In [1]:
```
print("Mahima Chauhan")
```

```
Mahima Chauhan
```

In [2]:
```
pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (1.4.
2)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.8/dist-package
s (from pandas) (1.22.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.8/dis
t-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.8/dist-packages
(from pandas) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from pytho
n-dateutil>=2.8.1->pandas) (1.14.0)
Note: you may need to restart the kernel to use updated packages.
```

In [23]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\kumar\appdata\local\programs\py
thon\python39\lib\site-packages (3.5.1)Note: you may need to restart the kernel to us
e updated packages.

Requirement already satisfied: cycler>=0.10 in c:\users\kumar\appdata\local\programs
\python\python39\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\kumar\appdata\local\prog
rams\python\python39\lib\site-packages (from matplotlib) (4.32.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\kumar\appdata\local\p
rograms\python\python39\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\kumar\appdata\local\progr
ams\python\python39\lib\site-packages (from matplotlib) (3.0.8)
Requirement already satisfied: numpy>=1.17 in c:\users\kumar\appdata\local\programs\p
ython\python39\lib\site-packages (from matplotlib) (1.22.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\kumar\appdata\local\programs
\python\python39\lib\site-packages (from matplotlib) (9.1.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\kumar\appdata\local\prog
rams\python\python39\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: packaging>=20.0 in c:\users\kumar\appdata\local\progra
ms\python\python39\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: six>=1.5 in c:\users\kumar\appdata\local\programs\pyth
on\python39\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.15.0)
```

In [52]:
```
pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\kumar\appdata\local\programs\pytho
n\python39\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\kumar\appdata\local\programs
\python\python39\lib\site-packages (from sklearn) (1.0.2)
Requirement already satisfied: scipy>=1.1.0 in c:\users\kumar\appdata\local\programs
\python\python39\lib\site-packages (from scikit-learn->sklearn) (1.8.0)
Requirement already satisfied: numpy>=1.14.6 in c:\users\kumar\appdata\local\programs
\python\python39\lib\site-packages (from scikit-learn->sklearn) (1.22.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\kumar\appdata\local\p
rograms\python\python39\lib\site-packages (from scikit-learn->sklearn) (3.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\kumar\appdata\local\programs
\python\python39\lib\site-packages (from scikit-learn->sklearn) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [101…
```
#pip install seaborn
```

In [83]:
```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importing Data
from sklearn.datasets import load_boston
boston = load_boston()
```

In [84]:
```
boston.data.shape
```

Out[84]:
```
(506, 13)
```

In [85]:
```
data = pd.DataFrame(boston.data)
data.columns = boston.feature_names

data.head(10)
```

Out[85]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10

In [86]:
```
# Adding 'Price' (target) column to the data
boston.target.shape
```

Out[86]:
```
(506,)
```

In [102…
```
data = pd.DataFrame(boston.data)
data.columns = boston.feature_names

data.head(10)
```

Out[102]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10

In [89]:
```
# Adding 'Price' (target) column to the data
boston.target.shape
```

Out[89]:
```
(506,)
```

In [90]:
```
data['Price'] = boston.target
data.head()
```

Out[90]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Pr
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	2
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	2
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	3
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	3
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	3

In [91]:
```
data.describe()
```

Out[91]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DI
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.79504
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.10571
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12960
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.10017
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.20745
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.18842
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12650

In [92]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  Price    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [93]:
```
# Input Data
x = boston.data

# Output Data
y = boston.target


# splitting data to training and testing dataset.

#from sklearn.cross_validation import train_test_split
#the submodule cross_validation is renamed and reprecated to model_selection
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,


print("xtrain shape : ", xtrain.shape)
print("xtest shape : ", xtest.shape)
```

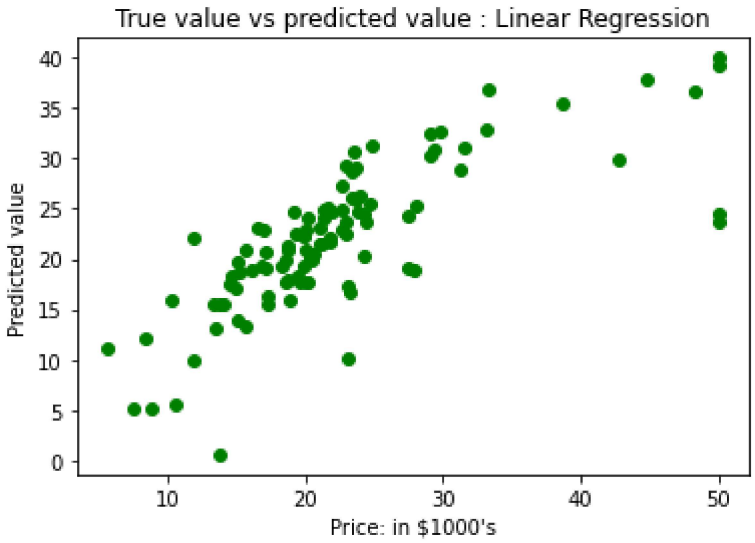
```
print("ytrain shape : ", ytrain.shape)
print("ytest shape : ", ytest.shape)
```

```
xtrain shape :  (404, 13)
xtest shape :  (102, 13)
ytrain shape :  (404,)
ytest shape :  (102,)
```

In [94]:
```
# Fitting Multi Linear regression model to training model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)

# predicting the test set results
y_pred = regressor.predict(xtest)
```

In [95]:
```
# Plotting Scatter graph to show the prediction
# results - 'ytrue' value vs 'y_pred' value
plt.scatter(ytest, y_pred, c = 'green')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```



In [96]:
```
# Results of Linear Regression.
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(ytest, y_pred)
print("Mean Square Error : ", mse)
```

```
Mean Square Error :  33.448979997676474
```