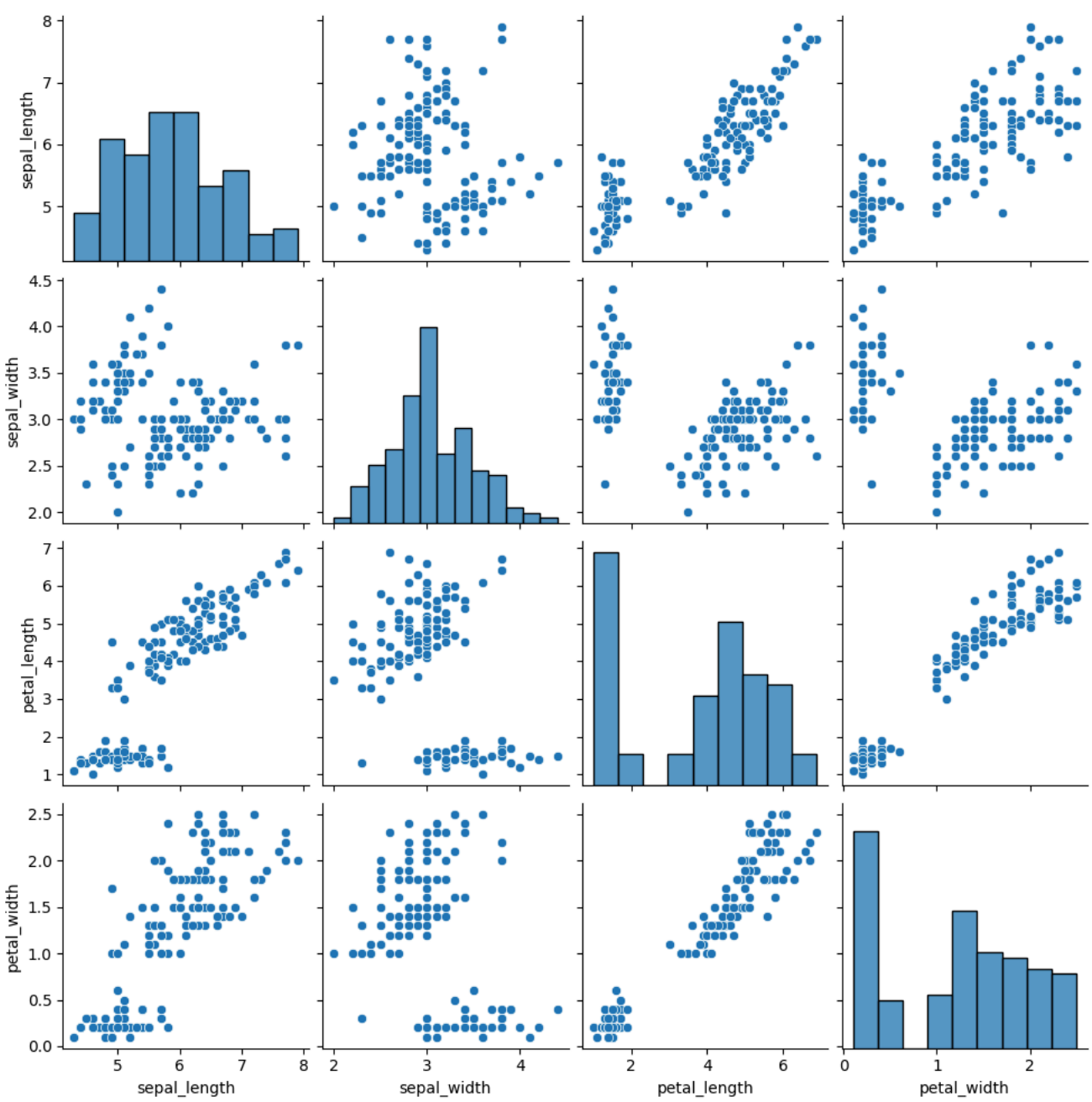
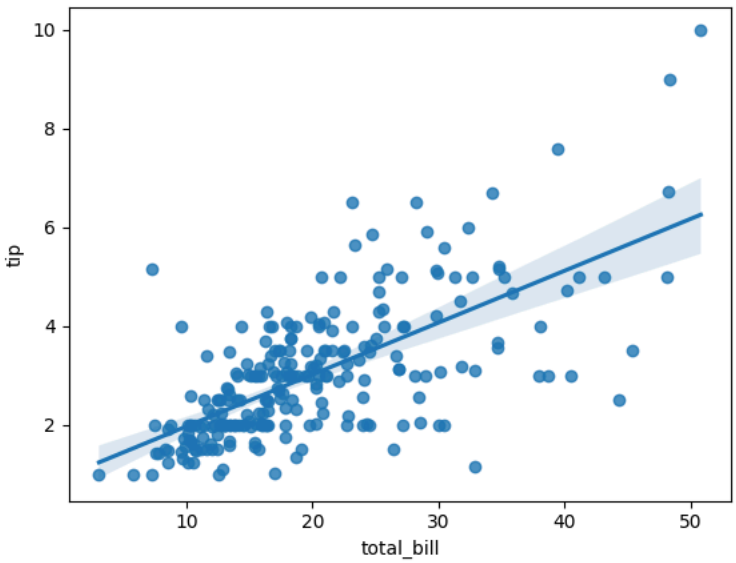
```
import matplotlib.pyplot as plt
import seaborn as sns
df=sns.load_dataset('iris')
#wihtout regression
sns.pairplot(df,kind="scatter")
plt.show()
```




```
import matplotlib.pyplot as plt
import seaborn as sb
df=sb.load_dataset('tips')
sb.regplot(x="total_bill",y="tip",data=df)
plt.show()
```




```
import matplotlib.pyplot as plt
from scipy import stats
```

create an array fro x and y is

```
x=[5,7,8,7,2,17,2,9,4,11,12,9,6]
y=[99,86,87,88,111,86,103,87,94,78,77,85,86]
```

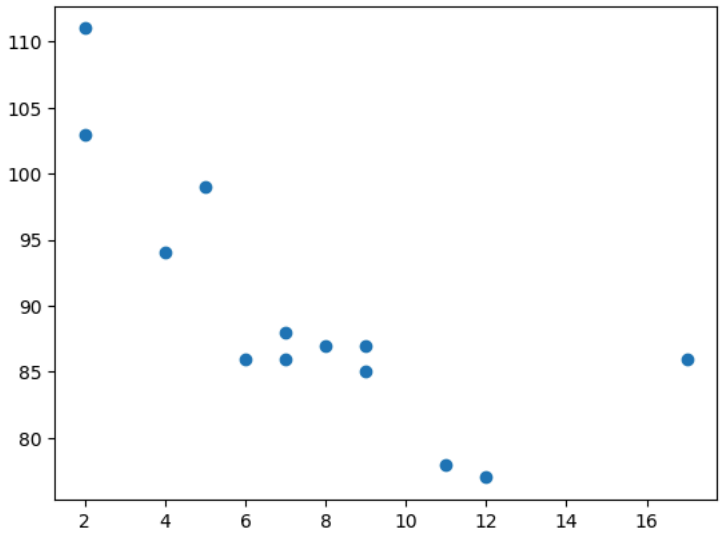
```
slope,intercept,r,p,std_err=stats.linregress(x,y)
#create a fucntion
def myfunc(x):
  return slope * x + intercept
```

```
mymodel=list(map(myfunc,x))
print(mymodel)
```

```
[94.3495217071376, 90.84694628403238, 89.09565857247976, 90.84694628403238, 99.60338484179543, 73.33406916850626, 99.603384841
```

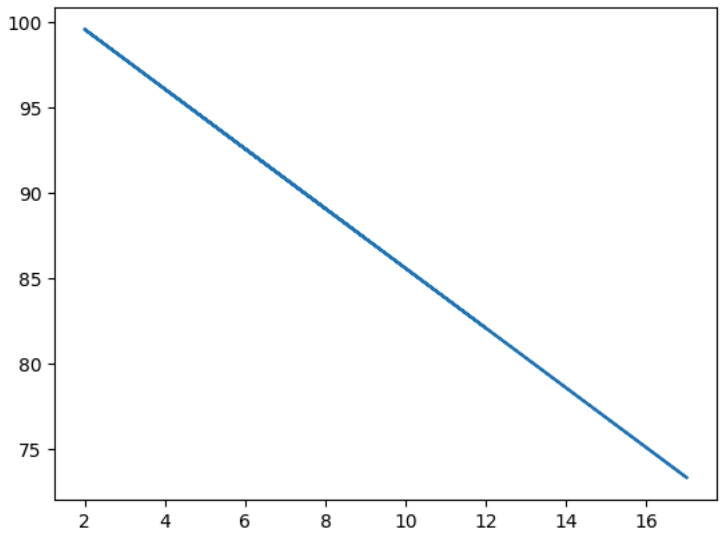
```
#drwa the original scatter point
plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7a625e83d600>
```




```
plt.plot(x,mymodel)
```

```
[<matplotlib.lines.Line2D at 0x7a625ed070d0>]
```




```
import numpy as nmp
import matplotlib.pyplot as plt
def estimate_coeff(p,q):

#here we will estmate the total number of points or observation
  n1=nmp.size(p)
  n2=nmp.size(q)



def estimate_coeff(p,q):

  n1=nmp.size(p)
  #now we will calculate the mean of a and b vctor
m_p=nmp.mean(p)
m_q=nmp.mean(q)
#here we will calculate the cross devaition and  deviation about a
SS_pq=nmp.sum(q*p)-n1*m_q*m_p
SS_pq=nmp.sum(p*p)-n1*m_p*m_p
#here we will calculate the regression coefficents
b_1=SS_pq/SS_pp
b_0=m_q-b_1*m_p
return (b_0,b_1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-29-76c9a448307c> in <cell line: 6>()
      4   #now we will calculate the mean of a and b vctor
      5 m_p=nmp.mean(p)
----> 6 m_q=nmp.mean(q)
      7 #here we will calculate the cross devaition and  deviation about a
      8 SS_pq=nmp.sum(q*p)-n1*m_q*m_p

NameError: name 'q' is not defined
```

```
def plot_regression_line(p,q,b):
  #now we will plot the actual ploints or obeservation as scatter plot
  mtplt.scatter(p,q,color="m",marker="o",s=30)
  #here  we will calculate the prdeicnted response vector
  q_pred=b[0]+b[1]*p#here we will plot the regression line
  mtplot.plot(p,q_pred,color="g")


#here we will put the labels
mtplot.xlabel('p')
mtplot.xlabel('q')
#here we will define the function to show plt
mtplot.show()
def main():
  #entering the observation points or data
  p=np.array([10,11,12,13,14,15,16,17,18,19])
  q=np.array([11,13,12,15,17,18,18,19,20,22])
#now we will estimate the coefficients
b=estimate_coeff(p,q)
print("Estiamted coefficiets are :\nb_0={} \ \nb_1={}".format(b[0],b[1]))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-30-717bc1e4c88a> in <cell line: 2>()
      1 #here we will put the labels
----> 2 mtplot.xlabel('p')
      3 mtplot.xlabel('q')
      4 #here we will define the function to show plt
      5 mtplot.show()

NameError: name 'mtplot' is not defined
```

```
#we will plot the regression line
plot_regression_line(p,q,b)
if __name__ == "__main__":
   main()
```

```
  File "<ipython-input-36-bd37160d2ad9>", line 4
    main()
    ^
IndentationError: expected an indented block after 'if' statement on line 3
```