In [1]:
```
import numpy as np
```

In [4]:
```
X=np.array([[1],[2],[3],[4],[5]]) #this is used to create the array(column)
Y=np.array([[1],[3],[2],[3],[5]])
```

In [5]:
```
X
```

Out[5]:
```
array([[1],
       [2],
       [3],
       [4],
       [5]])
```

In [6]:
```
mx=X.mean() #calculates mean here (1+2+3+4+5)/5
```

In [7]:
```
my=Y.mean()  #calculates mean of Y values
```

In [8]:
```
mx   #mean of all values in X array prints
```

Out[8]:
```
3.0
```

In [9]:
```
my   #prints mean of all values in Y array
```

Out[9]:
```
2.8
```

In [11]:
```
Z=np.array([1,2,3,4,5]) #this is used to create the array(row)
Z
```

Out[11]:
```
array([1, 2, 3, 4, 5])
```

In [12]:
```
F=np.array([[1,2,3],[4,5,6],[7,8,9]])
F
```

Out[12]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [13]:
```
vx=mx-X    #vx=mean-ActualValueIn(X) OR  bar(x)-X
vy=my-Y    #vy=mean-ActualValueIn(Y)
```

In [14]:
```
xy=vx*vy    #(meanOf X-X)*(meanOf Y-Y)
```

In [15]:
```
sum(xy)      #adds all the values('-' inclusive)
```

Out[15]:
```
array([8.])
```

In [16]:
```
m=sum(xy)/sum(np.square(vx)) #division by square of values of X
m
```

Out[16]:
```
array([0.8])
```

In [17]:
```
#LINEAR REGRESSION
from sklearn.linear_model import LinearRegression
```

In [18]:
```
obj=LinearRegression(fit_intercept=True) #if intercept is false no intercept used in calculations
```

In [19]:
```
model=obj.fit(X,Y)  #used to fit the linear model
```

In [20]:
```
model
```

Out[20]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [22]:
```
model.predict([[1]]) #predict using linear model
```

