```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
import kagglehub
jemishdonda_headbrain_path = kagglehub.dataset_download('jemishdonda/headbrain')

print('Data source import complete.')
```

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

```
df = pd.read_csv('../input/headbrain.csv')
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):
Gender                 237 non-null int64
Age Range              237 non-null int64
Head Size(cm^3)        237 non-null int64
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
```

- Our dataset has no categorical values we can move forward.
- we don't have any null values in our dataset.

```
df.shape
```

```
(237, 4)
```

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

```
X.shape
```

```
(237,)
```

```
Y.shape
```

```
(237,)
```

Method 1: munual coding

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

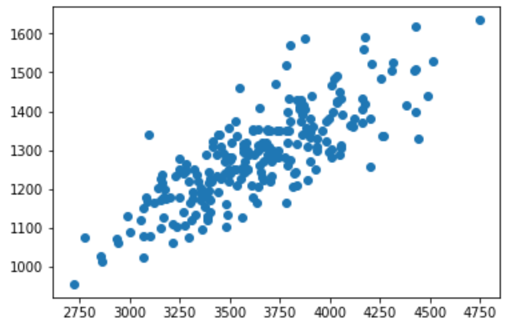
```
n = len(X)

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
```

Here , we calculate m and b. Now we need to find the line

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

```
<matplotlib.collections.PathCollection at 0x7f1e01f189b0>
```



creating dummy test set

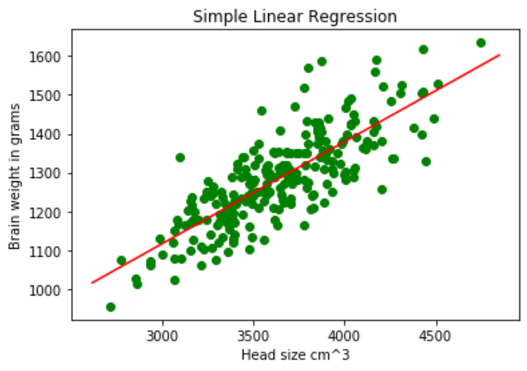
```
min_x = np.min(X)-100
max_x = np.max(X)+100
```

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

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

```
plt.scatter(X,Y,color='g')
plt.plot(x,y,color='r')
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(4177)
```

```
1425.9177720966638
```

here we predict the brain wieght for given head size(cm^3)

Method 2: using scikit learn

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

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

```
X.shape
```

```
(237, 1)
```

```
y.shape
```

```
(1000,)
```

```
lg = LinearRegression()
```

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

```
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
```

we got the same error R**2 value as above method-1

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

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

```
lg.intercept_
```

```
325.5734210494428
```