - y_pred)

    W1 = W1 + learningRate * (y - y_pred) * X
    B1 = B1 + learningRate * (y - y_pred)
```

```
35.563250397506884
33.94102795253988
33.37035367201209
33.18112653618077
33.10562956907963
33.06931500478888
33.04902685533888
33.036374879673694
33.027838810468204
33.021747860362424
33.01722543902457
33.01377199258259
33.01108219150133
33.00895787747545
33.007263717246815
33.005903289386886
33.004805518917415
33.00391660746525
33.003195007183784
33.00260814855248
33.00213021927027
33.00174059869694
33.001422717905356
33.00116320709228
33.00095124419984
33.00077804903429
33.000636485485174
33.00052074583832
33.00042609851035
33.000348685426886
33.00028535864664
33.0002335482442
33.000191155229444
33.000156464606185
33.000128074683566
33.00010483953694
33.00008582212917
33.00007025608828
33.00005751452311
33.00004708456775
33.00003854659281
33.000031557221746
33.00002583545099
33.000021151303855
33.00001731655326
33.0000141771347
33.000011606940376
33.00000950274236
33.000007780038814
33.000006369655175
```

In [45]:
```
plt.plot(loss_arr)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Loss vs Epochs")
plt.show()
```



In [46]:
```
print("Predicted Output:")
print(np.round(y_pred, 3))
```

```
Predicted Output:
[1. 1. 1. 1. 1.]
```