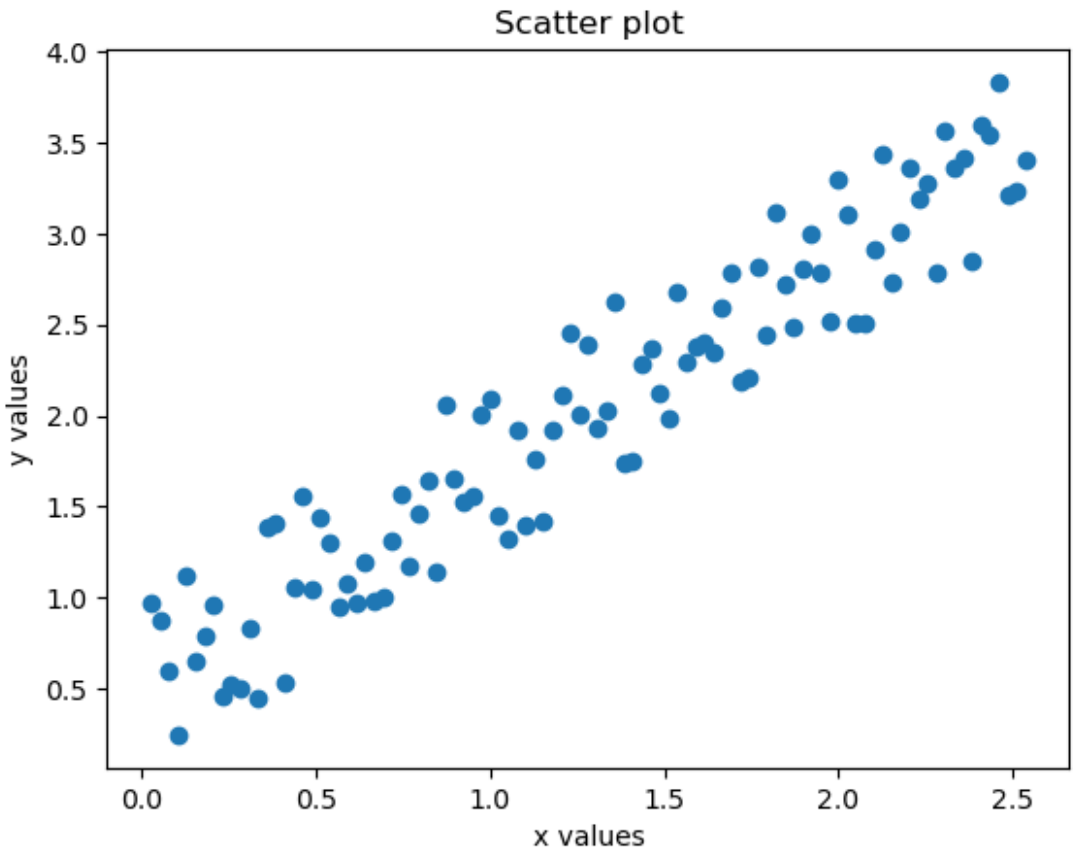
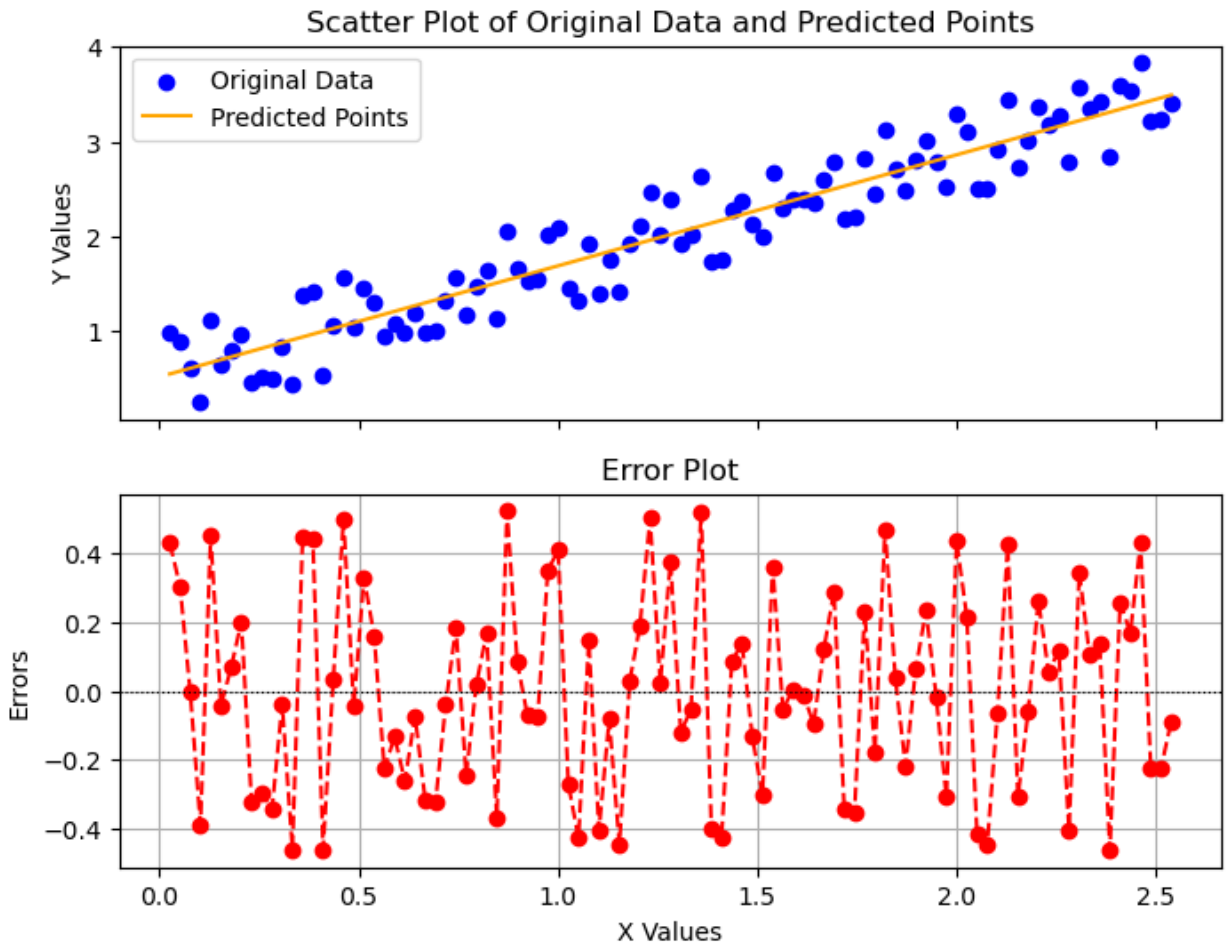
```
import numpy as np
import csv
import matplotlib.pyplot as plt
import scipy
from io import StringIO
datar=[]

%matplotlib inline
with open("linear-data-set-for-regression.csv", "r") as csvfile:
    csvreader=csv.reader(csvfile)
    for r in csvreader:
        datar.append(r)
data=np.array(datar)
yval=data[1:,0].astype(float)
xval=data[1:,1].astype(float)

plt.scatter(xval, yval)
plt.title("Scatter plot")
plt.ylabel("y values")
plt.xlabel("x values")
plt.show()

def xybar(yi, xi):
    count = 0
    prod_xy = 0
    for i in range(len(yi)):
        prod_xy += yi[i] * xi[i]
        count += 1

    return prod_xy / count

def avg(lst):   # This function calculates the average of a list. Will
be used for xbar and ybar
    total = sum(lst)
    count = len(lst)

    if count == 0:
        return 0

    mean = total / count
    return mean

def avg_of_square(lst):
    if len(lst) == 0:
        return 0

    square_sum = sum(x ** 2 for x in lst)
    count = len(lst)
    average = square_sum / count
```

```
    return average
a = (xybar(yval, xval) - avg(xval) * avg(yval)) / (avg_of_square(xval)
- avg(xval) ** 2)
b = (avg(yval) * avg_of_square(xval) - avg(xval) * xybar(yval, xval))
/ (avg_of_square(xval) - avg(xval) ** 2)
print(f"Coefficients: a = {a}, b = {b}")
# Function to calculate the predicted values ycap = a * x + b
def calc_ycap(x, a, b):
