In [4]:
```
# SALES PREDICTION USING PYTHON TASK(4)
```

In [5]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [6]:
```
df=pd.read_csv("advertising.csv")
```

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

Out[7]:
```
<bound method DataFrame.info of         TV  Radio  Newspaper  Sales
0    230.1   37.8       69.2   22.1
1     44.5   39.3       45.1   10.4
2     17.2   45.9       69.3   12.0
3    151.5   41.3       58.5   16.5
4    180.8   10.8       58.4   17.9
..     ...    ...        ...    ...
195   38.2    3.7       13.8    7.6
196   94.2    4.9        8.1   14.0
197  177.0    9.3        6.4   14.8
198  283.6   42.0       66.2   25.5
199  232.1    8.6        8.7   18.4

[200 rows x 4 columns]>
```

In [8]:
```
df.head()
```

Out[8]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

In [9]:
```
df.shape
```

Out[9]:
```
(200, 4)
```

In [10]:
```
df.describe()
```

Out[10]:

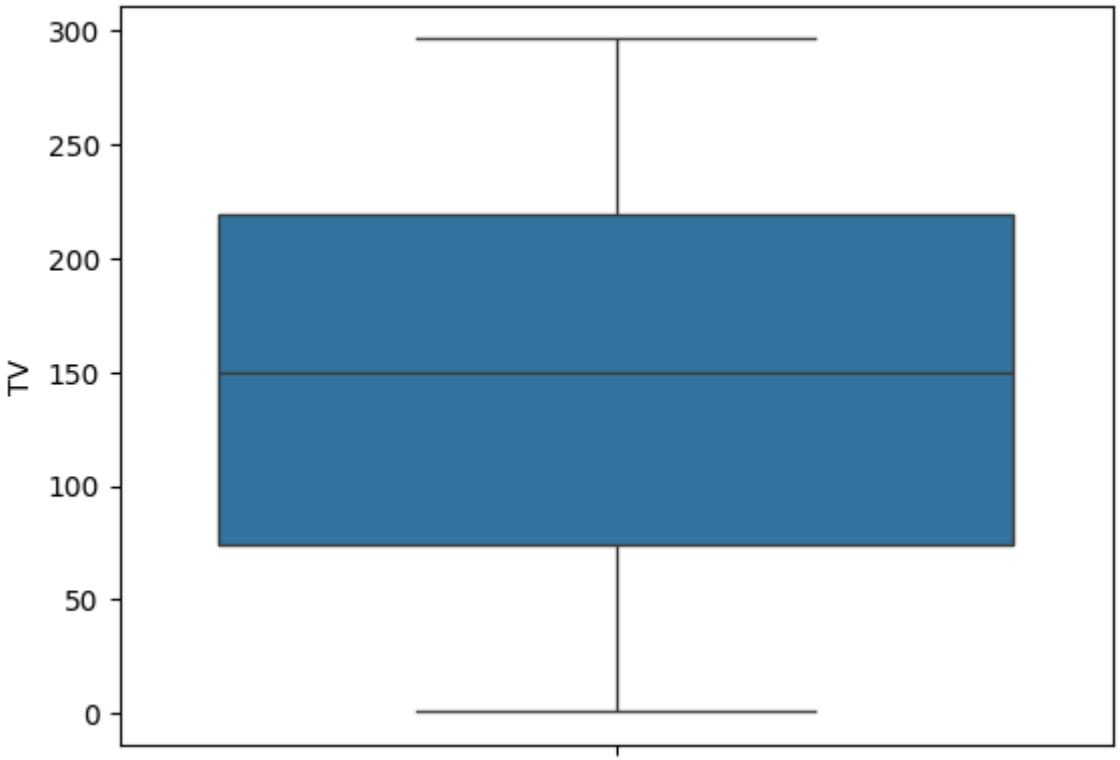
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

