In [121]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
from sklearn import datasets
```

In [122]:

```
data = datasets.load_boston()
print(data.DESCR)
```

```
Boston House Prices dataset
===========================

Notes
------
Data Set Characteristics:

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive

    :Median Value (attribute 14) is usually the target

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
http://archive.ics.uci.edu/ml/datasets/Housing


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
problems.

**References**

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learnin
g, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
   - many more! (see http://archive.ics.uci.edu/ml/datasets/Housing)
```

In [123]:

```
boston = pd.DataFrame(data.data, columns = data.feature_names)
boston.head()
```

Out[123]:

	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

In [124]:

```
# Adding Home Values(target array) to the data frame.
boston['MEDV']=data.target
```

In [125]:

```
boston.head()
```

Out[125]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
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	24.0
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	21.6
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	34.7
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	33.4
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	36.2

In [126]:

```
boston.shape
```

Out[126]: (506, 14)

In [196]:
```
#Create and reshape feature(X) and target(y) variables
X = boston['RM'].values.reshape(-1,1)
y = boston['MEDV'].values.reshape(-1,1)
```

In [128]:
```
#instatiate linear regression and create prediction space
from sklearn.linear_model import LinearRegression
reg = LinearRegression()

prediction_space = np.linspace(min(X), max(X)).reshape(-1,1)
```

In [129]:
```
#Fitting the linear regression model
reg.fit(X,y)
```

Out[129]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

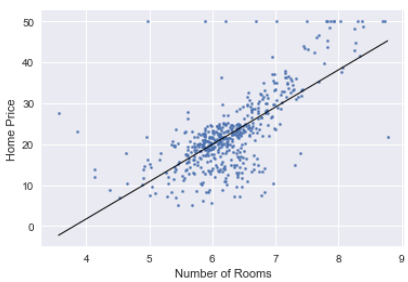
In [130]:
```
#Calculating predictions over the prediction_space.
y_pred = reg.predict(prediction_space)
```

In [131]:
```
#R^2 error value
print(reg.score(X, y))
```

```
0.4835254559913343
```

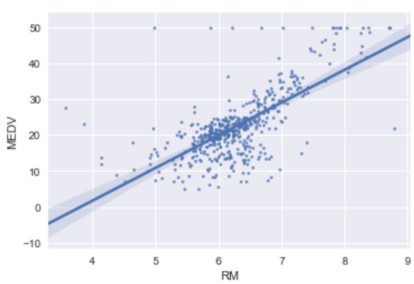
In [167]:
```
#Matplot lib graph with regression line
marker_size=5
plt.scatter(X,y, marker_size)
plt.xlabel('Number of Rooms')
plt.ylabel('Home Price')
plt.plot(prediction_space, y_pred, color='black', linewidth=1)
```

Out[167]:
```
[<matplotlib.lines.Line2D at 0x1a16644898>]
```



In [164]:
```
#Seaborn plot with regression line.
sns.regplot(boston['RM'],boston['MEDV'], data=boston, scatter = True, fit_reg=True, scatter_kws={'s':8})
```

Out[164]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a16461fd0>
```



In [168]:
```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

In [197]:
```
# Train, test, and fitting the linear regression model.
X_test, X_train, y_test, y_train = train_test_split(X, y, test_size=.3, random_state=4)
reg1 = LinearRegression()
reg1.fit(X_train, y_train)
```

Out[197]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [198]:
```
y_pred = reg1.predict(X_test)
```

In [199]:
```
# Calculating R^2 and RMSE model evalution metrics
print("R^2: {}".format(reg1.score(X_test, y_test)))
RMSE = np.sqrt(mean_squared_error(y_test,y_pred))
print('RMSE: {}'.format(RMSE))
```

```
R^2: 0.4644519760601598
RMSE: 6.349105770588851
```

In [207]:
```python
#Plot of predicted prices versus actual prices
marker_size=5
plt.scatter(y_test, y_pred, marker_size)
plt.xlabel('Prices')
plt.ylabel('Predicted Prices')
```

Out[207]: Text(0,0.5,'Predicted Prices')

In [209]:
```python
# 5-fold cross-validation scores for the linear regression model.
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(reg1, X, y, cv=5)
print(cv_scores)
```

```
[ 0.70708692  0.63476138  0.50385441 -0.21594318 -1.77736913]
```