Out[22]:
```
array([[1.2]])
```

In [25]:
```
model.predict([[4]])
```

Out[25]:
```
array([[3.6]])
```

In [27]:
```
model.predict([[5]])
```

Out[27]:
```
array([[4.4]])
```

In [26]:
```
model.coef_
```

Out[26]:
```
array([[0.8]])
```

In [28]:
```
model.intercept_
```

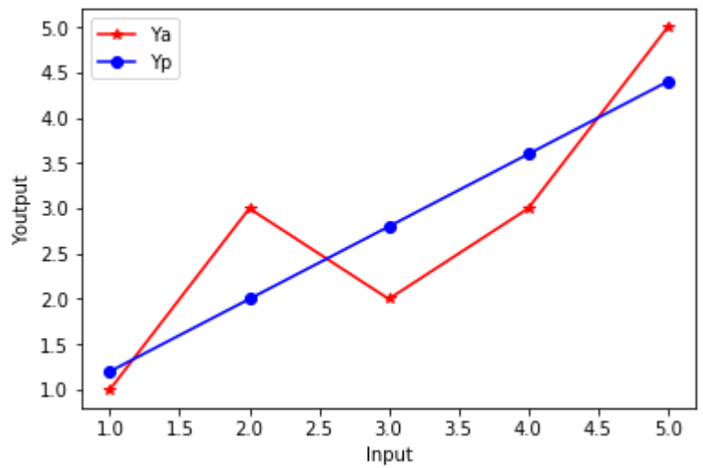
Out[28]:
```
array([0.4])
```

In [30]:
```
Ya=Y #X
Yp=model.predict(X)
```

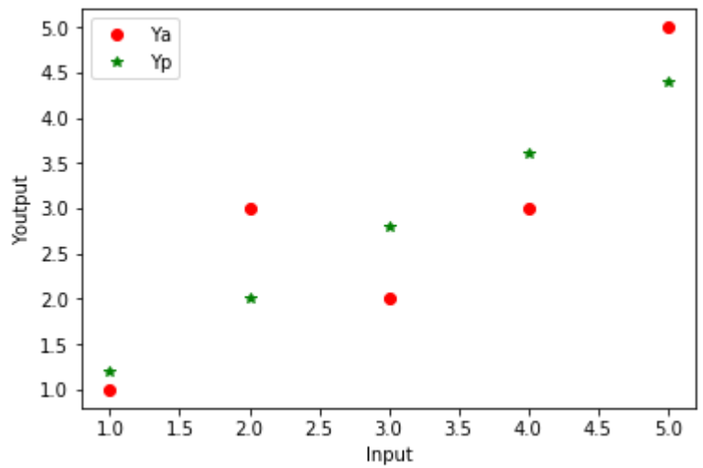
In [31]:
```
#matplotlib is a plotting library in python programming language
#and it's numerical extension is numpy

from matplotlib import pyplot as plt
```

In [32]:
```
plt.plot(X,Ya,color="red",marker="*")
plt.plot(X,Yp,color="blue",marker="o")
plt.xlabel("Input")
plt.ylabel("Youtput")
plt.legend(["Ya","Yp"])   #here whatever is written before is set to plotted in red with marker'*'
plt.show()
```



In [36]:
```
plt.plot(X,Ya,'ro')  #this consist of ro 'r' indicates red color and 'o' marker
plt.plot(X,Yp,'g*') #this means green color and '*' marker
plt.xlabel("Input")
plt.ylabel("Youtput")
plt.legend(["Ya","Yp"])
plt.show()
```



In [37]:
```
#Finding mean squared error
Ya      #Actual values
```

Out[37]:
```
array([[1],
       [3],
       [2],
       [3],
       [5]])
```

In [38]:
```
Yp     #predicted values
```

Out[38]:
```
array([[1.2],
       [2. ],
       [2.8],
       [3.6],
       [4.4]])
```

In [40]:
```
#s1=np.square(Ya)
#s2=np.square(Yp)
s=Ya-Yp       #ActualValue-PredictedValue
```

In [41]:
```
#s=s1-s2
s
```

Out[41]:
```
array([[-0.2],
       [ 1. ],
       [-0.8],
       [-0.6],
       [ 0.6]])
```

In [45]:
```
np.sqrt(sum(np.square(s))) # contains error
```

Out[45]:
```
array([1.54919334])
```

In [46]:
```
#it is used to find the mean squared error
from sklearn.metrics import mean_squared_error
```

In [47]:
```
#yp=.8x+.4+.69
me=mean_squared_error(Ya,Yp)
np.sqrt(me)
```

Out[47]:
```
0.692820323027551
```

In [50]:
```
#recorrected above calculations
np.sqrt(sum(np.square(s))/len(s))     #(1/n)sigma square(Ya-Yp)
```

Out[50]:
```
array([0.69282032])
```

In [51]:
```
#pandas is used for data Analysis
import pandas as pd
```

In [52]:
```
#this is used to read the file
mydata=pd.read_csv(r"E:\BOOKS pdf\EXTRA STUFFS\Internships\DataSets-master\data_brain_own_model_design.csv")
```

In [53]:
```
mydata
```

Out[53]:

	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
5	1	1	3585	1300
6	1	1	3785	1400
7	1	1	3559	1255
8	1	1	3613	1355
9	1	1	3982	1375
10	1	1	3443	1340
11	1	1	3993	1380
12	1	1	3640	1355
13	1	1	4208	1522
14	1	1	3832	1208
15	1	1	3876	1405
16	1	1	3497	1358
17	1	1	3466	1292
18	1	1	3095	1340
19	1	1	4424	1400
20	1	1	3878	1357
21	1	1	4046	1287
22	1	1	3804	1275
23	1	1	3710	1270
24	1	1	4747	1635