In [11]:
```
# Data Cleaning
```

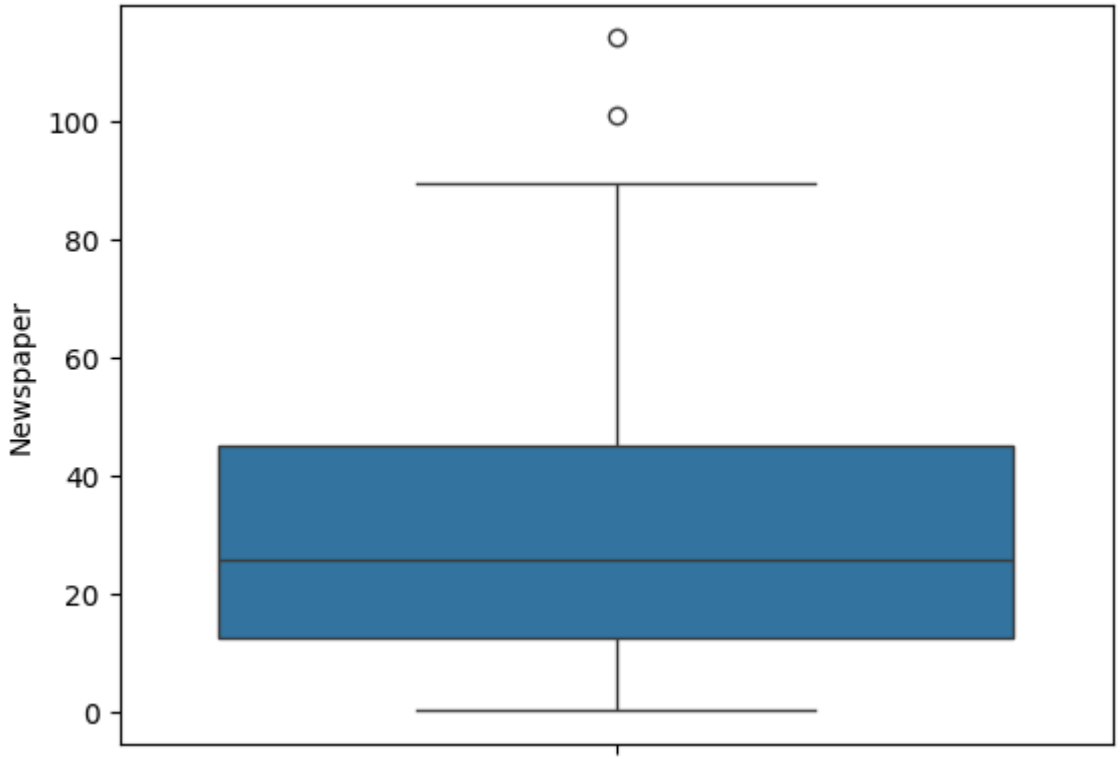
In [12]:
```
df.isnull().sum()*100
```

Out[12]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

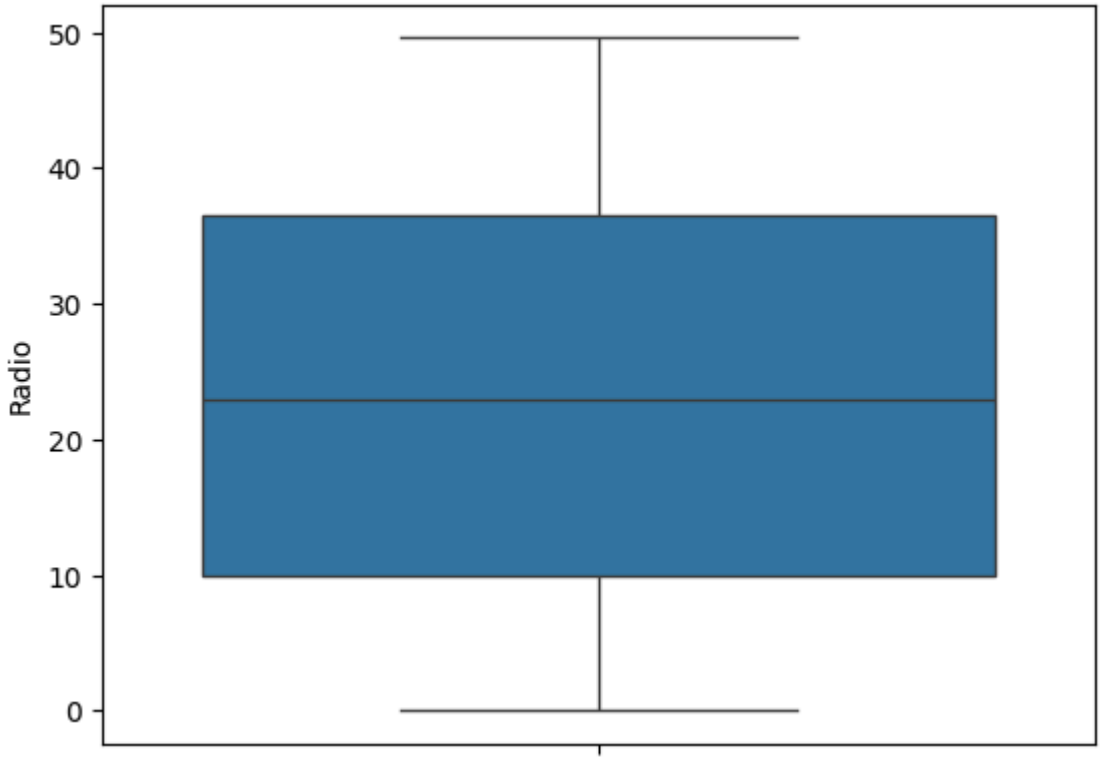
In [15]:
```
plt1 = sns.boxplot(df['TV'])
```



