In [2]:
```
import pandas as pd
df=pd.read_csv('HousingDB.csv')
```

In [3]:
```
df
```

Out[3]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

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

Out[5]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

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

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000

In [7]:
```
df.info()
```

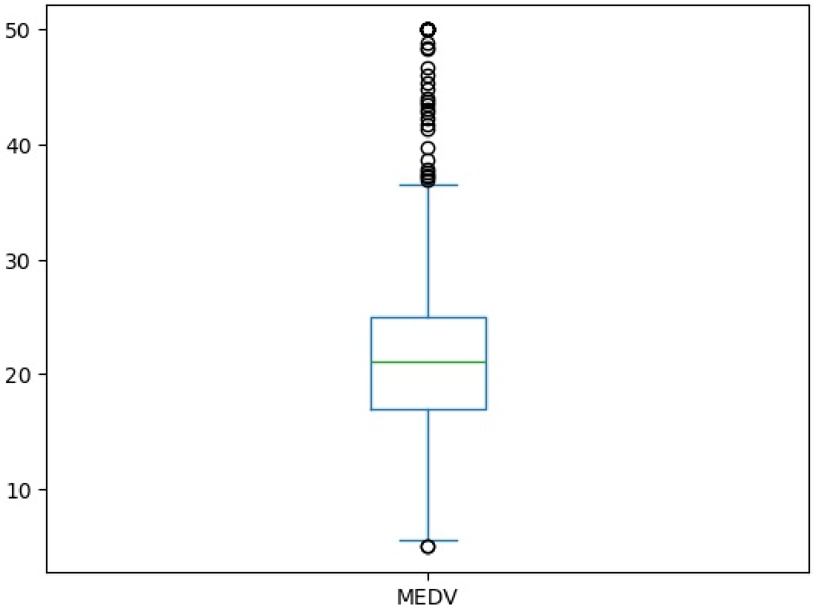
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 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    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
#To plot the graph embedded in the notebook
%matplotlib inline
```

In [9]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

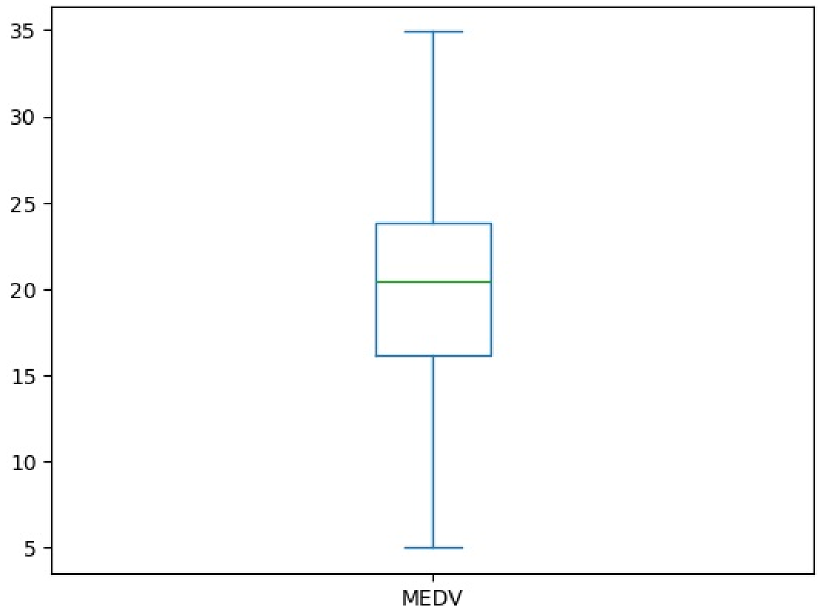
In [26]:
```
df['MEDV'].plot.box()
```

Out[26]: <Axes: >



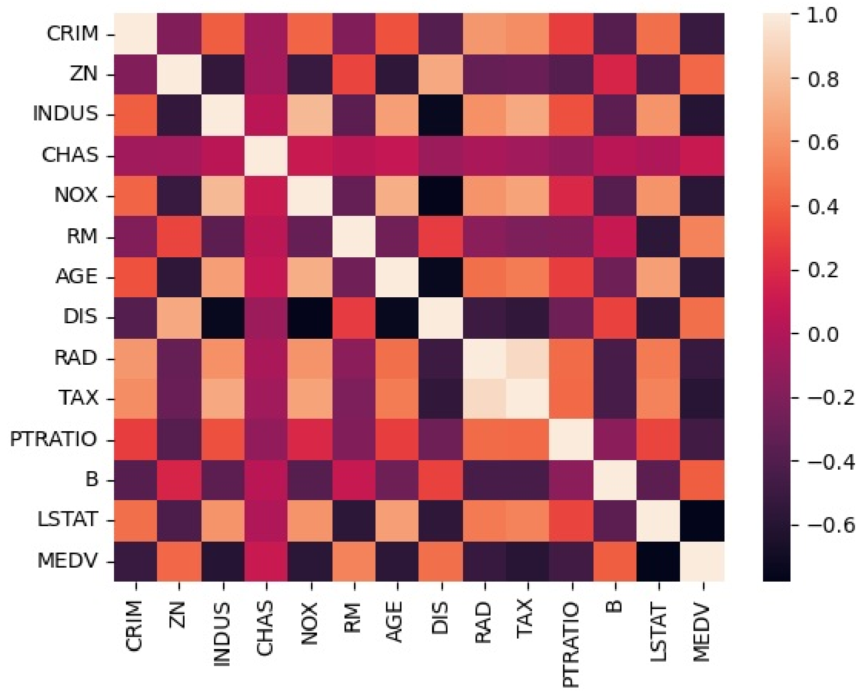
In [30]:
```
df=df[df['MEDV']<=35]
df['MEDV'].plot.box()
```

Out[30]: <Axes: >



In [31]:
```
correlation_matrix=df.corr().round(2)
sns.heatmap(data=correlation_matrix)
```

Out[31]: <Axes: >

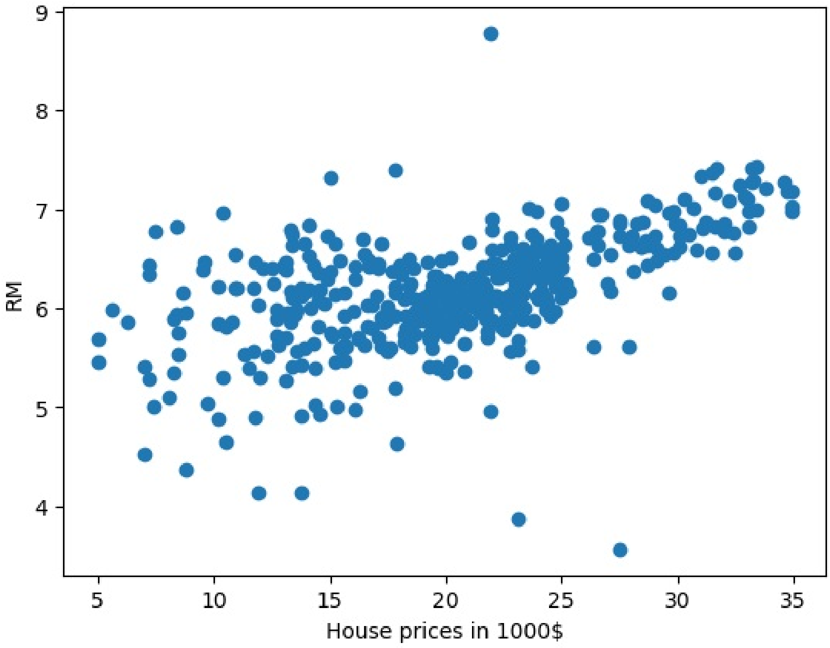


The correlation coefficient ranges from -1 to 1. If the value is close to 1, it means that there is a strong positive correlation between the two variables. When it is close to -1, the variables have a strong negative correlation

Look at the MEDV , it has strong correlation wiht RM

In [32]:
```
plt.scatter(df['MEDV'],df['RM'])
plt.xlabel('House prices in 1000$')
plt.ylabel('RM')
```

Out[32]: Text(0, 0.5, 'RM')



From above we can see that,as House prices increases, RM also increases

In [33]:
```
x=df[['RM']]
y=df['MEDV']
```

In [34]:
```
print(x.shape)
print(y.shape)
```

