① Implementing simple/single Linear regression using Python. [Here we have 1 dependent & 1 Independent variable.]

Area = Independent var.
Price = Dependent var.

Step-①:- Importing the packages.

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as pt.
```

Step-②:- Reading the data. [From Excel to Python platform.]

```
df = pd.read_csv("prc.csv")
```

→ File name.

```
print (df)
```

Step-③:- Creating the model.

↳ By creating an object of Linear Regression class.

```
model = linear_model.LinearRegression()
```

Step-④:- Train the model.

```
model.fit(df[['area']], df.price)
```

Step-⑤:- Finding slope and intercept.

```
print(model.coef_)
```

~~print(model.intercept)~~

```
print (model.intercept_)
```

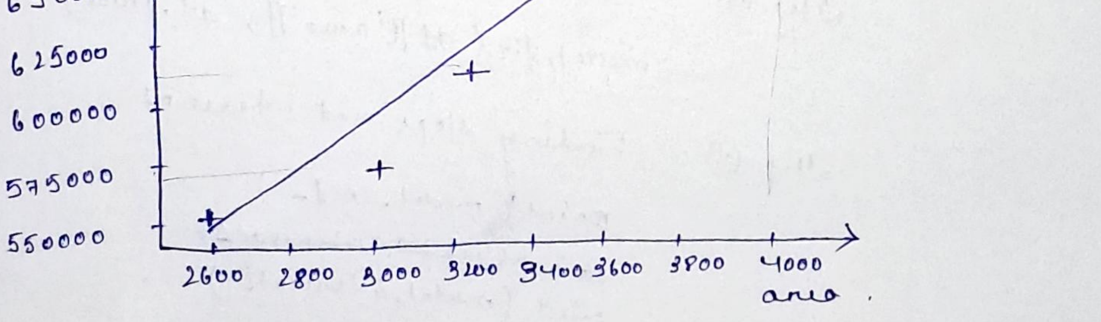
o/p of Prog-①:-

```
   area     price
0  2600     550000
1  3000     565000
2  3200     610000
3  3600     680000
4  4000     ~~680000~~ 725000
```

[135.78767123] → Coefficient / Slope.

180616.43835616432 → Intercept.

array([683030.82191]) → Predicted o/p.



Step-⑥:- Predicting the output.

```
model.predict([[3700]])
```

→ It will predict the price for area: 3700 sq.ft

Step-⑦:- Plotting the graph.

```
pt.scatter(df.area, df.price, color = "red", maker = "+")
pt.plot(df.area, model.predict(df[['area']]))
```

① Implementing multi-linear regression using python.

[Here we have, 1 → Dependent variable
multi → Independent variable]

Dependent = price.
Independent var = area, bedrooms, age.

Step-①:- Importing the packages.

```
import pandas as pd
import numpy as nm
from sklearn import linear_model
import matplotlib.pyplot as plt.
```

Step-②:- Reading the data. [From Excel]

```
df1 = pd.read_csv("homeprices.csv")
```

↳ file name

```
print(df1)
```

O/P of Prog - ②:-

[Initial dataset]-

```
     area   bedrooms   age   price.
0    2600   3.0        20    550000
1    3000   4.0        15    565000
2    3200   NaN        18    610000
3    3600   3.0        30    595000
4    4000   5.0        8     760000
5    4100   6.0        8     810000
```

Not a no.

[Final dataset]

```
     area   bedrooms   age   price.
0    2600   3.0
1    3000   4.0
2    3200   4.0
3    3600   3.0
4    4000   5.0
5    4100   6.0
```

NaN has been replaced with the value 4.0.

```
[112.06244194   23388.880077   -3231.71790]
```

[Coefficients]

```
221323.00186540384
```

→ [Intercept].

```
array([1001713.75489952])
```

↳ Predicted price when
area = 6000
bedrooms = 6
age = 10.

Step - (3):- Replacing the NAN values with median of that respective column.

```
df1.bedrooms = df1:bedrooms.fillna(df1.bedrooms.median())

print (df1)
```

Step - (4):- Creating the dataframe.] ⇒ Needed to drop a column.

```
df1 = pd.DataFrame (df1)
```

Step - (5):- Setting Dependent & Independent variable.

```
X = df1.drop( columns = ['price'])
Y = df1.price.
```

Step - (6):- Creating the model.
↳ By creating an object of LinearRegression class.

```
model = linear_model.LinearRegression()
```

Step - (7):- Train the model.

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

Step - (8):- Finding slope and coefficient of the model.

```
print(model.coef_)
print(model.intercept_)
```

Step-⑨:- Predicting the O/P from the model.

model.predict([[6000, 6, 10]])

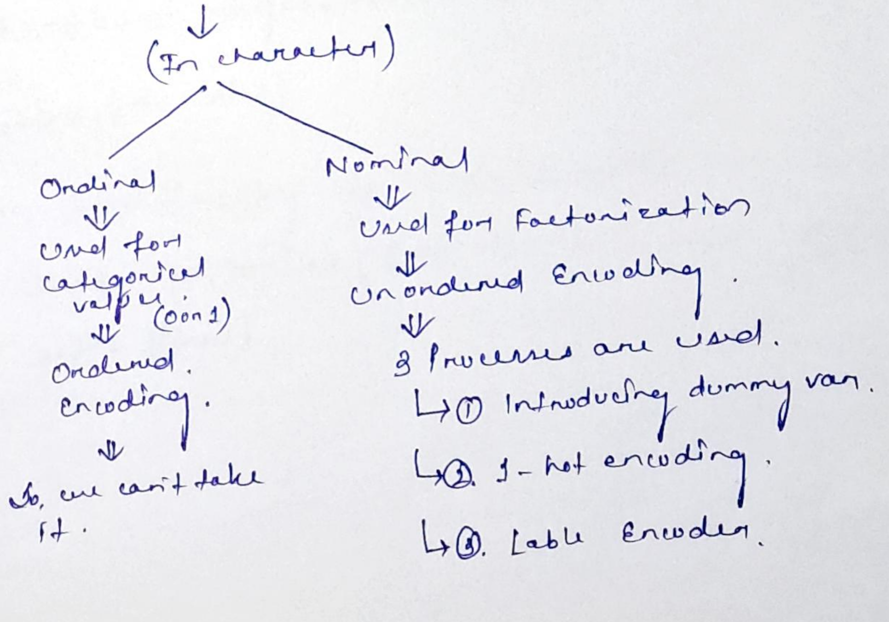
↳ Meaning:-
we want to find
out the value of
price, when
area = 6000,
bedrooms = 6
age = 10.

∴ Predicted O/P:-

1001713.75489952.

Q Converting the non-numeric formatted column into numeric formatted column. [using dummy var.]

Let the non-numeric formatted column be "text"


(In character)

Ordinal
⇓
Used for categorical values (0 on 1)
⇓
Ordered encoding.
⇓
So, we can't take it.

Nominal
⇓
Used for Factorization
⇓
Unordered Encoding.
⇓
3 Processes are used.
↳① Introducing dummy var.
↳②. 1-hot encoding.
↳③. Lable Encoder.

[Using Dummy var]

Step-①:- Importing the packages.

```
import pandas as pd
import numpy as nm
from sklearn import linear_model
import matplotlib.pyplot as mpl.
```

Step-②:- Reading the data (From Excel)

```
df = pd.read_csv('homeprices.csv')
                  ↳ Filename

print (df)
```

Step-③:- Introducing the dummies.

Since, here non-numeric formatted column is "town", so we need to create dummies for the "town" column only.

```
dummies = pd.get_dummies(df['town'], dtype=int)

print (dummies)
```

Step-④:- Concatinating the dummies with the datased.

```
final = pd.concat([df, dummies], axis = 'columns')

print (final)
```

O/p of Prog-③:-   [Original dataset].

```
              town             area     price
0    monroe township           2600    550000
1    monroe township           3000    565000
2    monroe township           3200    610000
3    monroe township           3600    680000
4    monroe township           4000    725000
5    west windsor              2600    585000
6    west windsor              2800    615000
7    west windsor              3300    650000
8    west windsor              3600    710000
9    robinsville               2600    575000
10   robinsville               2900    600000
11   robinsville               3100    620000
12   robinsville               3600    695000
```

[Dummy]

```
     monroe township    robinsville    west windsor
0          1                 0              0
1          1                 0              0
2          1                 0              0
3          1                 0              0
4          1                 0              0
5          0                 0              1
6          0                 0              1
7          0                 0              1
8          0                 0              1
9          0                 1              0
10         0                 1              0
11         0                 1              0
12         0                 1              0
```

[Continued] →

Step-⑤:- Dropping one of the dummies to avoid "Dummy variable trap".

↳ If it occurs, then it can be predicted from the other variable.

↳ So, we need to remove it.

```
final = final.drop(['west windsor'], axis = "columns")
print (final)
```

Step-⑥:- Dropping that column whose dummies have been created.

```
finale = final.drop(['town'], axis = "columns")
print (finale)
```

Step-⑦:- Creating the dataframe.

```
df1 = pd.DataFrame(finale)
```

Step-⑧:- Setting the dependent and independent var.

```
X = df1. drop (columns = ['price'])
Y = df1. price.
```

Step-⑨:- Creating the model.

↳ By creating an object of LinearRegression class.

```
model = linear_model.LinearRegression()
```

O/p of Prog - ③:-

→ [Continues] [Original data set] + [dummies] - ([town] + [west windsor])

	area	price	monroe township	robinsville
0	2600	550000	1	0
1	3000	565000	1	0
2	3200	610000	1	0
3	3600	680000	1	0
4	4000	725000	1	0
5	2600	585000	0	0
6	2800	615000	0	0
7	3300	650000	0	0
8	3600	710000	0	0
9	2600	575000	0	1
10	2900	600000	0	1
11	3100	620000	0	1
12	3600	695000	0	1

7907

[Slope / Coefficient of model]

↳ [126.89744141 -40013.97548914 -14327.56396474]

[Intercept]

↳ 249790.36766292521.

[Predicted value]

↳ array([590468.71640508])

∴ when area = 3000, town = monroe township,
price will be $590468.71

Step-(8):- Train the model.

```
model. fit (x, y)
```

Step-(9):- Finding the slope and coefficient of the model.

```
print (model.coef_)
print (model.intercept_)
```

Step-(10):- Predicting the O/p from the model.

```
model. predict ([[ 3000, 1, 0 ]])
```

↳ Meaning:- we want to predict the value of price from the model, when area = 3000 & town = monroe township.