In [17]:
```
plt2 = sns.boxplot(df['Newspaper'])
```

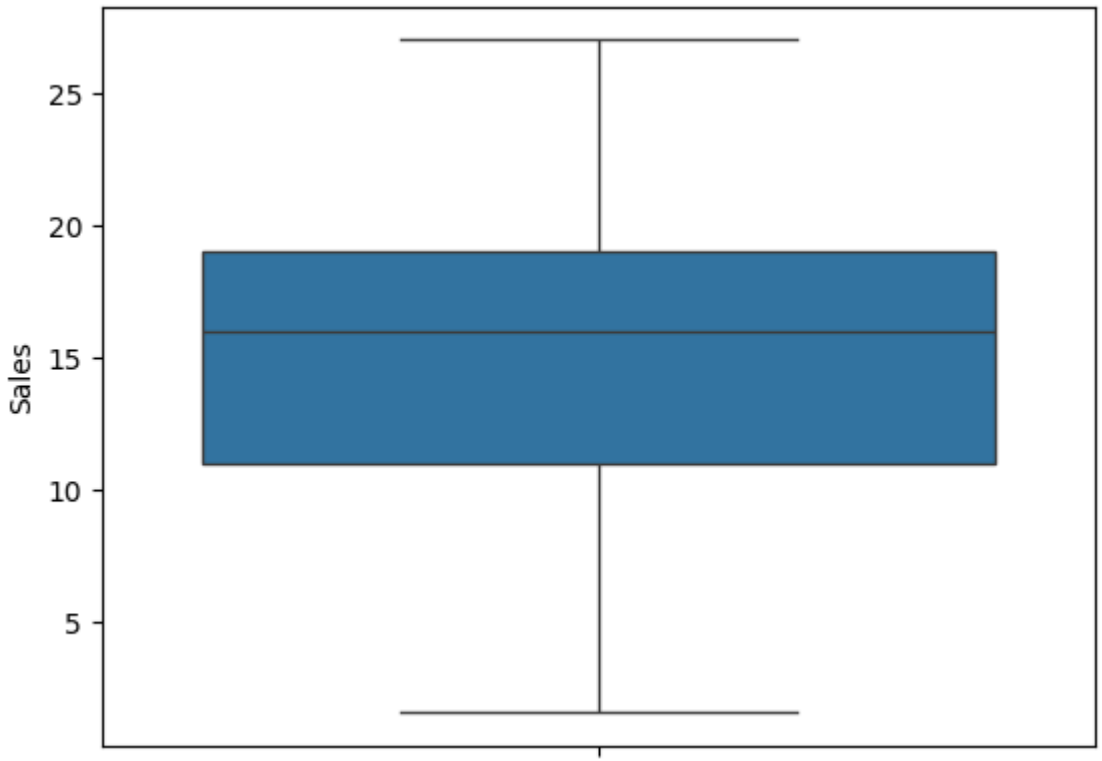


In [18]:
```
plt3 = sns.boxplot(df['Radio'])
```

