What is Linear Regression?

Linear Regression is a statistical supervised learning technique to predict the quantitative variable by forming a linear relationship with one or more independent features.

It helps determine:

→ If a independent variable does a good job in predicting the dependent variable.

→ Which independent variable plays a significant role in predicting the dependent variable.

Simple Linear Regression:

Simple Linear Regression helps to find the linear relationship between two continuous variables,One independent and one dependent feature.

Formula can be represented as y=mx+b

What is a slope?

In a regression context, the slope is very important in the equation because it tells you how much you can expect Y to change as X increases.

It is denoted by m in the formula y = mx+b.

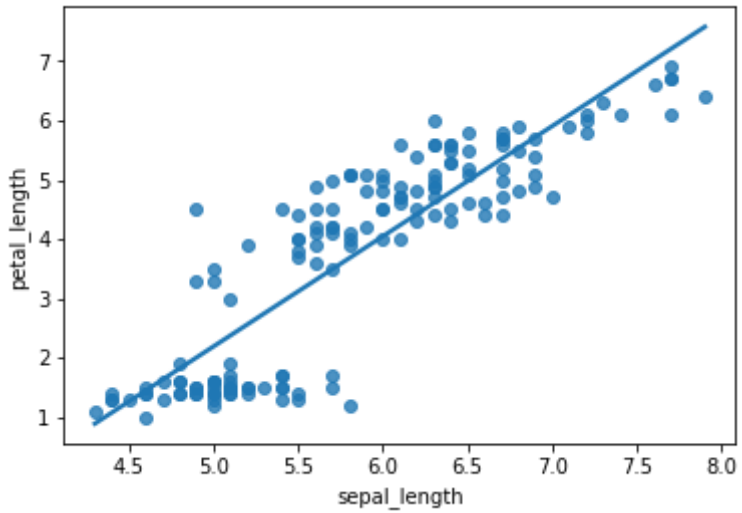
What is Intercept?

The y-intercept is the place where the regression line y = mx + b crosses the y-axis (where x = 0), and is denoted by b.

How it looks

In [3]:
```
import seaborn as sb
import matplotlib.pyplot as plt
df = sb.load_dataset('iris')

# use regplot
sb.regplot(x = "sepal_length",
           y = "petal_length",
           ci = None,
           data = df)
plt.show()
```



Cost Function of Linear Regression

Cost Function is a function that measures the performance of a Machine Learning model for given data.

Cost Function is basically the calculation of the error between predicted values and expected values and presents it in the form of a single real number.

Many people gets confused between Cost Function and Loss Function, Well to put this in simple terms Cost Function is the average of error of n-sample in the data and Loss Function is the error for individual data points.In other words,Loss Function is for one training example,Cost Function is the for the entire training set.

The Cost Function of a linear Regression is taken to be Mean Squared Error. some People may also take Root Mean Square Error.

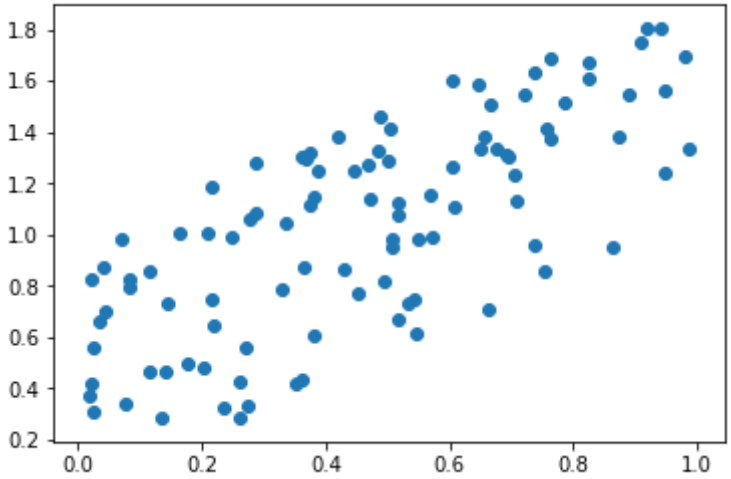
Both are basically same,However adding a Root significantly reduces the value and makes it easy to read.We,take Square here so that we don't get values in negative.

Cost Function (J) = 1/n*Σ(y_pred - y)^2

Cost Function (J) = 1/n*Σ((wx + b) - y)^2

Gradient Descent

In [4]:
```
import numpy as np
from numpy import random
import seaborn as sb
import matplotlib.pyplot as plt
X = random.rand(100)
res = 0.01*random.rand(100)
Y = random.rand(100) + res + X
plt.scatter(X, Y)
plt.show()
```



In [5]:
```
# Building the model
m = 0 #Initial Value of slope :  Hyper Parameter
c = 0 #Initial Value of Intercept : Hyper Parameter

L = 0.01  # The learning Rate : Hyper Parameter
epochs = 10000  # The number of iterations to perform gradient descent : Hyper Parameter

n = float(len(X)) # Number of elements in X

# Performing Gradient Descent
for i in range(epochs):
    Y_pred = m*X + c  # The current predicted value of Y
    D_m = (-2/n) * sum(X * (Y - Y_pred))  # Derivative wrt m
    D_c = (-2/n) * sum(Y - Y_pred)  # Derivative wrt c
    m = m - L * D_m  # Update m
    c = c - L * D_c  # Update c

print (m, c)

#m Final value of slope :  Parameters
#c Final value of intercept: Parameters
```

```
1.0943140899467156 0.527769481025532
```

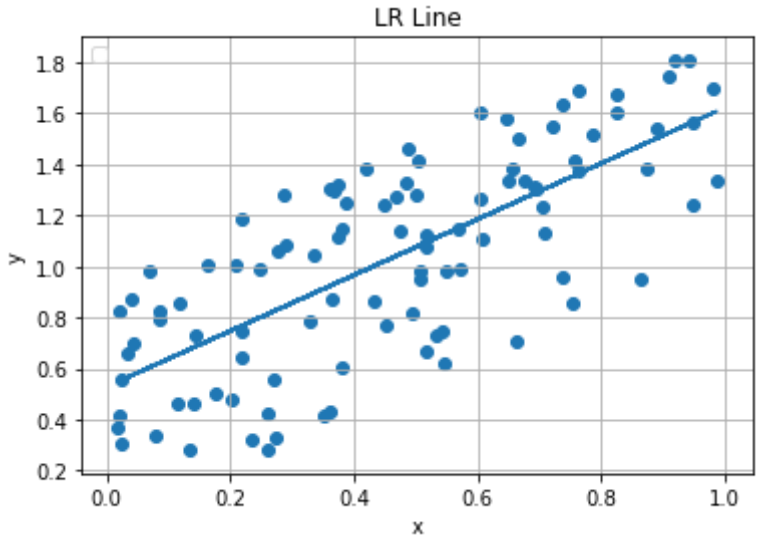
Gradient descent Steps:

We need to find the values of M and C for which the value of our cost function is minimum

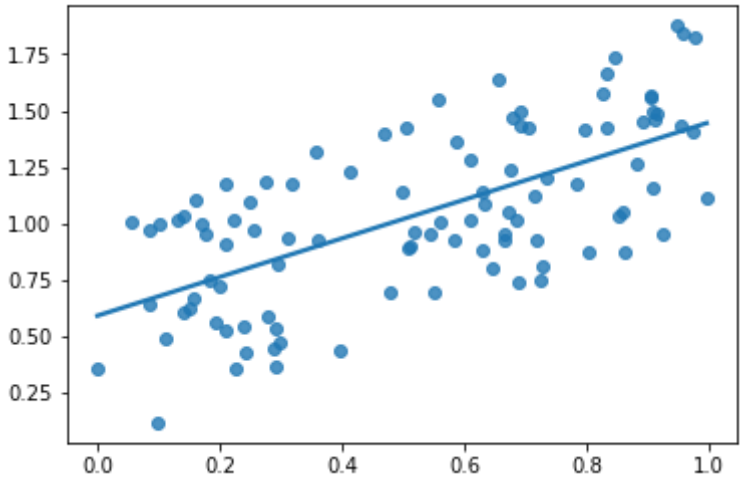
1. We define Initial values of M and C(Maybe random)
2. We derivate our slope and our intercept
3. We will take a step in that direction(the len of the step will be given by learning rate)
4. We keep doing this for the number of epochs defined

In [6]:
```
Line = m*X + c
plt.plot(X, Line)
plt.scatter(X,Y)
plt.title('LR Line')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc='upper left')
plt.grid()
plt.show()
```

```
No handles with labels found to put in legend.
```



In [17]:
```
sb.regplot(x =X,
           y =Y,
           ci = None,
           )
plt.show()
```



W : Weights(Slope) B : Bias(Intercept)

y = wx + b

y = w1x1 + w2x2 + w3x3 + b

Asumptions for Linear Regression:

1. Linear Relationship between x and y (X -> independent Var , Y -> Dependent Var)
2. No autocorrelation of errors

3. Multivariate Normality
4. No or very little multi-colinearity
5. Homoscedasticity

Colinearity

y = m1x1 + m2x2 + m3x3 + c

x1 is correlated with x2

y = m1x1 + m2(f(x1)) + m3x3

In [ ]:
```
1
```