ML ASSIGNMENT NAME:M.KAUSHIK SAI

FIRSTLY WE IMPORT PANDAS MODULE(FOR MAKING A TABLE) ,NUMPY(FOR RANDOM INTEGERS)

In [ ]:
```
import pandas as pd
import numpy as np
from numpy.random import rand
from random import randint
b=[]
c=[]
a =pd.DataFrame(rand(100,3)*10,columns='study sleep IQ'.split())
for i in range(100):
    b.append(randint(2,12))
    c.append(randint( 40 , 140))
a['IQ']=c
a['sleep']=b
```

WE ARE TAKING INPUTS AS IQ,STUDY,SLEEP,LOCATION FOR PREDICTION OF MARKS

In [ ]:
```
from random import choice

location = []
for i in range(100):

    location.append(choice(["kerala","Hyderabad","Foreign country","New Delhi" , "Pune

a['Location'] = location
```

WE USE CHOICE TO MAKE SURE THAT THERE IS SOME VARIATION IN DATA

In [ ]:
```
from random import choice
marks=[]
marks=(a['study']*2-a['sleep']+a['IQ']/2+choice([10,2,8,4,5,-2,-1,-6,-7,-4,-8])+randir
for i in range(len(marks)):
    marks[i]=int(marks[i])
a['marks']=marks


a
```

Out[ ]:

	study	sleep	IQ	Location	marks
0	0.859285	9	123	kerala	44.0
1	3.519429	8	85	Foreign country	31.0
2	0.551947	3	52	New Delhi	14.0
3	3.737677	6	54	New Delhi	18.0
4	0.235852	5	68	Pune	19.0
...	...	...	...	...	...
95	6.246655	8	71	Pune	29.0
96	0.349084	10	51	Bangalore	6.0
97	5.259997	11	114	kerala	46.0
98	0.904952	11	110	Pune	35.0
99	5.627944	6	119	Pune	54.0