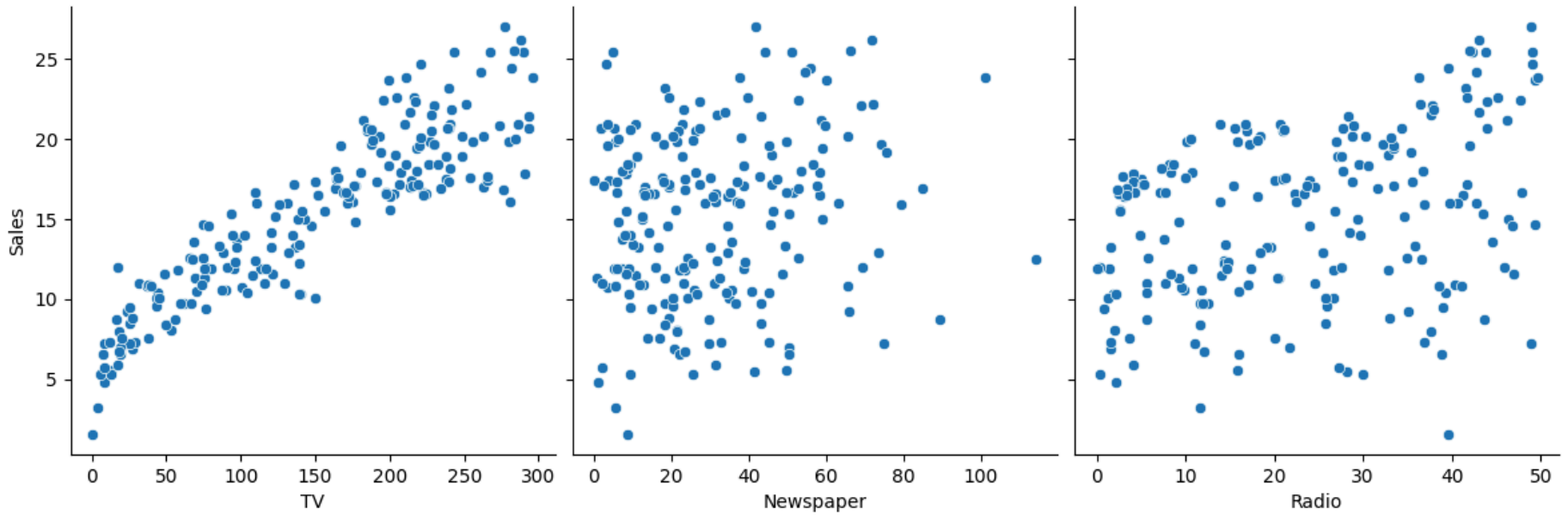


In [19]:
```
#  Data Analysis¶
```

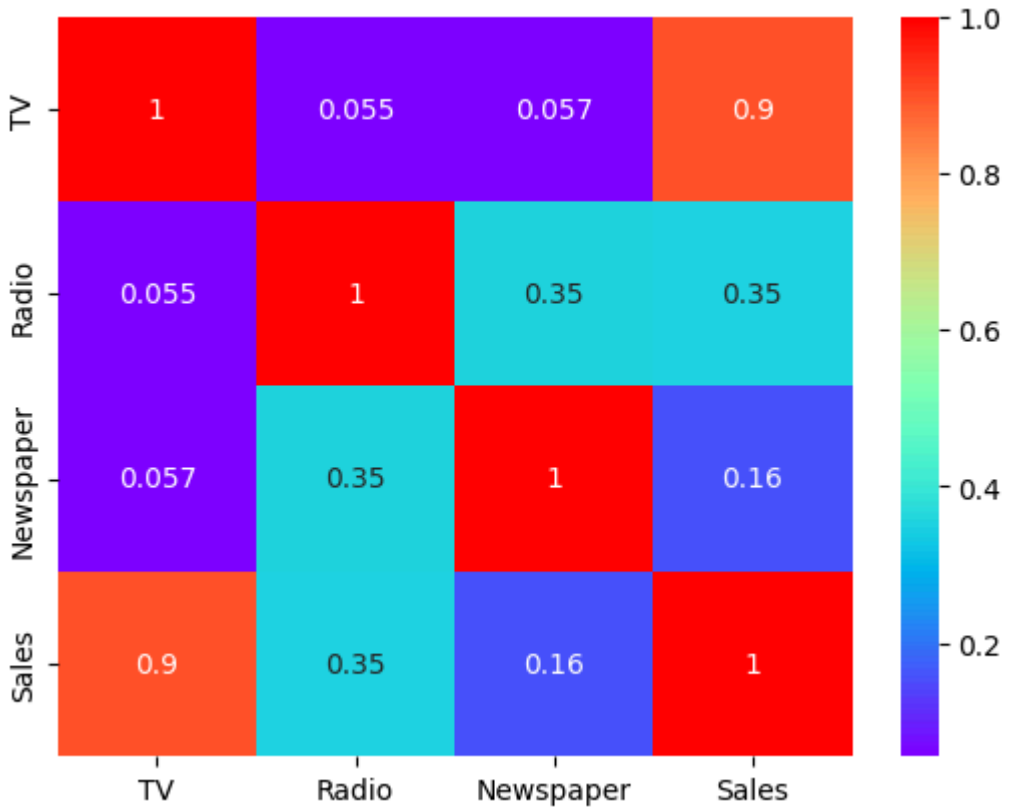
In [20]:
```
sns.boxplot(df['Sales'])
plt.show()
```



