Simple way to understand linear regression

In [48]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

I will be taking simple numbers to make you understand the concept of linear regression but usually real world dataset will be different eg (insurance cost and age)

In [74]:
```
X=np.array([1, 2, 3, 4, 5, 6])
```

here i deliberately took 30 after 40 to make you understand how linear reression works

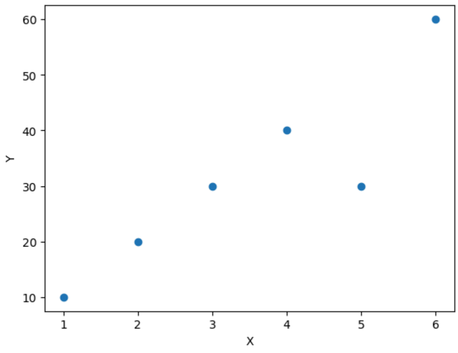
In [82]:
```
y=np.array([10, 20, 30, 40, 30 ,60])
```

why are we reshaping because scikit learn takes 2d array we have 1d array

In [84]:
```
X = X.reshape(-1, 1)
```

In [134…
```
plt.scatter(X,y)
plt.xlabel('X')
plt.ylabel('Y')
```

Out[134…
```
Text(0, 0.5, 'Y')
```



train_test_split is used to split the data as we discussed earlier

In [88]:
```
from sklearn.model_selection import train_test_split
```

The belolw code is used to split the data, what is test size ? the data used to test from training the model,and we also define random state so that it will not shuffle the data and our output will be same each time we execute

In [90]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [92]:
```
from sklearn.linear_model import LinearRegression # now we import linear regression model
```

In [94]:
```
lr = LinearRegression()
```

In [102…
```
lr.fit(X_train,y_train)# now we fit the data into linear regression
```

Out[102…
```
LinearRegression
LinearRegression()
```

In [104…
```
X_test #we have x test data
```

Out[104…
```
array([[5],
       [2]])
```

In [106…
```
y_test  #we have y test data
```

Out[106…
```
array([30, 20])
```

WE HAVE BOTH X TEST AND Y TEST BUT THE LINEAR REGRESSION MODEL DOESNT HAVE IDEA OF WHAT Y TEST VALUES ARE

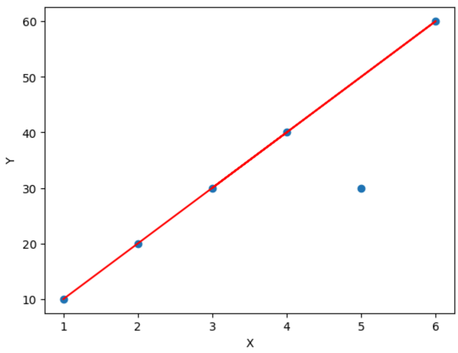
LETS SEE WHAT THE MODEL PREDICTS , AS WE KNOW HAD TAKEN VALUE OF 5 AS 30 WHICH IS ALSO KNOWN AS NOISE MEANING IRREGULAR

In [130…
```
lr.predict(X_test[0].reshape(1, -1))
```

Out[130…
```
array([50.])
```

In [150…
```
plt.scatter(X,y)
plt.plot(X_train,lr.predict(X_train.reshape(-1, 1)), color='red')
plt.xlabel('X')
plt.ylabel('Y')
```

Out[150…
```
Text(0, 0.5, 'Y')
```



this the best fit line

In [154…
```
m=lr.coef_
```

In [156…
```
b=lr.intercept_
```

y=mx+b slope of line.

why use slope of line?

eg if we want to predictthe value of number 7 which is not in the data set

we can use the coef and intercept to predict it

In [188…
```
m * 7 +b #y=mx+b
```

Out[188…
```
array([70.])
```