In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [ ]:
```
```

In [2]:
```
data = pd.read_csv('Boston.csv')
data.head(10)
```

Out[2]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90
5	6	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12
6	7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60
7	8	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90
8	9	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63
9	10	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71

In [3]:
```
data.shape
```

Out[3]:
```
(506, 15)
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  506 non-null    int64  
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64  
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64  
 10  tax         506 non-null    int64  
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

In [5]:
```
data.columns
```

Out[5]:
```
Index(['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'di
s',
       'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

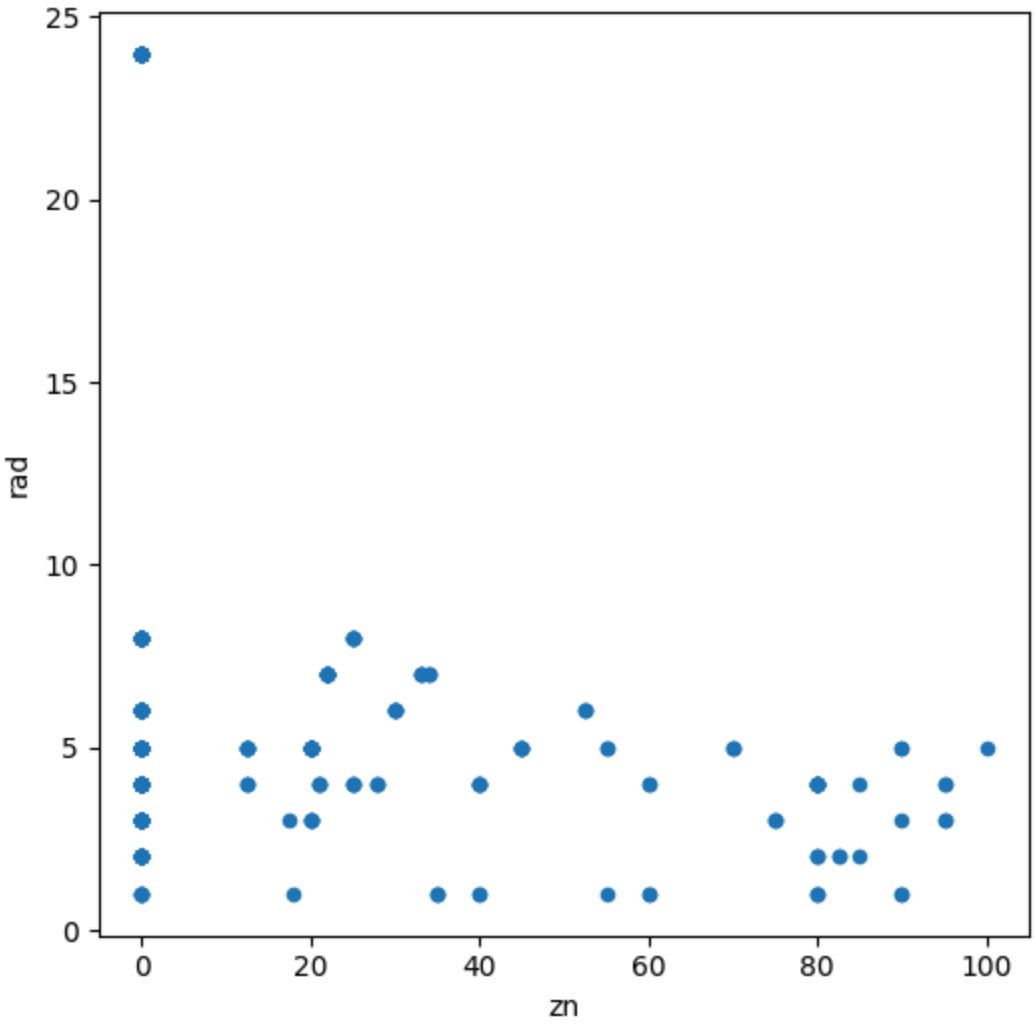
In [6]:
```
data.describe()
```

Out[6]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.(
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.5
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.1
min	1.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.9
25%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.(
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.5
75%	379.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.(
max	506.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.(

