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

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("Downloads/advertising.csv")
```

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

Out[3]:

	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 × 4 columns

In [4]:
```
df.shape
```

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

In [5]:
```
df.describe()
```

Out[5]:

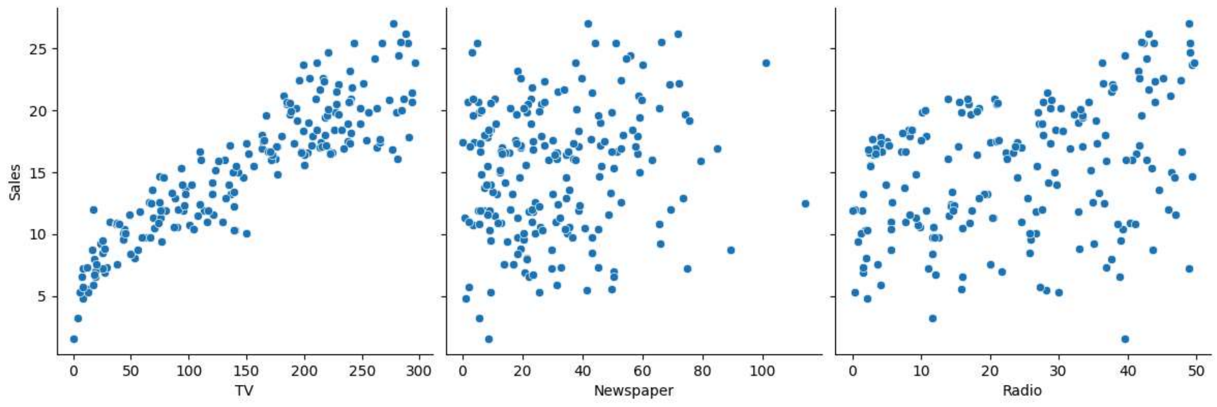
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000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

In [6]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [7]:
```
df.isnull().sum()*100
```

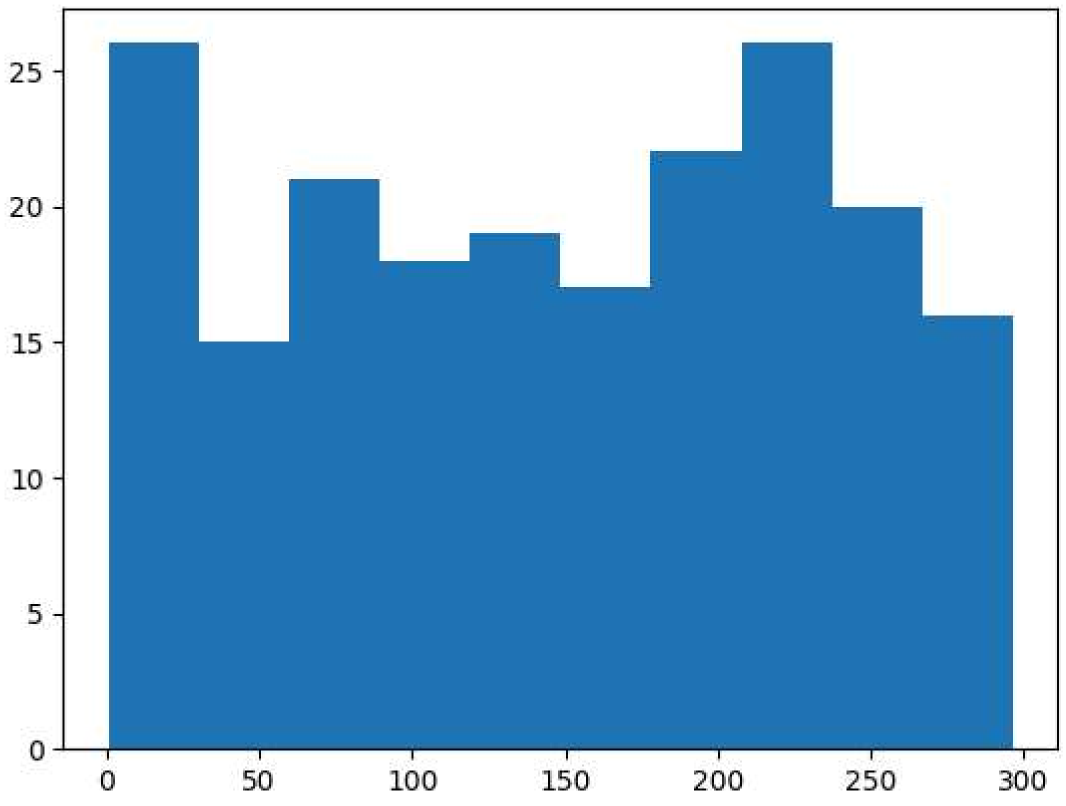
Out[7]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [8]:
```
sns.pairplot(df, x_vars=['TV', 'Newspaper', 'Radio'], y_vars='Sales', height=4, asp
plt.show()
```



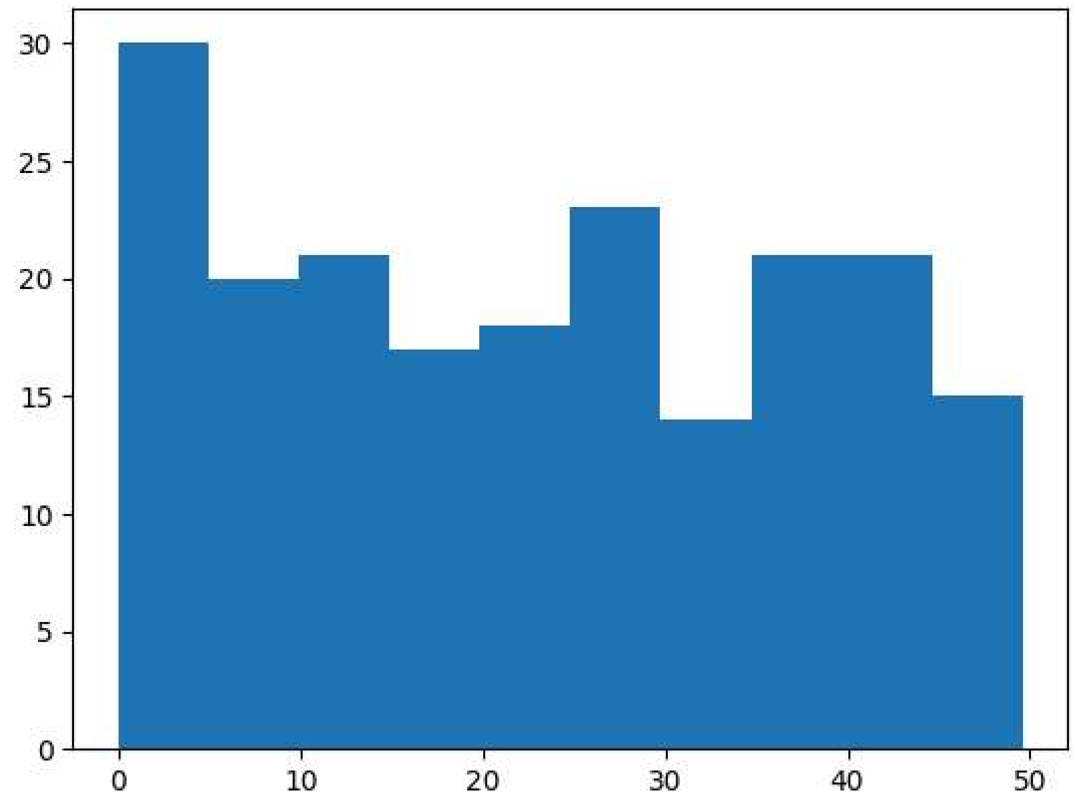
In [9]:
```
df['TV'].plot.hist(bina=10)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[9], line 1
----> 1 df['TV'].plot.hist(bina=10)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_co
re.py:1409, in PlotAccessor.hist(self, by, bins, **kwargs)
   1349 def hist(
   1350     self, by: IndexLabel | None = None, bins: int = 10, **kwargs
   1351 ) -> PlotAccessor:
   1352     """
   1353     Draw one histogram of the DataFrame's columns.
   1354
   (...)
   1407         >>> ax = df.plot.hist(column=["age"], by="gender", figsize=(10, 8))
   1408     """
-> 1409     return self(kind="hist", by=by, bins=bins, **kwargs)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_co
re.py:1030, in PlotAccessor.__call__(self, *args, **kwargs)
   1027             label_name = label_kw or data.columns
   1028             data.columns = label_name
-> 1030 return plot_backend.plot(data, kind=kind, **kwargs)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\__init__.py:71, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\core.py:501, in MPLPlot.generate(self)
    499 self._compute_plot_data()
    500 fig = self.fig
--> 501 self._make_plot(fig)
    502 self._add_table()
    503 self._make_legend()

