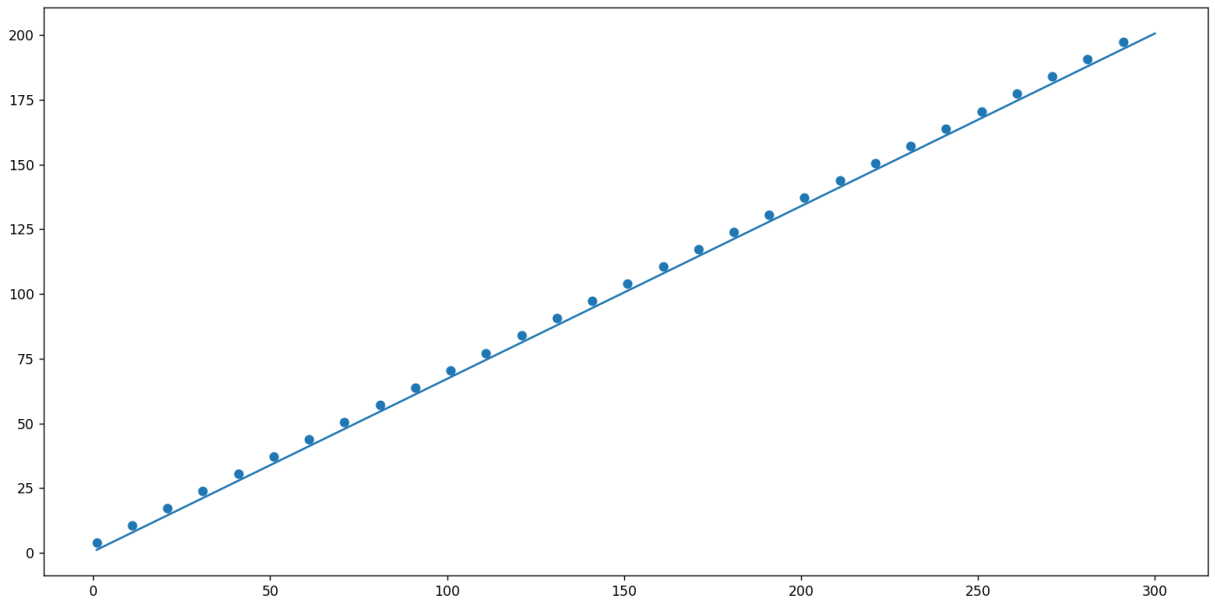
LR using numpy

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Training Data
df = pd.read_csv('Linear Regression - Sheet1.csv')
x_train = df.X.to_numpy()
y_train = df.Y.to_numpy()

# What is the logic and breaking it down into steps-
# step-1 : define the model
# step-2 : compute the cost function
# step-3 : Perform Gradient Descent
#         - simultaneously update w and b
#         - also need to update their gradients
#         - step-3 should be performed multiple times

# Defining the model -
def f(w,b,x):
  return w*x + b

def compute_cost(w,b,x,y):
  J = (((f(w,b,x)-y)**2).mean())*0.5
  return J

def compute_grad(w,b,x,y):
  dJ_dw = ((f(w,b,x)-y)*x).mean()
  dJ_db = (f(w,b,x)-y).mean()
  return dJ_dw, dJ_db

def train(w,b,x,y,epochs,lr):
  for i in range(epochs):
    dJ_dw, dJ_db = compute_grad(w,b,x,y)
    w_temp = w - lr*(dJ_dw)
    b_temp = b - lr*(dJ_db)
    w = w_temp
    b = b_temp
    print(f"epoch {i+1} : cost {compute_cost(w,b,x,y)}")
  print(f"final w : {w}, final b : {b}")
  return f(w,b,x)

# Lets initialize w and b
w = 0.5
b = 0.5

# Plotting the result
y_pred = train(w,b,x_train,y_train,500,0.000015)
plt.plot(x_train, y_pred, label='prediction')
plt.scatter(x_train[::10], y_train[::10],label='actual data')
plt.show()
```

Plot -


200
175
150
125
100
75
50
25
0
0
50
100
150
200
250
300