In [21]:
```
sns.pairplot(df, x_vars=['TV', 'Newspaper', 'Radio'], y_vars='Sales', height=4, aspect=1, kind='scatter')
plt.show()
```



In [23]:
```
sns.heatmap(df.corr(), cmap="rainbow", annot = True)
plt.show()
```



In [24]:
```
X = df['TV']
y = df['Sales']
```

In [46]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size = 0.3, random_state = 100)
```

In [47]:
```
X_train.head()
```

Out[47]:
```
74     213.4
3      151.5
185    205.0
26     142.9
90     134.3
Name: TV, dtype: float64
```

In [48]:
```
X_test.head()
```

Out[48]:
```
126      7.8
104    238.2
99     135.2
92     217.7
111    241.7
Name: TV, dtype: float64
```

In [49]:
```
 y_test.head()
```

Out[49]:
```
126     6.6
104    20.7
99     17.2
92     19.4
111    21.8
Name: Sales, dtype: float64
```

In [50]:
```
 y_train.head()
```

Out[50]:
```
74     17.0
3      16.5
185    22.6
26     15.0
90     14.0
Name: Sales, dtype: float64
```

In [51]:
```
import statsmodels.api as sm
```

In [52]:
```
X_train_sm = sm.add_constant(X_train)
```

In [53]:
```
lr = sm.OLS(y_train, X_train_sm).fit()
```

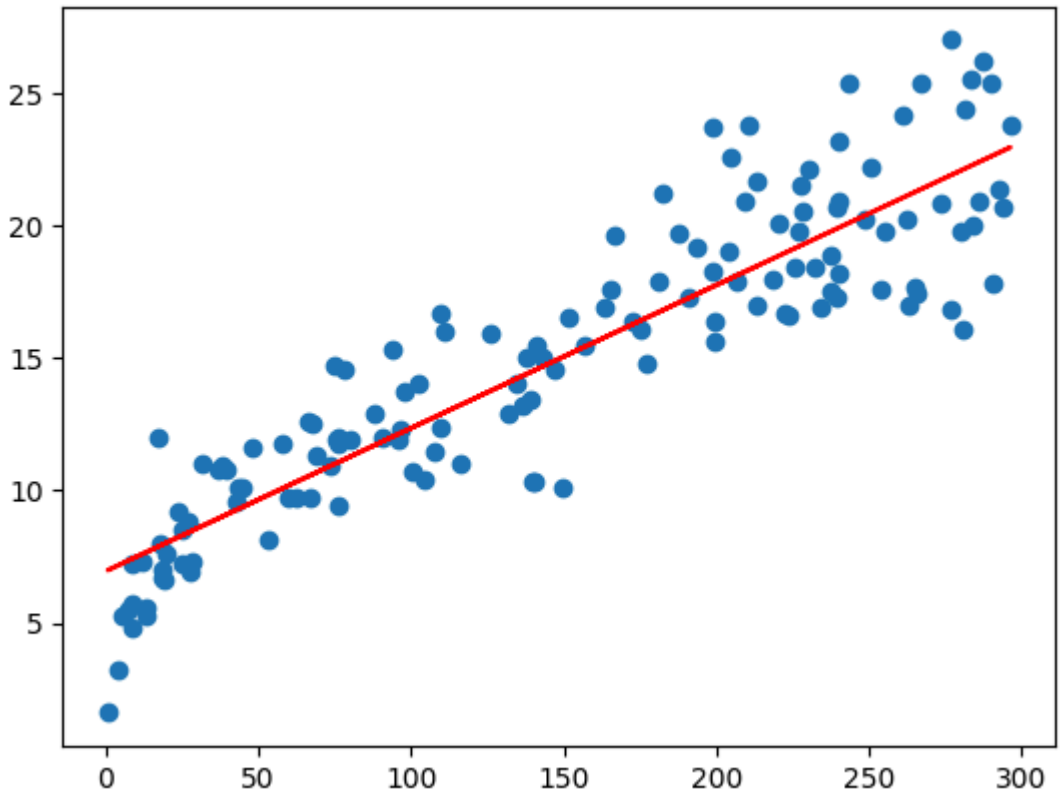
In [54]:
```
lr.params
```

Out[54]:
```
const    6.948683
TV       0.054546
dtype: float64
```

In [55]:
```
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.816
Model:                            OLS   Adj. R-squared:                  0.814
Method:                 Least Squares   F-statistic:                     611.2
Date:                Fri, 16 Aug 2024   Prob (F-statistic):           1.52e-52
Time:                        17:38:50   Log-Likelihood:                -321.12
No. Observations:                 140   AIC:                             646.2
Df Residuals:                     138   BIC:                             652.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.9487      0.385     18.068      0.000       6.188       7.709
TV             0.0545      0.002     24.722      0.000       0.050       0.059
==============================================================================
Omnibus:                        0.027   Durbin-Watson:                   2.196
Prob(Omnibus):                  0.987   Jarque-Bera (JB):                0.150
Skew:                          -0.006   Prob(JB):                        0.928
Kurtosis:                       2.840   Cond. No.                         328.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

