```
# Importing Necessary libraries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving headbrain.csv to headbrain (1).csv

```
import io
df = pd.read_csv(io.BytesIO(uploaded['headbrain.csv']))
df
```

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
0	1	1	4512	1530
1	1	1	3738	1297
2	1	1	4261	1335
3	1	1	3777	1282
4	1	1	4177	1590
...	...	...	...	...
232	2	2	3214	1110
233	2	2	3394	1215
234	2	2	3233	1104
235	2	2	3352	1170
236	2	2	3391	1120

237 rows × 4 columns

```
df.head()
```

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
0	1	1	4512	1530
1	1	1	3738	1297
2	1	1	4261	1335
3	1	1	3777	1282
4	1	1	4177	1590

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237 entries, 0 to 236
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Gender               237 non-null    int64
 1   Age Range            237 non-null    int64
 2   Head Size(cm^3)      237 non-null    int64
 3   Brain Weight(grams)  237 non-null    int64
dtypes: int64(4)
memory usage: 7.5 KB
```

```
df.isnull().sum()
```

```
Gender                 0
Age Range              0
Head Size(cm^3)        0
Brain Weight(grams)    0
dtype: int64
```

```
df.shape
```

```
(237, 4)
```

```
# Taking x and y variables
X = df['Head Size(cm^3)'].values
Y =  df['Brain Weight(grams)'].values
```

```
X
```

```
array([4512, 3738, 4261, 3777, 4177, 3585, 3785, 3559, 3613, 3982, 3443,
       3993, 3640, 4208, 3832, 3876, 3497, 3466, 3095, 4424, 3878, 4046,
       3804, 3710, 4747, 4423, 4036, 4022, 3454, 4175, 3787, 3796, 4103,
       4161, 4158, 3814, 3527, 3748, 3334, 3492, 3962, 3505, 4315, 3804,
       3863, 4034, 4308, 3165, 3641, 3644, 3891, 3793, 4270, 4063, 4012,
       3458, 3890, 4166, 3935, 3669, 3866, 3393, 4442, 4253, 3727, 3329,
       3415, 3372, 4430, 4381, 4008, 3858, 4121, 4057, 3824, 3394, 3558,
       3362, 3930, 3835, 3830, 3856, 3249, 3577, 3933, 3850, 3309, 3406,
       3506, 3907, 4160, 3318, 3662, 3899, 3700, 3779, 3473, 3490, 3654,
       3478, 3495, 3834, 3876, 3661, 3618, 3648, 4032, 3399, 3916, 4430,
       3695, 3524, 3571, 3594, 3383, 3499, 3589, 3900, 4114, 3937, 3399,
       4200, 4488, 3614, 4051, 3782, 3391, 3124, 4053, 3582, 3666, 3532,
       4046, 3667, 2857, 3436, 3791, 3302, 3104, 3171, 3572, 3530, 3175,
       3438, 3903, 3899, 3401, 3267, 3451, 3090, 3413, 3323, 3680, 3439,
       3853, 3156, 3279, 3707, 4006, 3269, 3071, 3779, 3548, 3292, 3497,
```

```
       3082, 3248, 3358, 3803, 3566, 3145, 3503, 3571, 3724, 3615, 3203,
       3609, 3561, 3979, 3533, 3689, 3158, 4005, 3181, 3479, 3642, 3632,
       3069, 3394, 3703, 3165, 3354, 3000, 3687, 3556, 2773, 3058, 3344,
       3493, 3297, 3360, 3228, 3277, 3851, 3067, 3692, 3402, 3995, 3318,
       2720, 2937, 3580, 2939, 2989, 3586, 3156, 3246, 3170, 3268, 3389,
       3381, 2864, 3740, 3479, 3647, 3716, 3284, 4204, 3735, 3218, 3685,
       3704, 3214, 3394, 3233, 3352, 3391])
```

```
X.shape
```

```
(237,)
```

```
Y
```

```
array([1530, 1297, 1335, 1282, 1590, 1300, 1400, 1255, 1355, 1375, 1340,
       1380, 1355, 1522, 1208, 1405, 1358, 1292, 1340, 1400, 1357, 1287,
       1275, 1270, 1635, 1505, 1490, 1485, 1310, 1420, 1318, 1432, 1364,
       1405, 1432, 1207, 1375, 1350, 1236, 1250, 1350, 1320, 1525, 1570,
       1340, 1422, 1506, 1215, 1311, 1300, 1224, 1350, 1335, 1390, 1400,
       1225, 1310, 1560, 1330, 1222, 1415, 1175, 1330, 1485, 1470, 1135,
       1310, 1154, 1510, 1415, 1468, 1390, 1380, 1432, 1240, 1195, 1225,
       1188, 1252, 1315, 1245, 1430, 1279, 1245, 1309, 1412, 1120, 1220,
       1280, 1440, 1370, 1192, 1230, 1346, 1290, 1165, 1240, 1132, 1242,
       1270, 1218, 1430, 1588, 1320, 1290, 1260, 1425, 1226, 1360, 1620,
       1310, 1250, 1295, 1290, 1290, 1275, 1250, 1270, 1362, 1300, 1173,
       1256, 1440, 1180, 1306, 1350, 1125, 1165, 1312, 1300, 1270, 1335,
       1450, 1310, 1027, 1235, 1260, 1165, 1080, 1127, 1270, 1252, 1200,
       1290, 1334, 1380, 1140, 1243, 1340, 1168, 1322, 1249, 1321, 1192,
       1373, 1170, 1265, 1235, 1302, 1241, 1078, 1520, 1460, 1075, 1280,
       1180, 1250, 1190, 1374, 1306, 1202, 1240, 1316, 1280, 1350, 1180,
       1210, 1127, 1324, 1210, 1290, 1100, 1280, 1175, 1160, 1205, 1163,
       1022, 1243, 1350, 1237, 1204, 1090, 1355, 1250, 1076, 1120, 1220,
       1240, 1220, 1095, 1235, 1105, 1405, 1150, 1305, 1220, 1296, 1175,
        955, 1070, 1320, 1060, 1130, 1250, 1225, 1180, 1178, 1142, 1130,
       1185, 1012, 1280, 1103, 1408, 1300, 1246, 1380, 1350, 1060, 1350,
       1220, 1110, 1215, 1104, 1170, 1120])
```

```
Y.shape
```

```
(237,)
```