    y_cap = [a * xi + b for xi in x]
    return y_cap

ycap = calc_ycap(xval, a, b)    # Predicted values, based on the
regression line
e = [yval[i] - ycap[i] for i in range(len(yval))]
print(scipy.stats.normaltest(e, axis=0, nan_policy='propagate'))
sst=0.0
sse=0.0
ssr=0.0
for i in range(len(yval)):
    sst=sst+(yval[i]-avg(yval))**2
    ssr=ssr+(ycap[i]-avg(yval))**2
    sse=sse+(yval[i]-ycap[i])**2
r2=ssr/sst
print(f"ssr+sse=",ssr+sse)
print(f"sst=",sst)
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(8, 6))
# Subplot 1: Scatter plot of original data + predicted points
ax1.scatter(xval, yval, color='blue', label='Original Data')
ax1.plot(xval, ycap, color='orange', label='Predicted Points')
ax1.set_ylabel('Y Values')
ax1.set_title('Scatter Plot of Original Data and Predicted Points')
ax1.legend()
# Subplot 2: Error plot
ax2.plot(xval, e, color='red', marker='o', linestyle='dashed')
ax2.axhline(y=0, color='black', linewidth=0.8, linestyle='dotted')
ax2.set_xlabel('X Values')
ax2.set_ylabel('Errors')
ax2.set_title('Error Plot')
ax2.grid()
```




```
Coefficients: a = 1.171951180445917, b = 0.5156466449319407
NormaltestResult(statistic=17.52189454756286,
pvalue=0.000156736066276324)
ssr+sse= 81.10703596594016
sst= 81.10703596594
```



Results

The results I got from excel last week were: a= 1.171951, b=0.515647, sst=81.10703597.

As we can see from the results I have got, the values match to a high degree of precision.