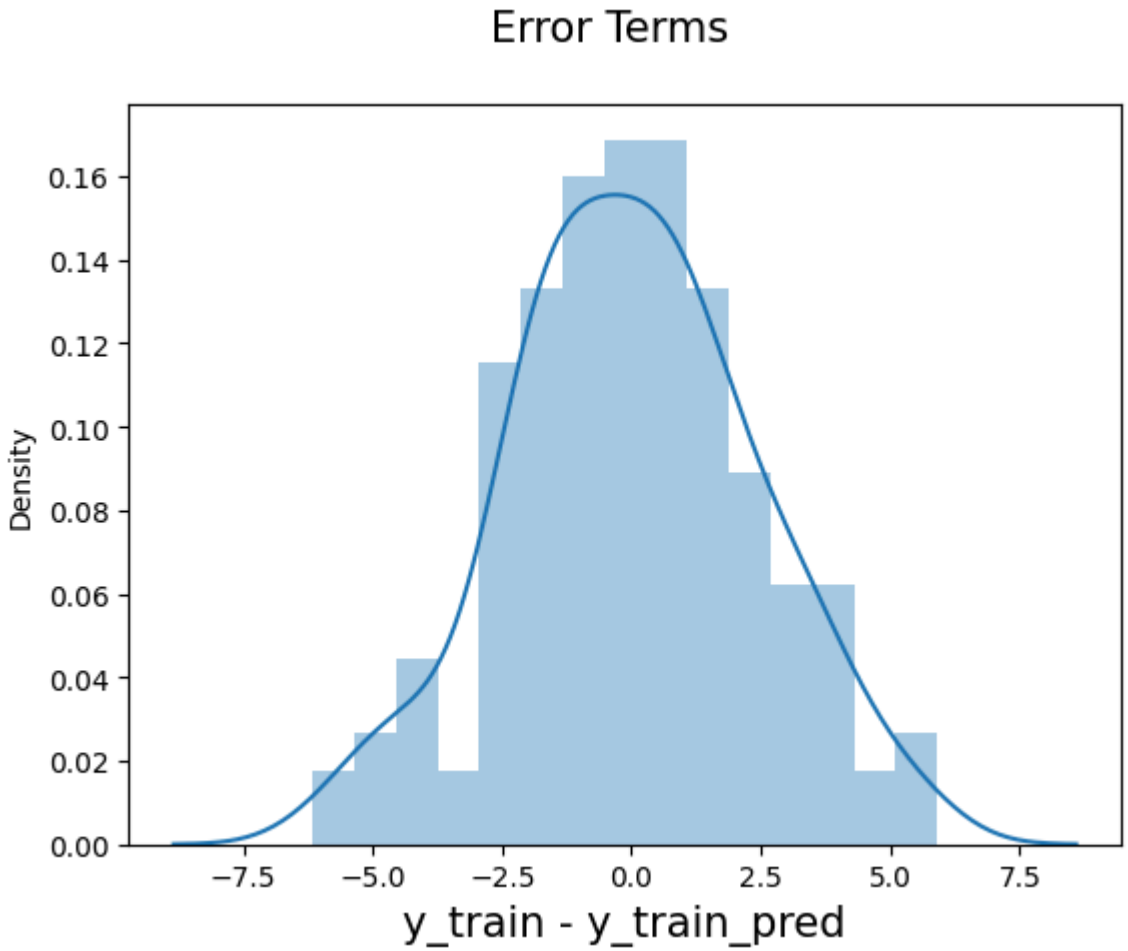
In [56]:
```
plt.scatter(X_train, y_train)
plt.plot(X_train, 6.948 + 0.054*X_train, 'r')
plt.show()
```



