Task: 01

Let us consider the task of developing a linear model as follows:

Linear model: $y = w * x$

- In this task, you will create a linear model for given `x_data` and `y_data` (Supervised Learning).
- The main goal of this task is to make you familiar with python and provde a hands-on expereince.
- You have to create functions definitions for the forward-pass and loss-function and write a code snippet to find the optimal value of w
- Eventually we also plot the value of `w` against the difference in the prediction and actual value.

In [9]:

```
#First let us import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
```

Let us make use of a randomly-created sample dataset

In [10]:

```
#sample-dataset
x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]
```

Task: 01 - a

Implement the forward and loss functions

In [11]:

```
#forward function to calculate y_pred for a given x according to the linear mode
l defined above
def forward(x):
    #implement the forward model to compute y_pred as w*x
    ## YOUR CODE STARTS HERE
  y_predicted=w*x
  return y_predicted


    ## YOUR CODE ENDS HERE

#loss-function to compute the mean-squared error between y_pred and y_actual
def loss(y_pred, y_actual):
    #calculate the mean-squared-error between y_pred and y_actual
    ## YOUR CODE STARTS HERE
  loss_=0.5*(y_actual-y_pred)**2
  return loss_
    ## YOUR CODE ENDS HERE
```

Task: 01 - b

Compute the loss for different values of w and identify the w with minimum loss value

In [12]:

```
#initialize wieght and loss lists to monitor
weight_list=[]
loss_list=[]

for w in np.arange(0.0,4.1,0.1): # w can vary between 0 and 4 (both included) wi
th a step-size of 0.1
    print("\nWeight : ", w)
    total_loss=0
    count = 0
    for x, y in zip (x_data, y_data):
        #call the forward and loss functions to compute the loss value for the g
iven data-point pair
        ## YOUR CODE STARTS HERE
        y_predicted=forward(x)
        current_loss=loss(y_predicted,y)
        ## YOUR CODE ENDS HERE
        total_loss += current_loss
        count += 1

    #calculate the average mse-loss across three samples in our dataset
    ## YOUR CODE STARTS HERE
    avg_mse=total_loss/count

    ## YOUR CODE ENDS HERE

    print("Average Mean Squared Error : ", avg_mse)
    weight_list.append(w)
    loss_list.append(avg_mse)
```

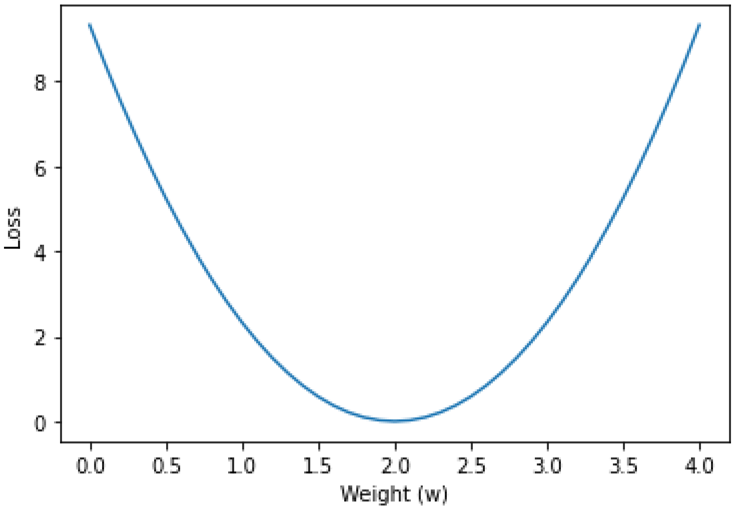
```
Weight :  0.0
Average Mean Squared Error :  9.333333333333334

Weight :  0.1
Average Mean Squared Error :  8.423333333333334

Weight :  0.2
Average Mean Squared Error :  7.560000000000001

Weight :  0.30000000000000004
Average Mean Squared Error :  6.743333333333332

Weight :  0.4
Average Mean Squared Error :  5.973333333333334

Weight :  0.5
Average Mean Squared Error :  5.25

Weight :  0.6000000000000001
Average Mean Squared Error :  4.5733333333333315

Weight :  0.7000000000000001
Average Mean Squared Error :  3.943333333333333

Weight :  0.8
Average Mean Squared Error :  3.3599999999999994

Weight :  0.9
Average Mean Squared Error :  2.823333333333333

Weight :  1.0
Average Mean Squared Error :  2.3333333333333335

Weight :  1.1
Average Mean Squared Error :  1.8899999999999995

Weight :  1.2000000000000002
Average Mean Squared Error :  1.4933333333333325

Weight :  1.3
Average Mean Squared Error :  1.1433333333333329

Weight :  1.4000000000000001
Average Mean Squared Error :  0.8399999999999997

Weight :  1.5
Average Mean Squared Error :  0.5833333333333334

Weight :  1.6
Average Mean Squared Error :  0.373333333333333

Weight :  1.7000000000000002
Average Mean Squared Error :  0.20999999999999974

Weight :  1.8
Average Mean Squared Error :  0.09333333333333325

Weight :  1.9000000000000001
Average Mean Squared Error :  0.023333333333333293

Weight :  2.0
```

```
Average Mean Squared Error :  0.0

Weight :  2.1
Average Mean Squared Error :  0.023333333333333418

Weight :  2.2
Average Mean Squared Error :  0.09333333333333349

Weight :  2.3000000000000003
Average Mean Squared Error :  0.21000000000000027

Weight :  2.4000000000000004
Average Mean Squared Error :  0.37333333333333396

Weight :  2.5
Average Mean Squared Error :  0.5833333333333334

Weight :  2.6
Average Mean Squared Error :  0.8400000000000004

Weight :  2.7
Average Mean Squared Error :  1.1433333333333346

Weight :  2.8000000000000003
Average Mean Squared Error :  1.493333333333334

Weight :  2.9000000000000004
Average Mean Squared Error :  1.8900000000000015

Weight :  3.0
Average Mean Squared Error :  2.3333333333333335

Weight :  3.1
Average Mean Squared Error :  2.823333333333334

Weight :  3.2
Average Mean Squared Error :  3.3600000000000017

Weight :  3.3000000000000003
Average Mean Squared Error :  3.943333333333334

Weight :  3.4000000000000004
Average Mean Squared Error :  4.573333333333335

Weight :  3.5
Average Mean Squared Error :  5.25

Weight :  3.6
Average Mean Squared Error :  5.973333333333335

Weight :  3.7
Average Mean Squared Error :  6.743333333333337

Weight :  3.8000000000000003
Average Mean Squared Error :  7.560000000000002

Weight :  3.9000000000000004
Average Mean Squared Error :  8.423333333333336

Weight :  4.0
Average Mean Squared Error :  9.333333333333334
```

Visualize the logs ¶

In [13]:

```
plt.plot(weight_list, loss_list)
plt.ylabel('Loss')
plt.xlabel('Weight (w)')
plt.show()
```



In [ ]:

```
# !pip install nbconvert
# !sudo apt-get install texlive-xetex texlive-fonts-recommended texlive-plain-ge
neric
# !jupyter nbconvert --to html "/content/drive/MyDrive/Colab Notebooks/Task-01_E
E21S060.ipynb"
```