100 rows × 5 columns

LABEL ENCODER HAS BEEN USED TO CONVERT (LOCATION) STRING TO INTEGER DATATYPE FOR UPCOMING CALCULATION

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
z=LabelEncoder()
a['Location'] = z.fit_transform(a['Location'])
```

HERE X=INPUT DATA,Y=OUTPUT DATA

In [ ]:
```
x=a.iloc[:,:-1].values
y=a.iloc[:,-1].values
```

In [ ]:
```
x
```

Out[ ]:
```
array([[8.59284561e-01, 9.00000000e+00, 1.23000000e+02, 5.00000000e+00],
       [3.51942949e+00, 8.00000000e+00, 8.50000000e+01, 1.00000000e+00],
       [5.51947382e-01, 3.00000000e+00, 5.20000000e+01, 3.00000000e+00],
       [3.73767741e+00, 6.00000000e+00, 5.40000000e+01, 3.00000000e+00],
       [2.35852462e-01, 5.00000000e+00, 6.80000000e+01, 4.00000000e+00],
       [7.55386644e+00, 7.00000000e+00, 4.80000000e+01, 5.00000000e+00],
       [4.29403481e+00, 4.00000000e+00, 5.10000000e+01, 5.00000000e+00],
       [4.79262440e+00, 3.00000000e+00, 1.00000000e+02, 1.00000000e+00],
       [7.51631057e+00, 1.20000000e+01, 1.05000000e+02, 2.00000000e+00],
       [4.00480669e+00, 3.00000000e+00, 8.50000000e+01, 3.00000000e+00],
       [2.09036520e+00, 9.00000000e+00, 7.40000000e+01, 2.00000000e+00],
       [6.40593123e+00, 7.00000000e+00, 5.60000000e+01, 4.00000000e+00],
       [1.34272065e+00, 4.00000000e+00, 1.24000000e+02, 0.00000000e+00],
       [2.43003553e+00, 1.10000000e+01, 7.70000000e+01, 2.00000000e+00],
       [2.64451314e+00, 7.00000000e+00, 8.90000000e+01, 1.00000000e+00],
       [7.78659407e+00, 2.00000000e+00, 5.10000000e+01, 2.00000000e+00],
       [5.26038972e+00, 3.00000000e+00, 7.30000000e+01, 2.00000000e+00],
       [6.04041461e+00, 9.00000000e+00, 5.60000000e+01, 3.00000000e+00],
       [3.27501105e+00, 3.00000000e+00, 8.70000000e+01, 2.00000000e+00],
       [2.82789370e+00, 4.00000000e+00, 1.08000000e+02, 3.00000000e+00],
       [8.75673587e+00, 5.00000000e+00, 4.30000000e+01, 2.00000000e+00],
       [1.11369145e+00, 1.10000000e+01, 1.09000000e+02, 2.00000000e+00],
       [2.20838935e+00, 1.20000000e+01, 1.37000000e+02, 3.00000000e+00],
       [8.33283135e+00, 7.00000000e+00, 6.60000000e+01, 3.00000000e+00],
       [7.06900636e+00, 1.20000000e+01, 1.01000000e+02, 3.00000000e+00],
       [6.29172611e+00, 1.00000000e+01, 4.70000000e+01, 3.00000000e+00],
       [6.31816087e+00, 1.00000000e+01, 7.00000000e+01, 5.00000000e+00],
       [1.33893106e+00, 4.00000000e+00, 9.30000000e+01, 2.00000000e+00],
       [3.99554973e+00, 5.00000000e+00, 5.20000000e+01, 1.00000000e+00],
       [8.39288095e+00, 8.00000000e+00, 9.40000000e+01, 3.00000000e+00],
       [9.23448464e+00, 8.00000000e+00, 1.18000000e+02, 2.00000000e+00],
       [1.94645039e+00, 4.00000000e+00, 7.30000000e+01, 0.00000000e+00],
       [9.79025668e+00, 7.00000000e+00, 6.00000000e+01, 5.00000000e+00],
       [5.59116001e+00, 8.00000000e+00, 5.10000000e+01, 3.00000000e+00],
       [6.03892929e+00, 6.00000000e+00, 5.50000000e+01, 1.00000000e+00],
       [9.60050032e+00, 1.00000000e+01, 1.38000000e+02, 1.00000000e+00],
       [7.93907741e+00, 6.00000000e+00, 7.50000000e+01, 4.00000000e+00],
       [8.37618536e+00, 1.00000000e+01, 7.30000000e+01, 0.00000000e+00],
       [8.08790981e+00, 1.00000000e+01, 1.38000000e+02, 5.00000000e+00],
       [1.19703464e+00, 5.00000000e+00, 1.03000000e+02, 5.00000000e+00],
       [1.54006679e+00, 1.00000000e+01, 7.70000000e+01, 2.00000000e+00],
       [9.24768531e+00, 1.10000000e+01, 8.80000000e+01, 0.00000000e+00],
       [9.06246665e+00, 1.20000000e+01, 8.40000000e+01, 5.00000000e+00],
       [7.35459743e+00, 5.00000000e+00, 8.50000000e+01, 4.00000000e+00],
       [8.91522992e+00, 1.20000000e+01, 5.50000000e+01, 0.00000000e+00],
       [8.39647239e+00, 3.00000000e+00, 4.60000000e+01, 2.00000000e+00],
       [3.15439253e+00, 1.00000000e+01, 1.25000000e+02, 4.00000000e+00],
       [6.92739997e+00, 1.00000000e+01, 7.40000000e+01, 1.00000000e+00],
       [5.85507512e-01, 1.20000000e+01, 1.20000000e+02, 3.00000000e+00],
       [8.17608566e+00, 9.00000000e+00, 8.10000000e+01, 0.00000000e+00],
       [6.49146700e+00, 6.00000000e+00, 7.50000000e+01, 4.00000000e+00],
       [4.39327693e+00, 1.00000000e+01, 1.09000000e+02, 3.00000000e+00],
       [1.06492944e+00, 9.00000000e+00, 1.21000000e+02, 5.00000000e+00],
       [3.98398231e+00, 1.00000000e+01, 8.10000000e+01, 1.00000000e+00],
       [5.43253119e+00, 9.00000000e+00, 1.27000000e+02, 4.00000000e+00],
       [7.79435798e+00, 9.00000000e+00, 1.39000000e+02, 2.00000000e+00],
       [7.86278030e+00, 8.00000000e+00, 6.70000000e+01, 0.00000000e+00],
       [4.98843428e+00, 7.00000000e+00, 7.40000000e+01, 5.00000000e+00],
       [5.11069594e+00, 7.00000000e+00, 4.60000000e+01, 4.00000000e+00],
       [7.71424316e+00, 7.00000000e+00, 8.10000000e+01, 0.00000000e+00],
```

```
[7.98750697e+00, 1.10000000e+01, 1.04000000e+02, 2.00000000e+00],
[4.92630252e+00, 7.00000000e+00, 8.00000000e+01, 4.00000000e+00],
[2.76916995e+00, 1.00000000e+01, 1.10000000e+02, 5.00000000e+00],
[6.36228119e+00, 3.00000000e+00, 7.30000000e+01, 1.00000000e+00],
[4.49484108e+00, 7.00000000e+00, 1.39000000e+02, 4.00000000e+00],
[9.20188640e+00, 1.00000000e+01, 9.00000000e+01, 3.00000000e+00],
[6.73384637e+00, 6.00000000e+00, 1.34000000e+02, 1.00000000e+00],
[6.01598058e-01, 6.00000000e+00, 6.00000000e+01, 2.00000000e+00],
[6.29656844e-01, 1.00000000e+01, 1.31000000e+02, 4.00000000e+00],
[6.69882374e+00, 8.00000000e+00, 1.22000000e+02, 0.00000000e+00],
[2.43721907e+00, 8.00000000e+00, 1.37000000e+02, 1.00000000e+00],
[3.92050020e+00, 4.00000000e+00, 1.31000000e+02, 2.00000000e+00],
[7.18916932e+00, 1.10000000e+01, 7.90000000e+01, 5.00000000e+00],
[3.91343168e+00, 5.00000000e+00, 4.10000000e+01, 1.00000000e+00],
[8.67541172e-02, 9.00000000e+00, 4.20000000e+01, 0.00000000e+00],
[1.97326997e+00, 1.20000000e+01, 1.02000000e+02, 5.00000000e+00],
[6.52352103e+00, 2.00000000e+00, 1.12000000e+02, 2.00000000e+00],
[5.07316321e+00, 1.20000000e+01, 4.80000000e+01, 3.00000000e+00],
[4.75628502e+00, 1.10000000e+01, 9.40000000e+01, 0.00000000e+00],
[1.34846608e+00, 1.10000000e+01, 7.40000000e+01, 4.00000000e+00],
[7.39263726e+00, 3.00000000e+00, 5.40000000e+01, 0.00000000e+00],
[4.30379768e+00, 1.20000000e+01, 1.21000000e+02, 3.00000000e+00],
[3.32994366e+00, 8.00000000e+00, 7.30000000e+01, 4.00000000e+00],
[9.26572181e+00, 7.00000000e+00, 9.50000000e+01, 3.00000000e+00],
[4.52662524e+00, 2.00000000e+00, 6.10000000e+01, 4.00000000e+00],
[5.73858136e+00, 1.10000000e+01, 1.01000000e+02, 2.00000000e+00],
[1.06207240e-01, 4.00000000e+00, 9.40000000e+01, 5.00000000e+00],
[3.16873488e-01, 1.20000000e+01, 1.08000000e+02, 2.00000000e+00],
[7.87344004e+00, 1.10000000e+01, 1.26000000e+02, 5.00000000e+00],
[8.50615330e+00, 8.00000000e+00, 1.16000000e+02, 3.00000000e+00],
[5.66322776e+00, 1.10000000e+01, 5.40000000e+01, 1.00000000e+00],
[1.93940907e+00, 1.10000000e+01, 1.19000000e+02, 4.00000000e+00],
[4.98740997e+00, 7.00000000e+00, 5.90000000e+01, 4.00000000e+00],
[1.83577719e+00, 1.00000000e+01, 1.28000000e+02, 5.00000000e+00],
[4.18217228e+00, 1.00000000e+01, 4.70000000e+01, 2.00000000e+00],
[6.24665451e+00, 8.00000000e+00, 7.10000000e+01, 4.00000000e+00],
[3.49083786e-01, 1.00000000e+01, 5.10000000e+01, 0.00000000e+00],
[5.25999708e+00, 1.10000000e+01, 1.14000000e+02, 5.00000000e+00],
[9.04951880e-01, 1.10000000e+01, 1.10000000e+02, 4.00000000e+00],
[5.62794444e+00, 6.00000000e+00, 1.19000000e+02, 4.00000000e+00]])
```

SKLEARN MODULE IS USED TO SPLIT DATA INTO DATA FOR TESTING AND DATA FOR TRAINING

In [ ]:
```
from sklearn.model_selection import train_test_split
Xtrain,Xtest,ytrain,ytest=train_test_split(x,y,test_size=1/4)
print(Xtest)
print(ytest)
```

```
[[  0.85928456   9.         123.           5.        ]
 [  8.17608566   9.          81.           0.        ]
 [  8.39288095   8.          94.           3.        ]
 [  0.34908379  10.          51.           0.        ]
 [  4.30379768  12.         121.           3.        ]
 [  2.43721907   8.         137.           1.        ]
 [  6.04041461   9.          56.           3.        ]
 [  2.20838935  12.         137.           3.        ]
 [  4.52662524   2.          61.           4.        ]
 [  8.39647239   3.          46.           2.        ]
 [  7.35459743   5.          85.           4.        ]
 [  3.99554973   5.          52.           1.        ]
 [  6.92739997  10.          74.           1.        ]
 [  3.98398231  10.          81.           1.        ]
 [  4.7926244    3.         100.           1.        ]
 [  3.9205002    4.         131.           2.        ]
 [  3.51942949   8.          85.           1.        ]
 [  7.39263726   3.          54.           0.        ]
 [  1.54006679  10.          77.           2.        ]
 [  5.59116001   8.          51.           3.        ]
 [  9.60050032  10.         138.           1.        ]
 [  8.33283135   7.          66.           3.        ]
 [  9.79025668   7.          60.           5.        ]
 [  1.93940907  11.         119.           4.        ]
 [  2.43003553  11.          77.           2.        ]]
[44. 37. 45.  6. 47. 55. 21. 50. 27. 26. 42. 18. 30. 28. 46. 59. 31. 28.
 21. 18. 68. 32. 32. 42. 22.]
```

LINEAR REGRESSION GIVES A RELATION BETWEEN INPUT AND OUTPUT

In [ ]:
```
from sklearn.linear_model import LinearRegression
s=LinearRegression()
s.fit(Xtrain,ytrain)
```

Out[ ]:
```
▾ LinearRegression
LinearRegression()
```

PREDICT:PREDICTION OF OUTPUT USING LINEAR REGRESSION

In [ ]:
```
predictor=s.predict(Xtest)
```

In [ ]:
```
predictor
for i in range(25):
    predictor[i]=int(predictor[i])
```

In [ ]:
```
ytest
```

Out[ ]:
```
array([44., 37., 45.,  6., 47., 55., 21., 50., 27., 26., 42., 18., 30.,
       28., 46., 59., 31., 28., 21., 18., 68., 32., 32., 42., 22.])
```

In [ ]:
```
f=pd.DataFrame(rand(25,2),columns='expected_outcome predicted_outcome'.split())
f['expected_outcome']=ytest
f['predicted_outcome']=predictor
```

In [ ]:
```
f
```

Out[ ]:

	expected_outcome	predicted_outcome
0	44.0	43.0
1	37.0	37.0
2	45.0	45.0
3	6.0	5.0
4	47.0	46.0
5	55.0	54.0
6	21.0	20.0
7	50.0	50.0
8	27.0	27.0
9	26.0	26.0
10	42.0	41.0
11	18.0	18.0
12	30.0	30.0
13	28.0	27.0
14	46.0	46.0
15	59.0	58.0
16	31.0	31.0
17	28.0	28.0
18	21.0	21.0
19	18.0	18.0
20	68.0	67.0
21	32.0	32.0
22	32.0	32.0
23	42.0	41.0
24	22.0	21.0