**Code:**

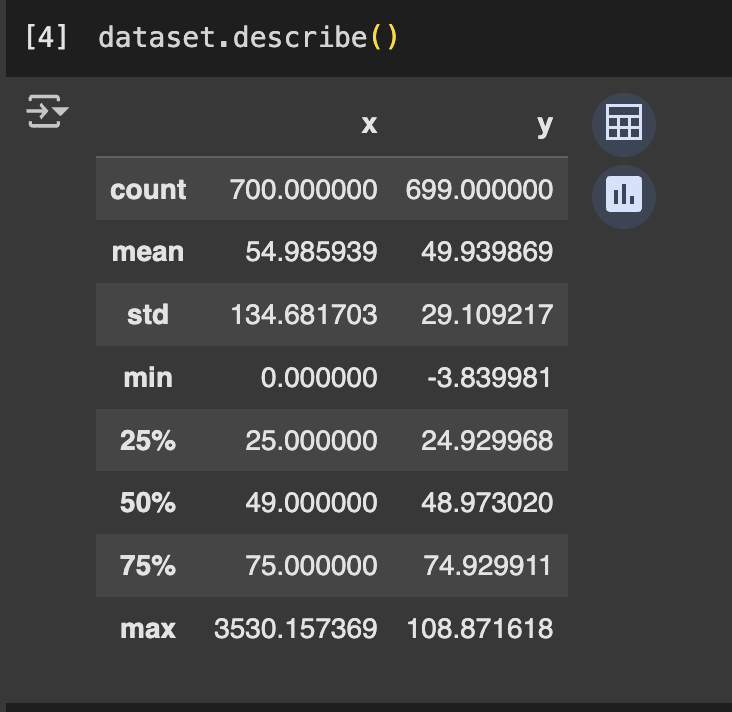
import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

dataset=pd.read\_csv("/content/exp-1\_train.csv")

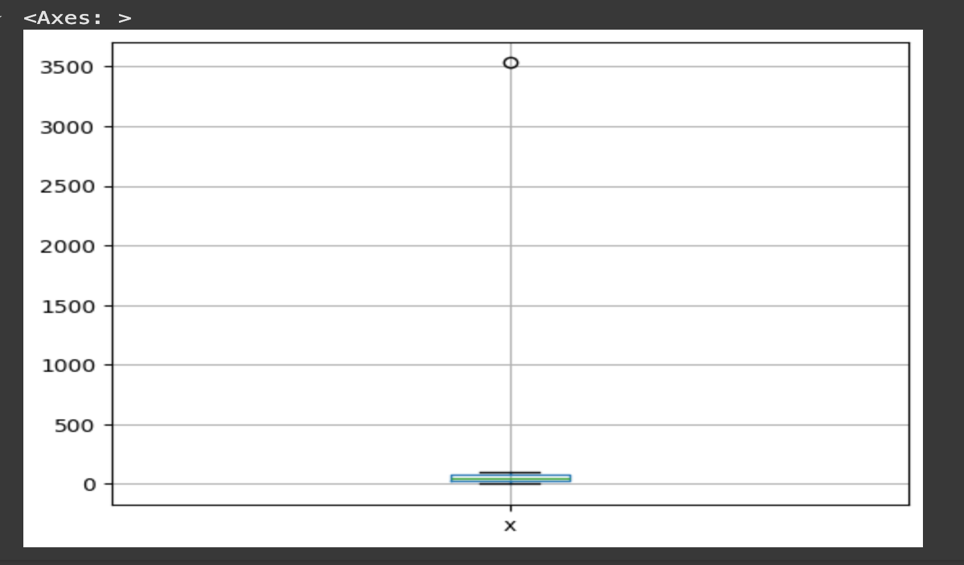
dataset.describe()



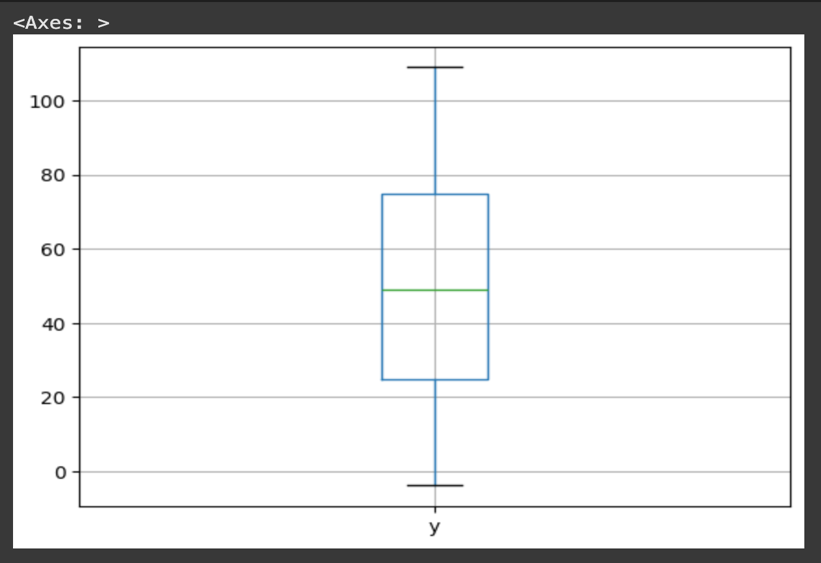
x=dataset.iloc[0:700,0:1]