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\hist.py:168, in HistPlot._make_plot(self, fig)
    164     kwds["weights"] = type(self)._get_column_weights(self.weights, i, y)
    166 y = reformat_hist_y_given_by(y, self.by)
--> 168 artists = self._plot(ax, y, column_num=i, stacking_id=stacking_id, **kwds)
    170 # when by is applied, show title for subplots to know which group it is
    171 if self.by is not None:

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\hist.py:122, in HistPlot._plot(cls, ax, y, style, bottom, column_num, stack
ing_id, bins, **kwds)
    120 bottom = bottom + cls._get_stacked_values(ax, stacking_id, base, kwds["labe
l"])
    121 # ignore style
--> 122 n, bins, patches = ax.hist(y, bins=bins, bottom=bottom, **kwds)
    123 cls._update_stacker(ax, stacking_id, n)
    124 return patches

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\__init_
_.py:1478, in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1475 @functools.wraps(func)
   1476 def inner(ax, *args, data=None, **kwargs):
   1477     if data is None:
-> 1478         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1480     bound = new_sig.bind(ax, *args, **kwargs)
   1481     auto_label = (bound.arguments.get(label_namer)
   1482                   or bound.kwargs.get(label_namer))

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\axes\_ax
es.py:7015, in Axes.hist(self, x, bins, range, density, weights, cumulative, bottom,
histtype, align, orientation, rwidth, log, color, label, stacked, **kwargs)
   7013 if patch:
   7014     p = patch[0]
-> 7015     p._internal_update(kwargs)
   7016     if lbl is not None:
   7017         p.set_label(lbl)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\artist.p
y:1219, in Artist._internal_update(self, kwargs)
   1212 def _internal_update(self, kwargs):
   1213     """
   1214     Update artist properties without prenormalizing them, but generating
   1215     errors as if calling `set`.
   1216 
   1217     The lack of prenormalization is to maintain backcompatibility.
   1218     """
-> 1219     return self._update_props(
   1220         kwargs, "{cls.__name__}.set() got an unexpected keyword argument "
   1221         "{prop_name!r}")

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\artist.p
y:1193, in Artist._update_props(self, props, errfmt)
   1191             func = getattr(self, f"set_{k}", None)
   1192             if not callable(func):
-> 1193                 raise AttributeError(
   1194                     errfmt.format(cls=type(self), prop_name=k))
   1195             ret.append(func(v))
   1196 if ret:

AttributeError: Rectangle.set() got an unexpected keyword argument 'bina'
```

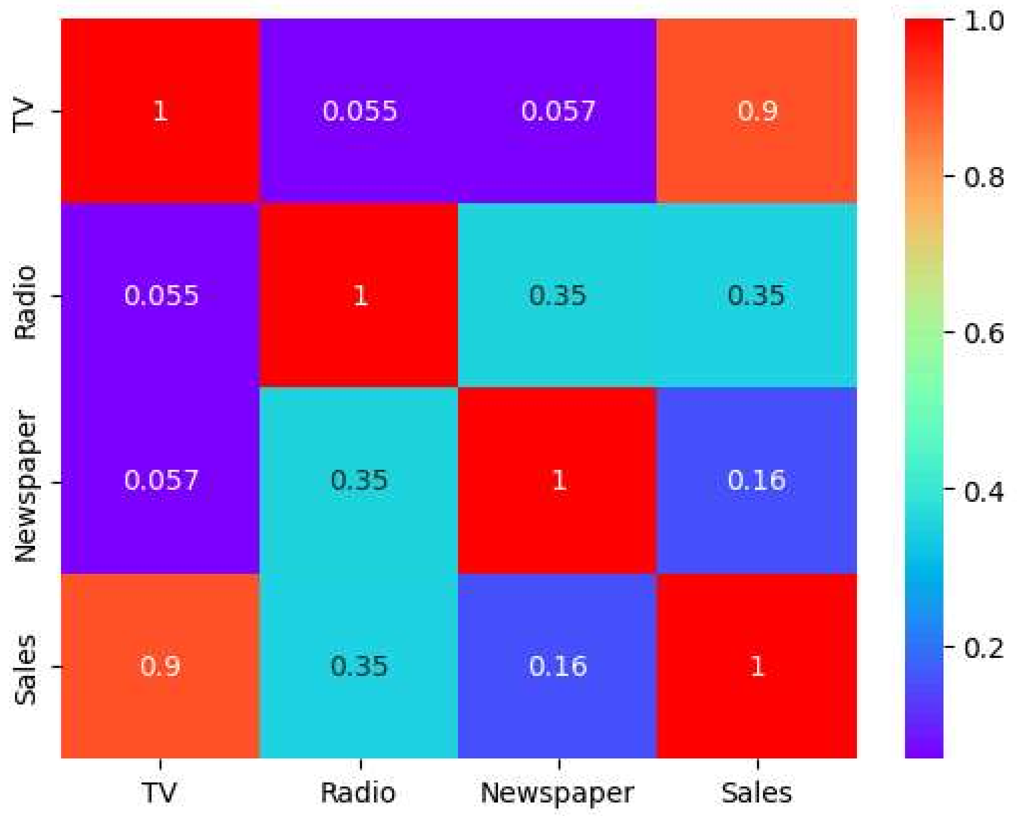


In [10]:
```
df['Radio'].plot.hist(bina=10)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[10], line 1
----> 1 df['Radio'].plot.hist(bina=10)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_co
re.py:1409, in PlotAccessor.hist(self, by, bins, **kwargs)
   1349 def hist(
   1350     self, by: IndexLabel | None = None, bins: int = 10, **kwargs
   1351 ) -> PlotAccessor:
   1352     """
   1353     Draw one histogram of the DataFrame's columns.
   1354
   (...)
   1407         >>> ax = df.plot.hist(column=["age"], by="gender", figsize=(10, 8))
   1408     """
-> 1409     return self(kind="hist", by=by, bins=bins, **kwargs)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_co
re.py:1030, in PlotAccessor.__call__(self, *args, **kwargs)
   1027             label_name = label_kw or data.columns
   1028             data.columns = label_name
-> 1030 return plot_backend.plot(data, kind=kind, **kwargs)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\__init__.py:71, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\core.py:501, in MPLPlot.generate(self)
    499 self._compute_plot_data()
    500 fig = self.fig
--> 501 self._make_plot(fig)
    502 self._add_table()
    503 self._make_legend()