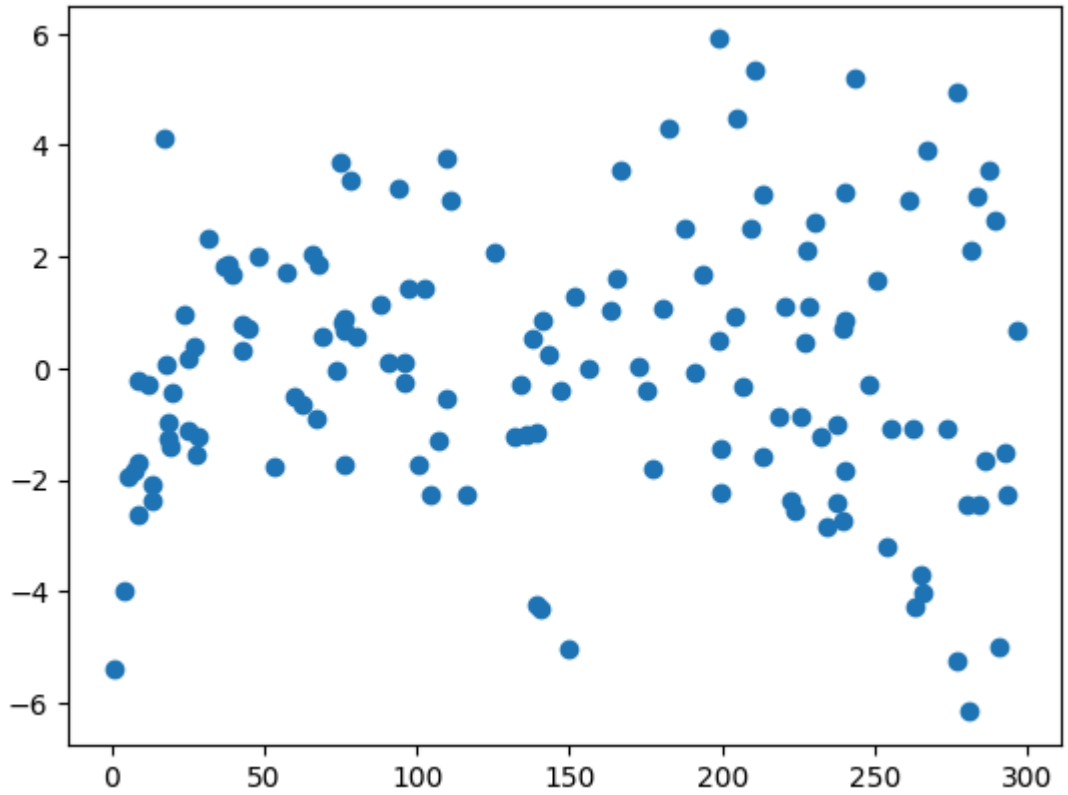
In [57]:
```
y_train_pred = lr.predict(X_train_sm)
res = (y_train - y_train_pred)
```