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
25	1	1	4423	1505
26	1	1	4036	1490
27	1	1	4022	1485
28	1	1	3454	1310
29	1	1	4175	1420
...	...	...	...	...
207	2	2	3995	1296
208	2	2	3318	1175
209	2	2	2720	955
210	2	2	2937	1070
211	2	2	3580	1320
212	2	2	2939	1060
213	2	2	2989	1130
214	2	2	3586	1250
215	2	2	3156	1225
216	2	2	3246	1180
217	2	2	3170	1178
218	2	2	3268	1142
219	2	2	3389	1130
220	2	2	3381	1185
221	2	2	2864	1012
222	2	2	3740	1280
223	2	2	3479	1103
224	2	2	3647	1408
225	2	2	3716	1300
226	2	2	3284	1246
227	2	2	4204	1380

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
228	2	2	3735	1350
229	2	2	3218	1060
230	2	2	3685	1350
231	2	2	3704	1220
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

In [54]:
```
mydata[:2]  #upto 1 index print
```

Out[54]:

	Gender	Age Range	Head Size(cm^3)	Brain Weight(grams)
0	1	1	4512	1530
1	1	1	3738	1297

In [55]:
```
X_input=mydata.iloc[:,0:3]     #contains Gender,AgeRange,HeadSize
Y_out=mydata.iloc[:,3]         #only contains BrainWeight
```

In [56]:
```
type(X_input)
```

Out[56]: pandas.core.frame.DataFrame

In [57]:
```
type(Y_out)
```

Out[57]: pandas.core.series.Series

In [58]:
```
X_input
```

Out[58]:

	Gender	Age Range	Head Size(cm^3)
0	1	1	4512
1	1	1	3738
2	1	1	4261
3	1	1	3777
4	1	1	4177
5	1	1	3585
6	1	1	3785
7	1	1	3559
8	1	1	3613
9	1	1	3982
10	1	1	3443
11	1	1	3993
12	1	1	3640
13	1	1	4208
14	1	1	3832
15	1	1	3876
16	1	1	3497
17	1	1	3466
18	1	1	3095
19	1	1	4424
20	1	1	3878
21	1	1	4046
22	1	1	3804
23	1	1	3710
24	1	1	4747

	Gender	Age Range	Head Size(cm^3)
25	1	1	4423
26	1	1	4036
27	1	1	4022
28	1	1	3454
29	1	1	4175
...	...	...	...
207	2	2	3995
208	2	2	3318
209	2	2	2720
210	2	2	2937
211	2	2	3580
212	2	2	2939
213	2	2	2989
214	2	2	3586
215	2	2	3156
216	2	2	3246
217	2	2	3170
218	2	2	3268
219	2	2	3389
220	2	2	3381
221	2	2	2864
222	2	2	3740
223	2	2	3479
224	2	2	3647
225	2	2	3716
226	2	2	3284
227	2	2	4204

	Gender	Age Range	Head Size(cm^3)
228	2	2	3735
229	2	2	3218
230	2	2	3685
231	2	2	3704
232	2	2	3214
233	2	2	3394
234	2	2	3233
235	2	2	3352
236	2	2	3391

237 rows × 3 columns

In [59]:
```
Y_out
```

Out[59]:
```
0      1530
1      1297
2      1335
3      1282
4      1590
5      1300
6      1400
7      1255
8      1355
9      1375
10     1340
11     1380
12     1355
13     1522
14     1208
15     1405
16     1358
17     1292
18     1340
19     1400
20     1357
21     1287
22     1275
23     1270
24     1635
25     1505
26     1490
27     1485
28     1310
29     1420
       ...
207    1296
208    1175
209     955
210    1070
211    1320
212    1060
213    1130
214    1250
215    1225
```

```
216    1180
217    1178
218    1142
219    1130
220    1185
221    1012
222    1280
223    1103
224    1408
225    1300
226    1246
227    1380
228    1350
229    1060
230    1350
231    1220
232    1110
233    1215
234    1104
235    1170
236    1120
Name: Brain Weight(grams), Length: 237, dtype: int64
```

In [60]:
```
mydata.shape   #In columns except index number
```

Out[60]:
```
(237, 4)
```

In [61]:
```
XA=X_input.values
YA=Y_out.values
```

In [62]:
```
print(type(XA))
print(type(YA))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

In [63]:
```
XA   #2D array
```

Out[63]:
```
array([[   1,    1, 4512],
       [   1,    1, 3738],
       [   1,    1, 4261],
       [   1,    1, 3777],
       [   1,    1, 4177],
       [   1,    1, 3585],
       [   1,    1, 3785],
       [   1,    1, 3559],
       [   1,    1, 3613],
       [   1,    1, 3982],
       [   1,    1, 3443],
       [   1,    1, 3993],
       [   1,    1, 3640],
       [   1,    1, 4208],
       [   1,    1, 3832],
       [   1,    1, 3876],
       [   1,    1, 3497],
       [   1,    1, 3466],
       [   1,    1, 3095],
```

In [64]:
```
YA   #1D array
```

Out[64]:
```
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], dtype=int64)
```

In [65]:
```
#split
X_train=XA[:168]   #first training and than testing
X_test=XA[168:]
Y_train=YA[:168]
Y_test=YA[168:]
```

In [66]:
```
#training using train data
from sklearn.linear_model import LinearRegression
```

In [67]:
```
model=obj.fit(X_train,Y_train)
```

In [73]:
```
#Testing for prediction
Yaa=Y_test
Ypp=model.predict(X_test)
```

In [74]:
```
model.coef_
```

Out[74]:
```
array([-18.60685691, -30.44156273,   0.23767608])
```

In [76]:
```
Ypp
```

Out[76]:
```
array([1331.87019264, 1275.54096211, 1175.47933319, 1260.56736918,
       1276.7293425 , 1313.09378246, 1287.18708994, 1189.26454573,
       1285.76103347, 1274.35258172, 1373.7011824 , 1267.69765153,
       1304.77511973, 1178.56912221, 1379.88076044, 1184.03567201,
       1254.86314331, 1293.60434405, 1291.22758327, 1126.97438852,
       1204.21911393, 1277.66102209, 1149.79129203, 1194.71207081,
       1110.57473913, 1273.85820484, 1242.7226386 , 1056.62226938,
       1124.35995166, 1192.33531002, 1227.74904567, 1181.16453435,
       1196.13812727, 1164.76488495, 1176.41101279, 1312.83708167,
       1126.49903637, 1275.04658523, 1206.12052256, 1347.06243692,
       1186.15573199, 1044.02543723, 1095.6011462 , 1248.42686448,
       1096.07649836, 1107.96030227, 1249.85292095, 1147.65220732,
       1169.04305436, 1150.97967242, 1174.27192808, 1203.03073354,
       1201.12932492, 1078.25079249, 1286.45503699, 1224.42158058,
       1264.35116171, 1280.75081111, 1178.07474533, 1396.73673727,
       1285.2666566 , 1162.38812417, 1273.38285269, 1277.89869817,
       1161.43741986, 1204.21911393, 1165.95326534, 1194.23671865,
       1203.5060857 ])
```

In [78]:
```
#accuracy biased calculation
model.intercept_
```

Out[78]:
```
495.64334383936534
```

In [79]:
```
from sklearn.metrics import mean_squared_error
```

In [80]:
```
error=mean_squared_error(Yaa,Ypp)
```

In [81]:
```
error
```

Out[81]:
```
3971.5387326225064
```

In [82]:
```
np.sqrt(error)
```

Out[82]:
```
63.0201454506613
```

In [2]:
```
from matplotlib import pyplot as plt
```

In [3]:
```
#plot graph between ActualHeight vs weight and predicted height vs weight
plt.plot(X,Yaa,'ro')
plt.plot(X,Ypp,'g*')
plt.xlabel("ActualHeight")
plt.ylabel("Weight")
plt.legend(["Yaa","Ypp"])
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-f569b4969215> in <module>
      1 #plot graph between ActualHeight vs weight and predicted height vs weight
----> 2 plt.plot(X,Yaa,'ro')
      3 plt.plot(X,Ypp,'g*')
      4 plt.xlabel("ActualHeight")
      5 plt.ylabel("Weight")

NameError: name 'X' is not defined
```

In [ ]:
```
```