y=dataset.iloc[0:700,1:2]

x.boxplot(column=['x'])



y.boxplot(column=['y'])



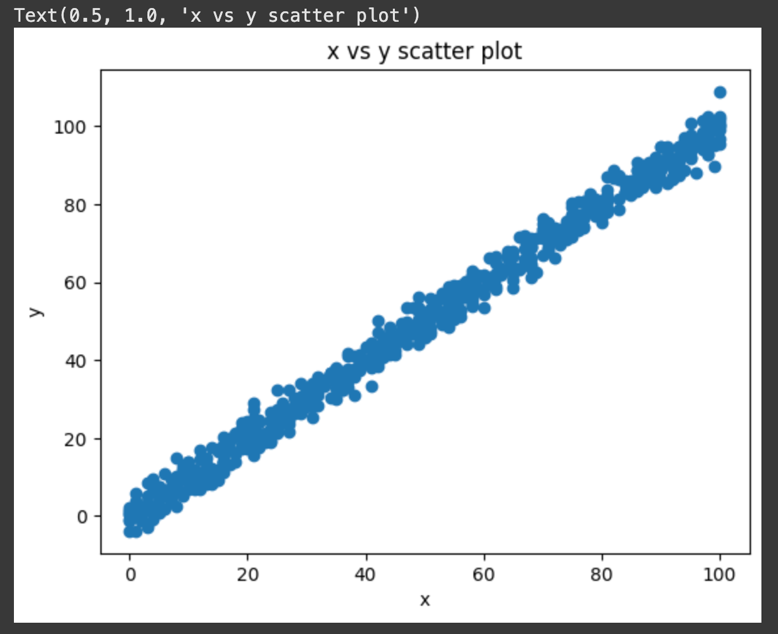
#plot the scatter plot

plt.scatter(x,y)

plt.xlabel('x')

plt.ylabel('y')

plt.title('x vs y scatter plot')



#linear regression

def hypothesis(theta\_array,x):

return theta\_array[0]+theta\_array[1]\*x

def Cost\_Function(theta\_array,x,y , m):

error=0

for i in range(m):

error=error+(hypothesis(theta\_array, x[i])-y[i])\*\*2 # Use hypothesis function

return error/(2\*m)

def Gradient\_Descent(theta\_array , x, y , m ,alpha) :

summation\_0 = 0

summation\_1 = 0

for i in range(m):

prediction = hypothesis(theta\_array, x[i]) # Use hypothesis function

summation\_0 += (prediction - y[i])

summation\_1 += x[i]\*(prediction - y[i])

new\_theta0 = theta\_array[0] - (alpha/m)\*summation\_0

new\_theta1 = theta\_array[1] - (alpha/m)\*summation\_1

updated\_new\_theta = [new\_theta0 , new\_theta1]

return updated\_new\_theta

def Training(x, y, alpha, iters):

theta\_0 = 0

theta\_1 = 0

cost\_values = []

theta\_array = [theta\_0, theta\_1]

m=x.size

for i in range(iters):

theta\_array = Gradient\_Descent(theta\_array, x, y, m, alpha)

cost\_values.append(Cost\_Function(theta\_array, x, y, m))

return theta\_array, cost\_values # Return theta\_array and cost\_values

#feesing the input data

Training\_data=dataset.dropna()

Training\_data.shape

x\_value=Training\_data['x']

y\_value=Training\_data['y']

type(x\_value)

x\_value=x\_value.values.reshape(x\_value.size)

y\_value=y\_value.values.reshape(y\_value.size)

type(x\_value)

alpha = 0.0001

iters = 50

theta\_array, cost\_values = Training(x\_value, y\_value, alpha, iters)

x\_axis = np.arange(0, len(cost\_values), step=1)

plt.plot(x\_axis, cost\_values)

plt.xlabel("Iterations")

plt.ylabel("Cost Values")

plt.title("Loss Graph")

plt.show()