```
(458, 1)
(458,)
```

Split data into 20:80 ratio. 80 will be training data sn 20 will be testing data

In [35]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=5)
```

In [36]:
```
model = LinearRegression()
model.fit(x_train, y_train)
```

Out[36]:
```
▾ LinearRegression
LinearRegression()
```

In [37]:
```
y_pred=model.predict(x_test)
```

In [38]:
```
mse=mean_squared_error(y_test,y_pred)
mse
```

Out[38]: 22.905315826290995

In [39]:
```
import numpy as np
rmse=np.sqrt(mse)
rmse
```

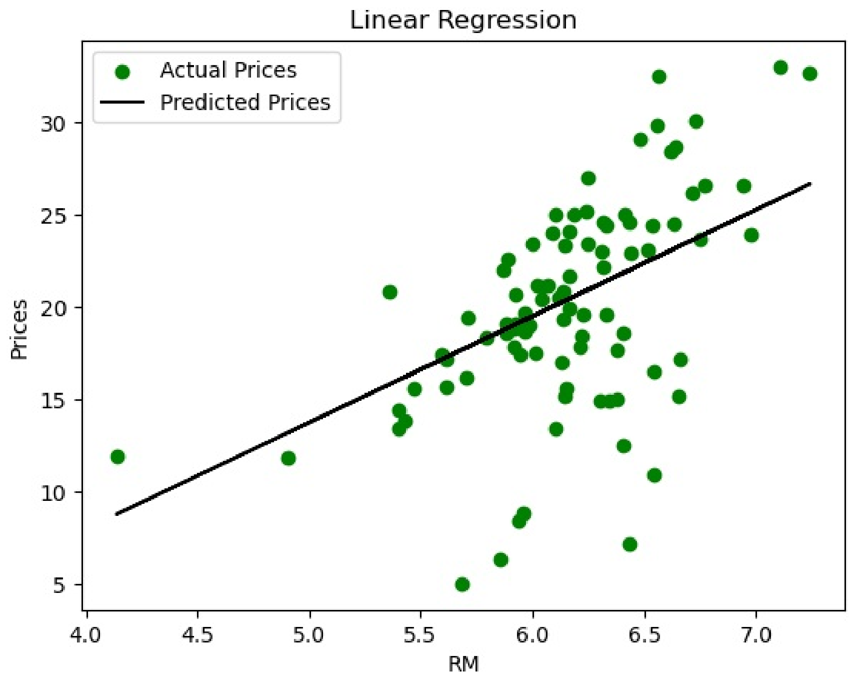
Out[39]: 4.785949835329555

In [40]:
```
r2_score=model.score(x_test,y_test)
r2_score
```

Out[40]: 0.2951091986432811

In [41]:
```
plt.scatter(x_test,y_test,label='Actual Prices',color="green")
plt.plot(x_test,y_pred,label="Predicted Prices",color="black")
plt.title("Linear Regression")
plt.xlabel('RM')
plt.ylabel('Prices')
plt.legend(loc="best")
```

Out[41]: <matplotlib.legend.Legend at 0x29f24796250>



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js