In [1]:
```
import numpy as np
```

In [2]:
```
#Gradient descent for linear regression
# Yhat = wx + b
# loss = (Y-Yhat)**2 / N

#initialize parameters

#data to train the model
x = np.random.randn(10,1)
y = 2*x+np.random.randn()

#assign values to the parameters
w = 0.0
b = 0.0

#assign hyperparameter -- must be a number except 0.0
learning_rate = 0.01

#create gradient descent function
def descend(x, y, w, b, learning_rate):
    dldw = 0.0
    dldb = 0.0
    N = x.shape[0]

    #loss = (y-yhat)**2 / N or (y-wx+b)**2 / N
    #find partial derivatives

    for xi, yi in zip(x, y):

        #use chain rule:
        dldw = -2*xi*(yi-(w*xi+b))
        dldb = -2*(yi-(w*xi+b))

    #update the w and b parameters:
    w = w-learning_rate*(1/N)*dldw
    b = b-learning_rate*(1/N)*dldb

    return w, b

#iteratively make updates
for epoch in range(400):
    w, b = descend(x,y,w,b,learning_rate)
    yhat = (w*x + b)
    loss = np.divide(np.sum((y-yhat)**2, axis=0), x.shape[0])
    print(f'{epoch} loss is {loss}, parameters w:{w}, b:{b}')
```