In [58]:
```
fig = plt.figure()
sns.distplot(res, bins = 15)
fig.suptitle('Error Terms', fontsize = 15)                  # Plot heading
plt.xlabel('y_train - y_train_pred', fontsize = 15)         # X-label
plt.show()
```

```
C:\Users\rushi\AppData\Local\Temp\ipykernel_7924\3003513444.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(res, bins = 15)
```



In [59]:
```
plt.scatter(X_train,res)
plt.show()
```



In [60]:
```
X_test_sm = sm.add_constant(X_test)
y_pred = lr.predict(X_test_sm)
```

In [61]:
```
y_pred.head()
```

Out[61]:
```
126     7.374140
104    19.941482
99     14.323269
92     18.823294
111    20.132392
dtype: float64
```

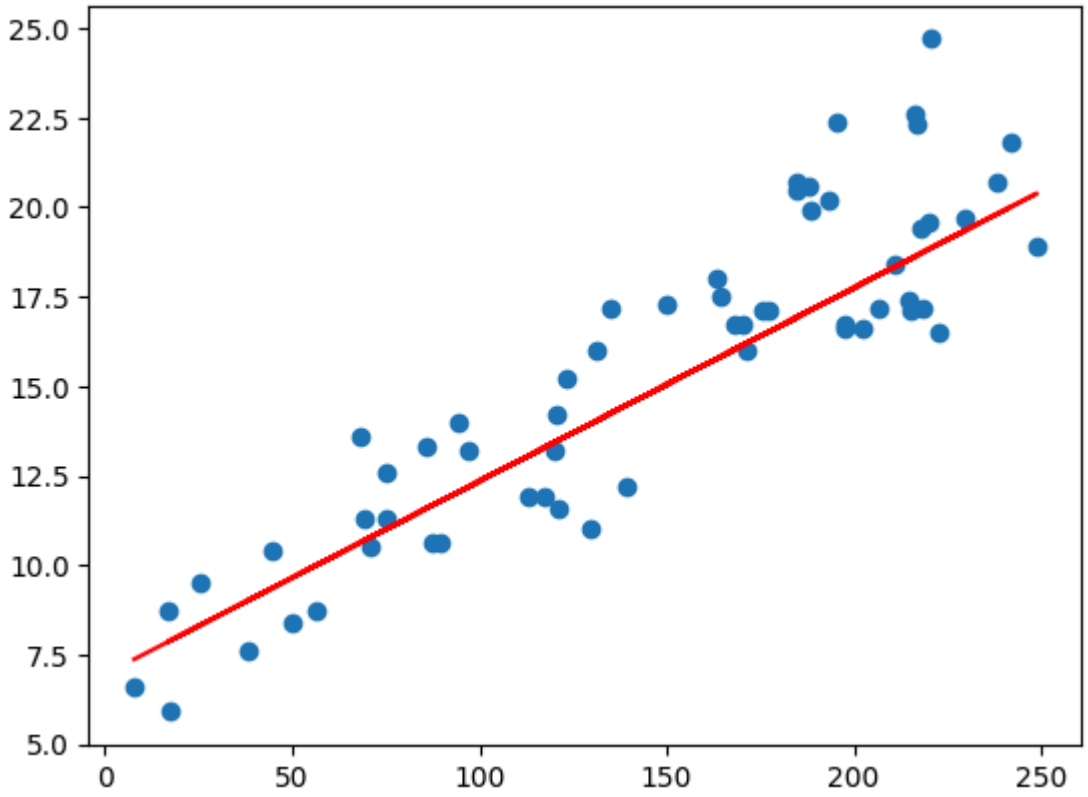
In [62]:
```
np.sqrt(mean_squared_error(y_test, y_pred))
```

Out[62]:
```
2.019296008966232
```

In [63]:
```
r_squared = r2_score(y_test, y_pred)
r_squared
```

Out[63]:
```
0.792103160124566
```

In [64]:
```
plt.scatter(X_test, y_test)
plt.plot(X_test, 6.948 + 0.054 * X_test, 'r')
plt.show()
```



In [ ]: