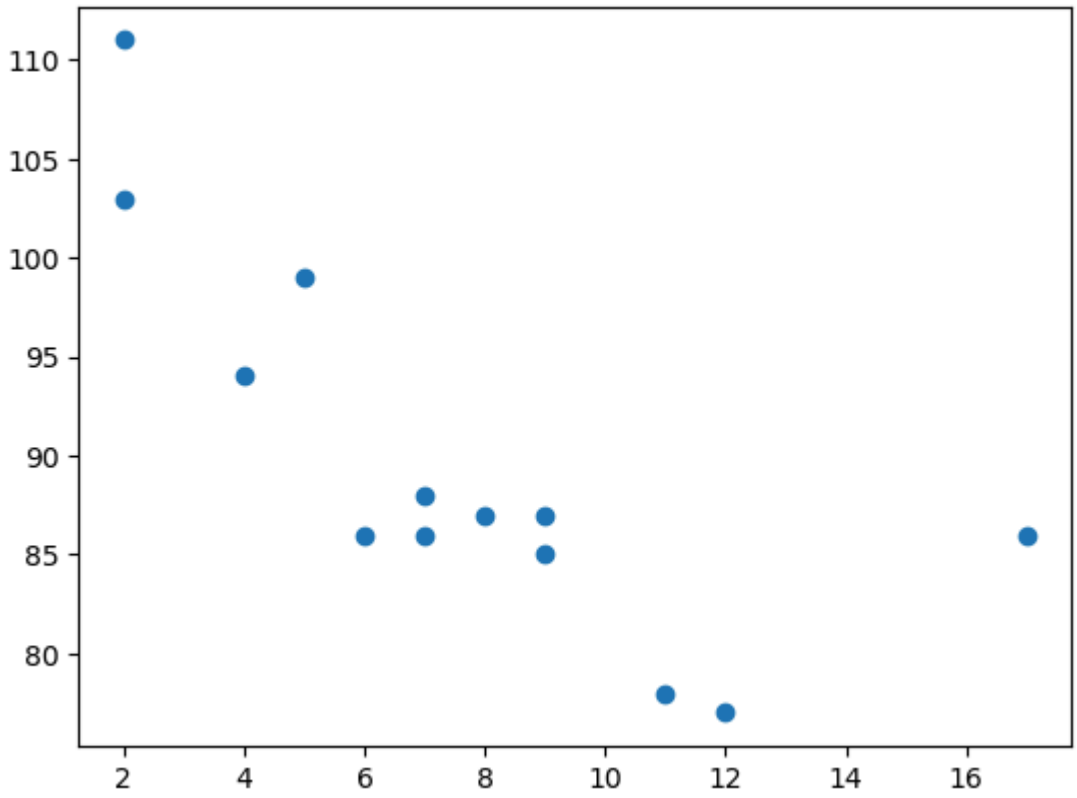
Linear Regression

In [3]:
```
import matplotlib.pyplot as plt
from scipy import stats
```

In [4]:
```
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]
plt.scatter(x,y)
plt.show()
```



Execute a method that returns some important key values of Linear Regression:

In [5]:
```
slope,intercept,r,p,std_err=stats.linregress(x,y)
print("Slope : ",slope)
print("intercept : ",intercept)
print("r : ",r)
print("P-Value : ",p)
print("Standard Error : ",std_err)
```

```
Slope :  -1.7512877115526118
intercept :  103.10596026490066
r :  -0.758591524376155
P-Value :  0.0026468739224561064
Standard Error :  0.453536157607742
```

Create a function that uses the slope and intercept values to return a new value. This new value represents where on the y-axis the corresponding x value will be placed:

In [6]:
```
def myfunction(x):
    return slope * x + intercept
```

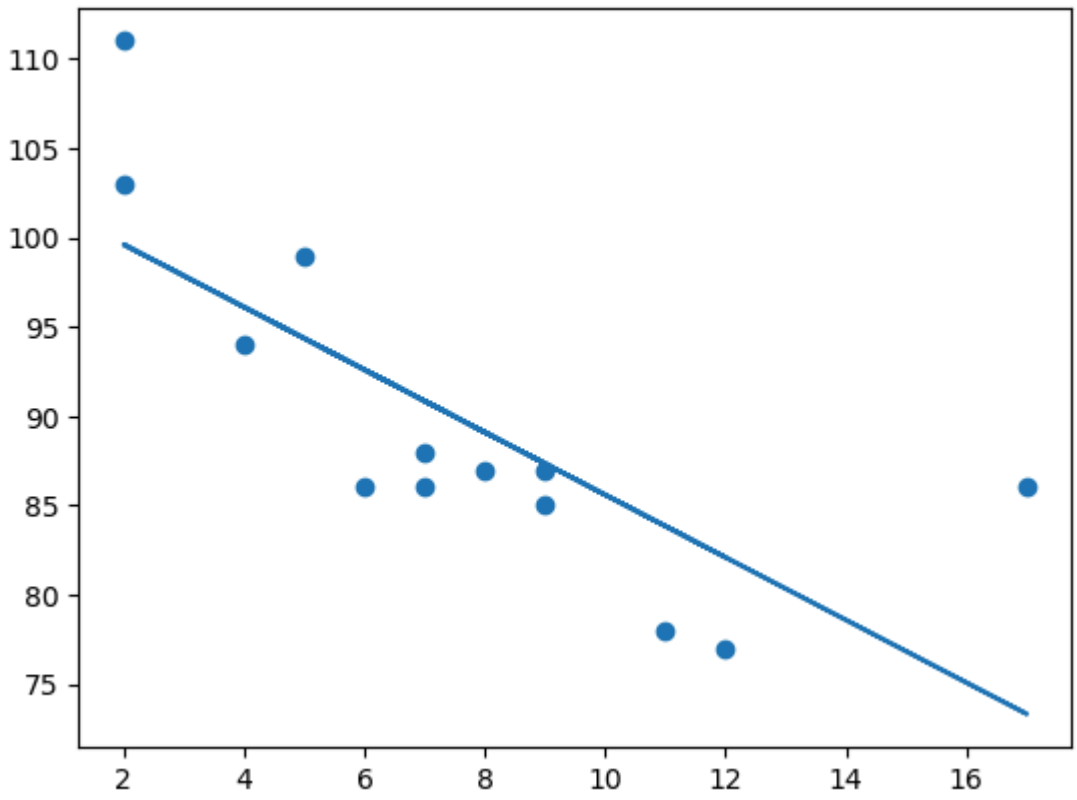
Run each value of the x array through the function. This will result in a new array with new values for the y-axis:

In [7]:
```
mymodel=list(map(myfunction,x))
```

Draw the original scatter plot :

Draw the line of linear regression :

In [12]:
```
plt.scatter(x,y)
plt.plot(x,mymodel)
plt.show()
```



Predict Future Values

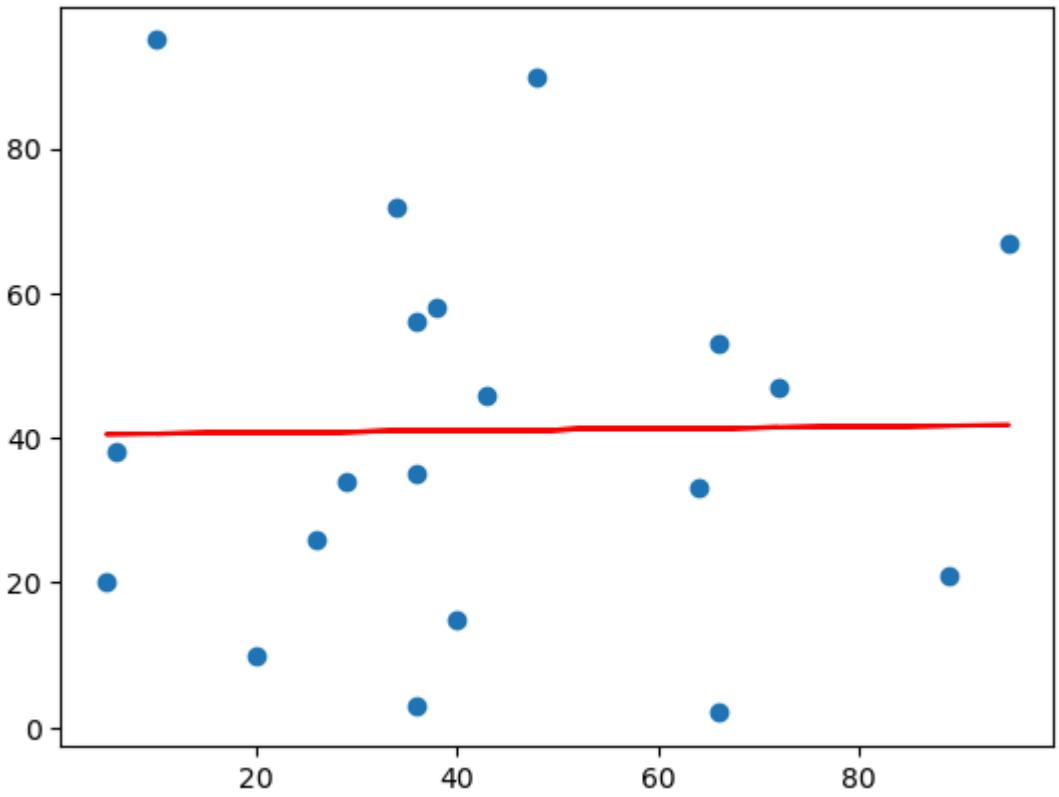
Predict the speed of a 10 years old car:

In [13]:
```
from scipy import stats

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)

def myfunction(x):
  return slope * x + intercept

speed = myfunction(10)

print(speed)
```

```
85.59308314937454
```

he example predicted a speed at 85.6, which we also could read from the diagram:

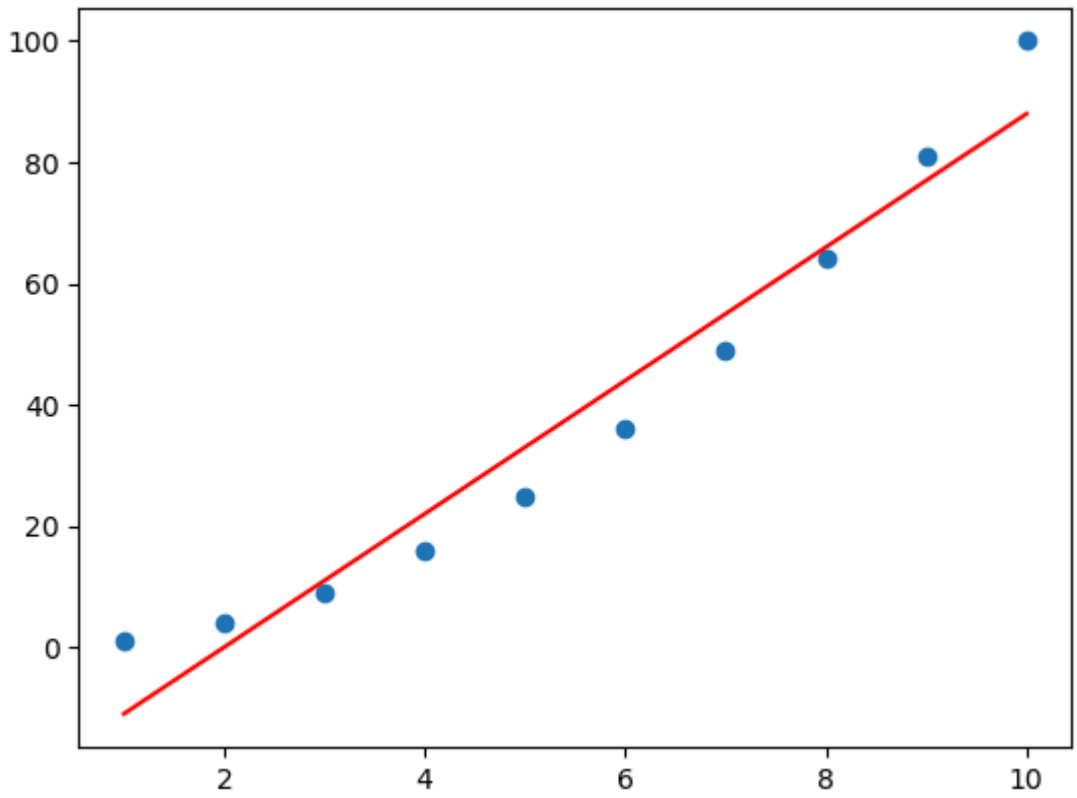
In [19]:
```
import matplotlib.pyplot as plt
from scipy import stats

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]
slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunction(x):
  return slope * x + intercept

my_model=list(map(myfunction,x))

plt.scatter(x,y)
plt.plot(x,my_model,color="r")
plt.show()
```



In [16]:
```
import numpy
from scipy import stats

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

slope, intercept, r, p, std_err = stats.linregress(x, y)

print("Slope : ",slope)
print("intercept : ",intercept)
print("r : ",r)
print("P-Value : ",p)
print("Standard Error : ",std_err)
```

```
Slope :  0.01391658139845263
intercept :  40.452282828936454
r :  0.01331814154297491
P-Value :  0.955558800440106
Standard Error :  0.24627150586388075
```

In [3]:
```
import matplotlib.pyplot as plt
from scipy import stats
x=[1,2,3,4,5,6,7,8,9,10]
y=[1,4,9,16,25,36,49,64,81,100]
slope,intercept,r,p,std_err=stats.linregress(x,y)
def my_function(x):
    return slope * x + intercept

my_model=list(map(my_function,x))

plt.scatter(x,y)
plt.plot(x,my_model,color="r")
plt.show()
```



In [4]:
```
import matplotlib.pyplot as plt
from scipy import stats
x=[1,2,3,4,5,6,7,8,9,10]
y=[1,4,9,16,25,36,49,64,81,100]
slope,intercept,r,p,std_err=stats.linregress(x,y)
print("Slope :",slope)
print("Intercept :",intercept)
print("Correlation Coefficient(r):",r)
print("P-Value :",p)
print("Standard Error :",std_err)
```

```
Slope : 11.0
Intercept : -22.0
Correlation Coefficient(r): 0.9745586289152093
P-Value : 1.7775387117245872e-06
Standard Error : 0.8944271909999157
```

In [6]:
```
import matplotlib.pyplot as plt
from scipy import stats
x=[1,2,3,4,5,6,7,8,9,10]
y=[1,4,9,16,25,36,49,64,81,100]
slope,intercept,r,p,std_err=stats.linregress(x,y)
def my_function(x):
    return slope * x + intercept
Predicted_Value=my_function(11)
print("Predicted Value :",Predicted_Value)
```

```
Predicted Value : 99.0
```

In [ ]:
```

```