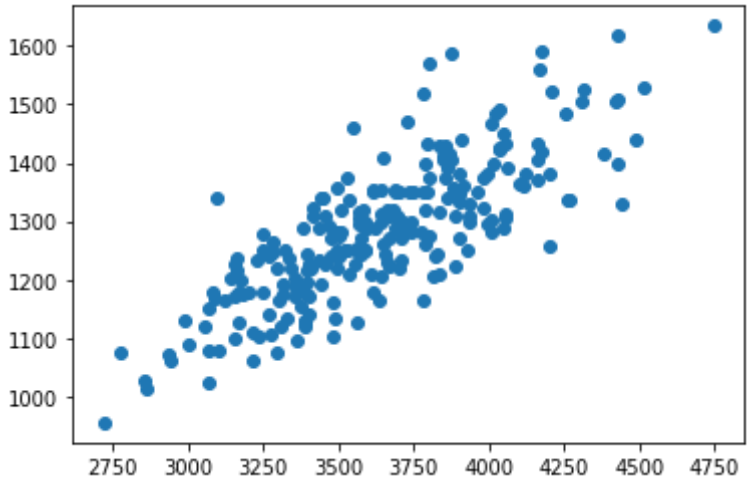
```
mean_X = np.mean(X)
mean_Y = np.mean(Y)

n = len(X)

num =0
denom = 0

for i in range(n):
    num += (X[i]-mean_X)* (Y[i]-mean_Y)
    denom +=(X[i]-mean_X)**2
m = num/denom
c = mean_Y - (m*mean_X)
```

```
print(m,',',c)
```

```
0.26342933948939945 , 325.57342104944223
```

```
plt.scatter(X,Y)
```

```
<matplotlib.collections.PathCollection at 0x7efec5ab1ed0>
```



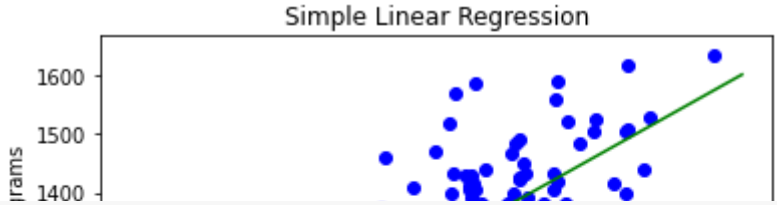

```
min_x = np.min(X)-100
max_x = np.max(X)+100
```

```
x = np.linspace(min_x,max_x,1000)
```

```
y = m*x+c
```

```
plt.scatter(X,Y,color='B')
plt.plot(x,y,color='G')
plt.title('Simple Linear Regression')
plt.xlabel('Head size cm^3')
plt.ylabel('Brain weight in grams')
```

```
Text(0, 0.5, 'Brain weight in grams')
```




```
#Calculating the error
sum_pred = 0
sum_act = 0

for i in range(n):
    y_pred = (m*X[i]+c)
    sum_pred += (Y[i]-y_pred)**2
    sum_act +=(Y[i]-mean_Y)**2

r2 = 1-(sum_pred/sum_act)
print(r2)
```

```
0.6393117199570003
```

```
#Here we can observe that we got R**2> 0.5 . so we have good model
```

```
def predict(x):
    y = m*x + c
    print(y)
```

```
predict(4277)
```

```
1452.2607060456037
```

```
predict(4512)
```

```
1514.1666008256125
```

```
# USING SKLEARN MODEL
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X  = X.reshape((n,1))
```

```
X.shape
```

```
(237, 1)
```

```
y.shape
```

```
y.shape
```

```
(1000,)
```

```
lg = LinearRegression()
```

```
lg.fit(X,Y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_pred = lg.predict(X)
```

```
mse = mean_squared_error(Y,y_pred)
```

```
rmse = np.sqrt(mse)
```

```
r2_score = lg.score(X,Y)
```

```
print(rmse)
print(r2_score)
```

```
72.1206213783709
0.639311719957
```

```
lg.predict([[4177]])
```

```
array([1425.9177721])
```

```
lg.intercept_
```

```
325.5734210494426
```