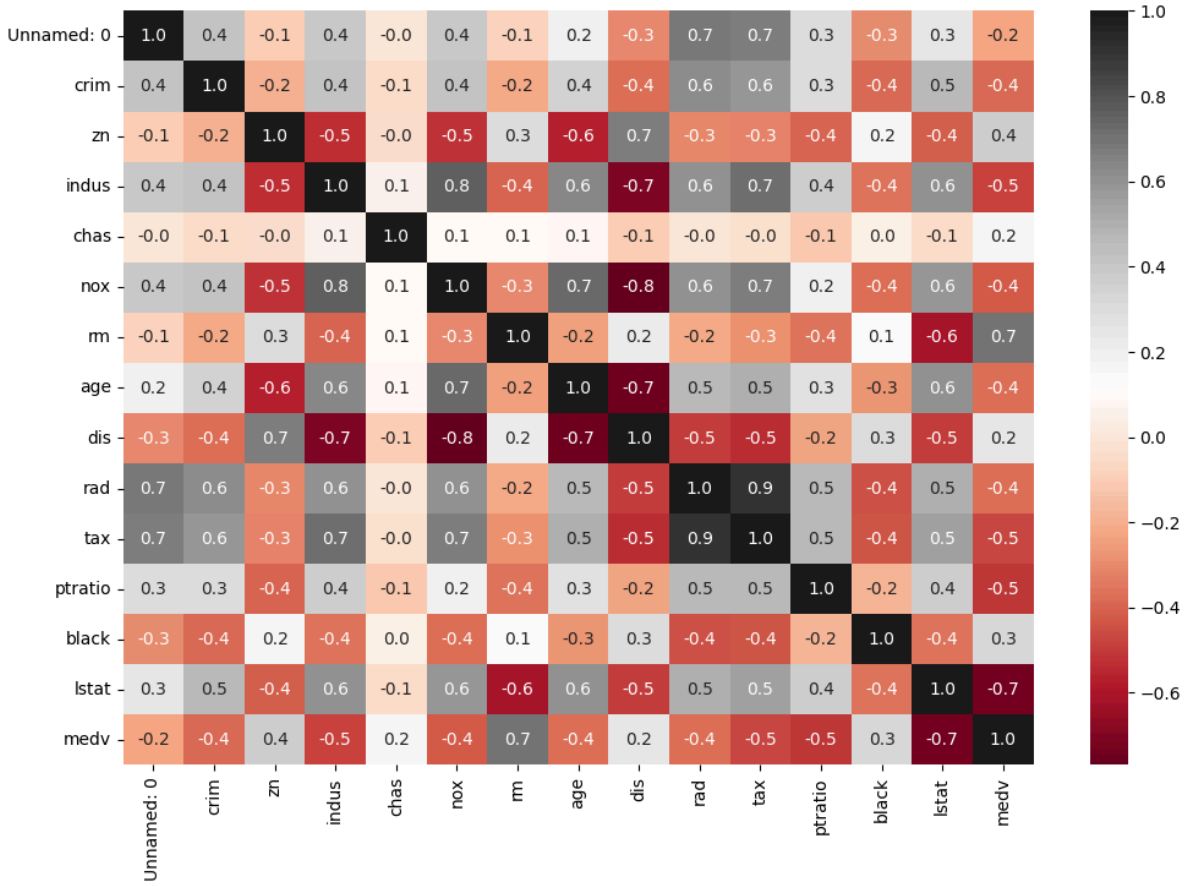
In [8]:
```
data.plot.scatter('zn', 'rad', figsize=(6, 6))
```

Out[8]: <Axes: xlabel='zn', ylabel='rad'>



In [9]:
```
plt.subplots(figsize=(12,8))
sns.heatmap(data.corr(), cmap = 'RdGy', annot = True, fmt = '.1f')
```

Out[9]: <Axes: >



In [17]:
```
X = data[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'ta
x','ptratio','black','lstat']]
Y = data['medv']
```

