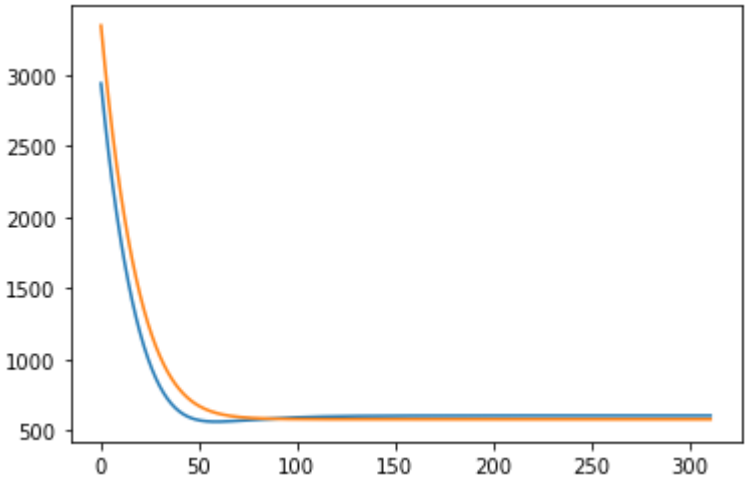
In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split, KFold
data = pd.read_csv('x06Simple.csv')

#1. Reads in the data, ignoring the first row (header) and first column (index).
y = data.to_numpy()
np.random.seed(0)
#2. Randomizes the data
np.random.shuffle(y)
size = np.size(y[:,1])
#3. Selects the first 2/3 (round up) of the data for training and the remaining for testing
x = math.ceil(size*(2/3))

Yan = np.array(y[0:x,[3]]) #2/3 train data
yanTest = np.array(y[x:,[3]]) #1/3 test data

mat = np.array(y[:,1:3])
one = np.ones((size,1))
mat = np.concatenate((one, mat), axis=1)
mat1= mat[:,[1,2]]

#4. Standardizes the data (except for the last column of course) base on the training data
mean2 = np.mean(mat1 ,axis=0)
std2 = np.std(mat1, axis=0,ddof=1)

mat1 = (mat1-mean2)/std2
mat[:,[1,2]]=mat1
mat2= np.array(mat[0:x,:])
mat_test = np.array(mat[x:,:])
N = len(mat_test)


itter = 1000
learn_rate=0.01
theta = np.zeros((3,1))
for j in range(3):
    theta[j]=(np.random.uniform(low=-1, high=1, size=(1,)))
```

```
y = np.zeros((itter,1))

change = float(3.000)
meanold = float(3.000)
i = 0

RMSET = []
RMSETrain = []
#5. While the termination criteria (mentioned above in the implementation details) hasn't been met
while(i < itter and change > 2**-23):
    #(c) Update each parameter using batch gradient descent
    dir1 = 2*mat2.T @ ((mat2@theta)-Yan)
    theta = theta - (learn_rate/N*dir1)
    y1 = mat_test@theta
    yTrain = mat2@theta
    #(a) Compute the RMSE of the training data

    rmeantemp = np.sqrt(mean_squared_error(yanTest, y1))
    RMSET.append(rmeantemp)
    if(i>1):
        change = np.abs((meanold-rmeantemp)/rmeantemp)
    meanold = rmeantemp

    #(b) While we can't let the testing set affect our training process,
    #also compute the RMSE of the testing error at each iteration of the algorithm (it'll be interesting to see)
    RMSETrain.append(np.sqrt(mean_squared_error(Yan, yTrain)))
    i += 1


RMSET = np.array(RMSET)
RMSETrain = np.array(RMSETrain)

#6.Compute the RMSE of the testing data.
print(RMSET[-1])
plt.plot(range(np.size(RMSET)),RMSET)
plt.plot(range(np.size(RMSETrain)),RMSETrain)
print("Y = {0}+{1}*x1-{2}*x2".format(theta[0],theta[1],theta[2]))
```

```
601.9286312069191
Y = [3180.48226339]+[1150.04415084]*x1-[-265.08116507]*x2
```



In [ ]:

In [ ]: