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

```
dataset=pd.read_csv("/content/exp-1_train.csv")
```

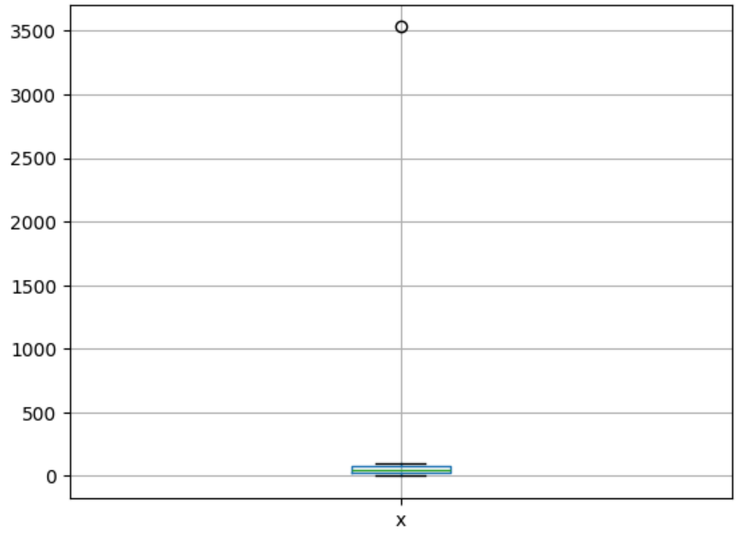
```
dataset.describe()
```

	x	y
count	700.000000	699.000000
mean	54.985939	49.939869
std	134.681703	29.109217
min	0.000000	-3.839981
25%	25.000000	24.929968
50%	49.000000	48.973020
75%	75.000000	74.929911
max	3530.157369	108.871618

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

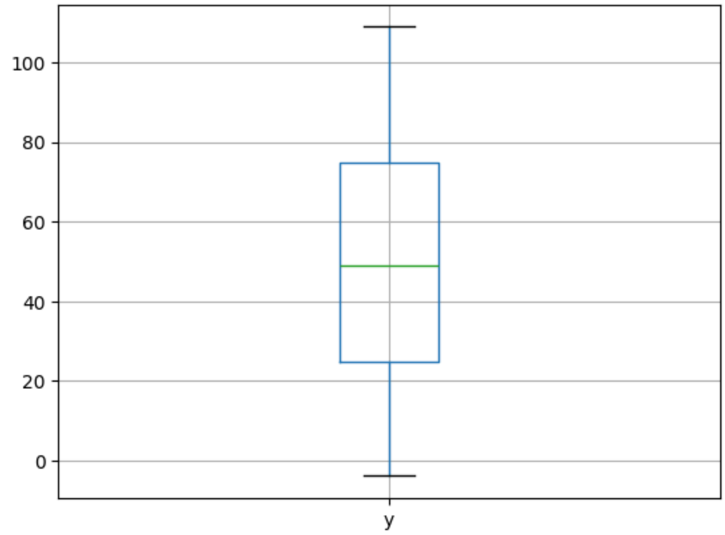
```
x.boxplot(column=['x'])
```

```
<Axes: >
```



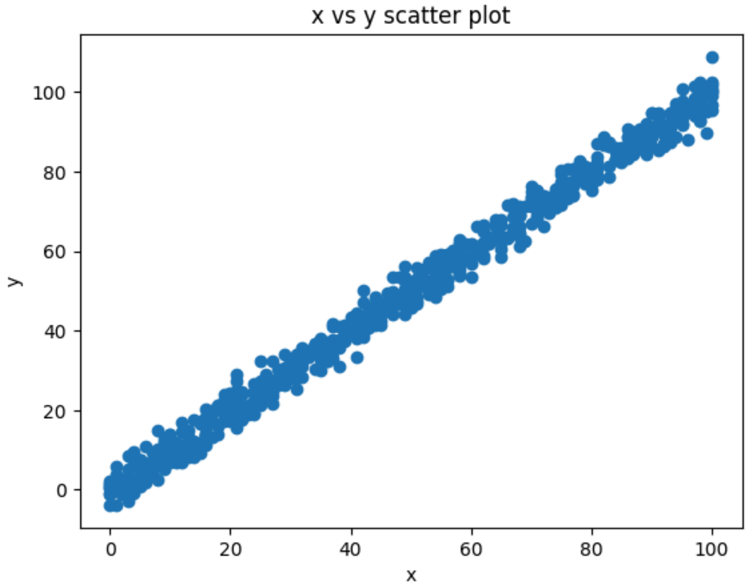
```
y.boxplot(column=['y'])
```

```
<Axes: >
```



```
#plot the scatter plot
plt.scatter(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('x vs y scatter plot')
```

```
Text(0.5, 1.0, '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
```

```
(699, 2)
```

```
x_value=Training_data['x']
y_value=Training_data['y']
```

```
type(x_value)
```

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool |
None=None, fastpath: bool | lib.NoDefault=lib.no_default) -> None

Operations between Series (+, -, /, \*, \*\*) align values based on their
associated index values-- they need not be the same length. The result
index will be the sorted union of the two indexes.

Parameters
----------
```

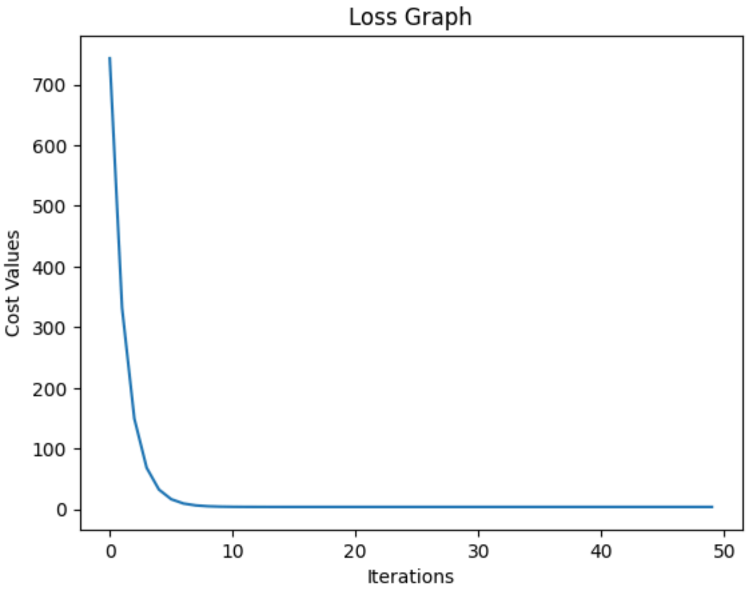
```
x_value=x_value.values.reshape(x_value.size)
y_value=y_value.values.reshape(y_value.size)
```

```
type(x_value)
```

```
numpy.ndarray
```

```
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()
```


Loss Graph
Cost Values
Iterations