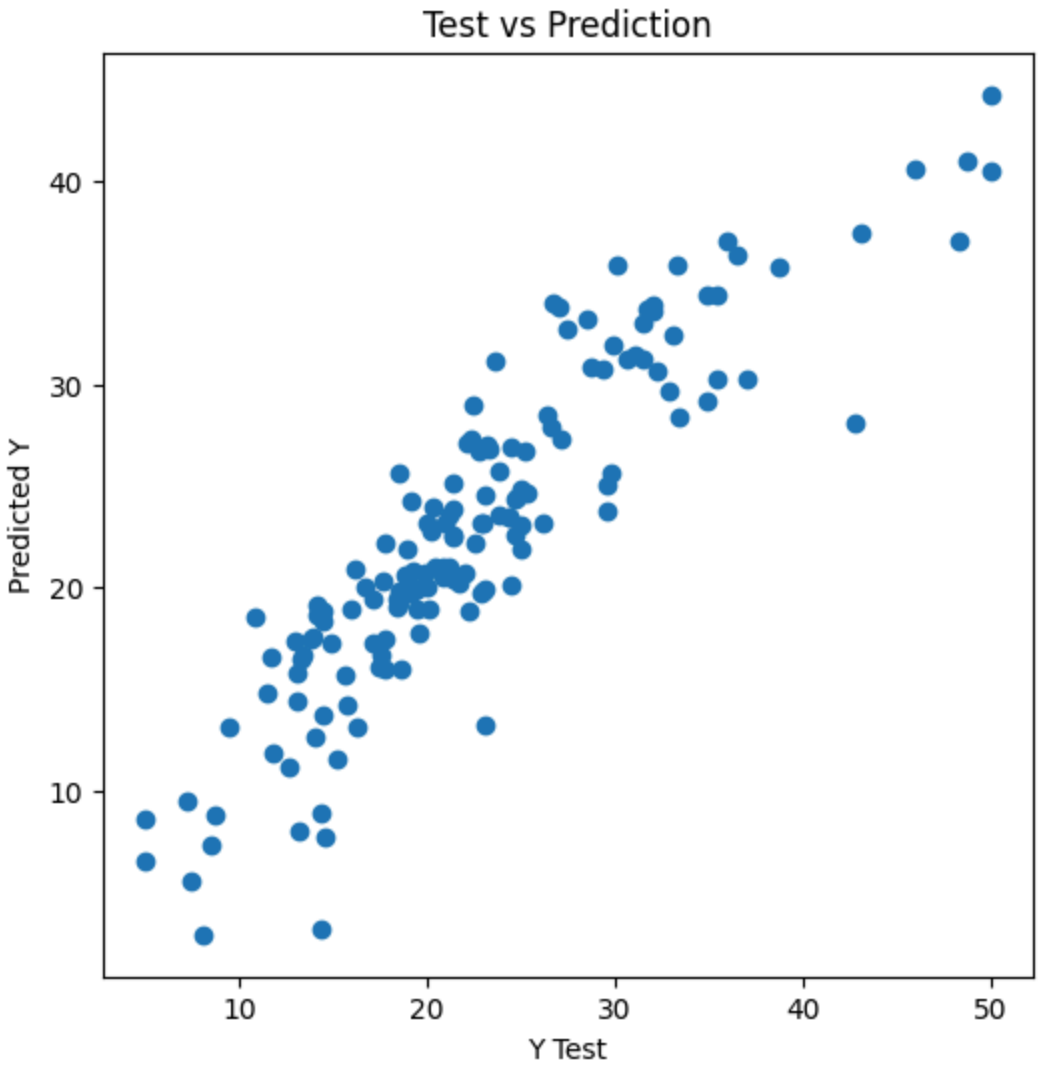
In [18]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
```

```
Train Dataset Size - X: (354, 13), Y: (354,)
Test Dataset Size - X: (152, 13), Y: (152,)
```

In [19]:
```
lm = LinearRegression()
lm.fit(X_train,Y_train)
predictions = lm.predict(X_test)
```

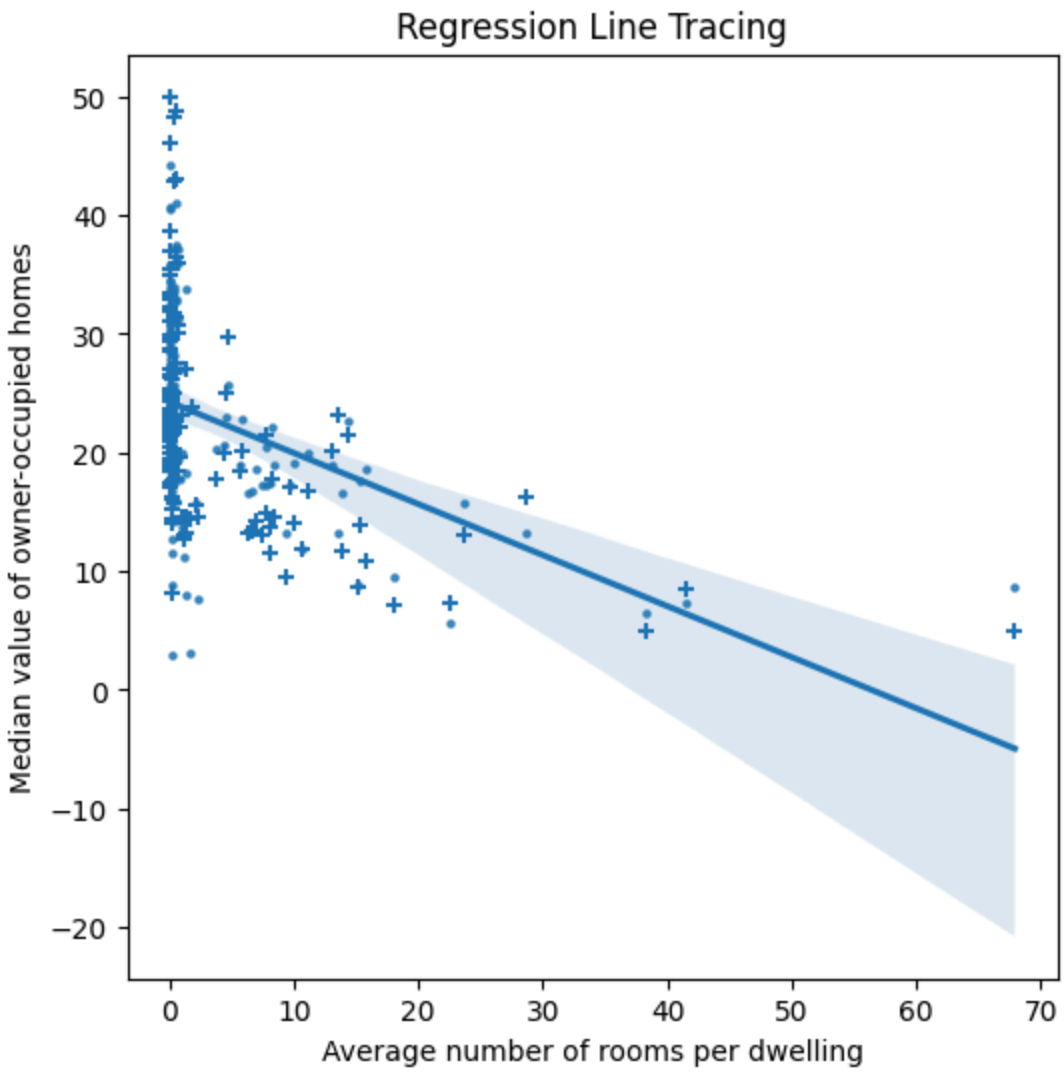
In [20]:
```
plt.figure(figsize=(6, 6))
plt.scatter(Y_test, predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.title('Test vs Prediction')
```

Out[20]: Text(0.5, 1.0, 'Test vs Prediction')



In [22]:
```
plt.figure(figsize=(6, 6))
sns.regplot(x = X_test['crim'], y = predictions, scatter_kws={'s':5})
plt.scatter(X_test['crim'], Y_test, marker = '+')
plt.xlabel('Average number of rooms per dwelling')
plt.ylabel('Median value of owner-occupied homes')
plt.title('Regression Line Tracing')
```

Out[22]: Text(0.5, 1.0, 'Regression Line Tracing')



In [23]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(Y_test, prediction
s))
print('Mean Square Error:', metrics.mean_squared_error(Y_test, predictions))
print('Root Mean Square Error:', np.sqrt(metrics.mean_squared_error(Y_test, pr
edictions,)))
```

```
Mean Absolute Error: 2.9200428009774444
Mean Square Error: 14.889045529138269
Root Mean Square Error: 3.858632598361532
```

In [24]:
```
coefficients = pd.DataFrame(lm.coef_.round(2), X.columns)
coefficients.columns = ['coefficients']
coefficients
```

Out[24]:

	coefficients
crim	-0.11
zn	0.05
indus	0.01
chas	2.86
nox	-21.46
rm	3.28
age	0.02
dis	-1.60
rad	0.37
tax	-0.01
ptratio	-1.04
black	0.01
lstat	-0.59

In [ ]:
```

```