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\hist.py:168, in HistPlot._make_plot(self, fig)
    164     kwds["weights"] = type(self)._get_column_weights(self.weights, i, y)
    166 y = reformat_hist_y_given_by(y, self.by)
--> 168 artists = self._plot(ax, y, column_num=i, stacking_id=stacking_id, **kwds)
    170 # when by is applied, show title for subplots to know which group it is
    171 if self.by is not None:

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\plotting\_ma
tplotlib\hist.py:122, in HistPlot._plot(cls, ax, y, style, bottom, column_num, stack
ing_id, bins, **kwds)
    120 bottom = bottom + cls._get_stacked_values(ax, stacking_id, base, kwds["labe
l"])
    121 # ignore style
--> 122 n, bins, patches = ax.hist(y, bins=bins, bottom=bottom, **kwds)
    123 cls._update_stacker(ax, stacking_id, n)
    124 return patches

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\__init_
_.py:1478, in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1475 @functools.wraps(func)
   1476 def inner(ax, *args, data=None, **kwargs):
   1477     if data is None:
-> 1478         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1480     bound = new_sig.bind(ax, *args, **kwargs)
   1481     auto_label = (bound.arguments.get(label_namer)
   1482                   or bound.kwargs.get(label_namer))

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\axes\_ax
es.py:7015, in Axes.hist(self, x, bins, range, density, weights, cumulative, bottom,
histtype, align, orientation, rwidth, log, color, label, stacked, **kwargs)
   7013 if patch:
   7014     p = patch[0]
-> 7015     p._internal_update(kwargs)
   7016     if lbl is not None:
   7017         p.set_label(lbl)

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\artist.p
y:1219, in Artist._internal_update(self, kwargs)
   1212 def _internal_update(self, kwargs):
   1213     """
   1214     Update artist properties without prenormalizing them, but generating
   1215     errors as if calling `set`.
   1216 
   1217     The lack of prenormalization is to maintain backcompatibility.
   1218     """
-> 1219     return self._update_props(
   1220         kwargs, "{cls.__name__}.set() got an unexpected keyword argument "
   1221         "{prop_name!r}")

File ~\AppData\Local\Programs\Python\Python312\Lib\site-packages\matplotlib\artist.p
y:1193, in Artist._update_props(self, props, errfmt)
   1191             func = getattr(self, f"set_{k}", None)
   1192             if not callable(func):
-> 1193                 raise AttributeError(
   1194                     errfmt.format(cls=type(self), prop_name=k))
   1195             ret.append(func(v))
   1196 if ret:

AttributeError: Rectangle.set() got an unexpected keyword argument 'bina'
```



```
In [11]: sns.heatmap(df.corr(), cmap="rainbow", annot = True)
         plt.show()
```



In [12]:
```
X = df['TV']
y = df['Sales']
```

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_si
```

In [15]:
```
y_train.info()
```

```
<class 'pandas.core.series.Series'>
Index: 140 entries, 74 to 8
Series name: Sales
Non-Null Count  Dtype
--------------  -----
140 non-null    float64
dtypes: float64(1)
memory usage: 2.2 KB
```

In [16]:
```
y_train.describe()
```

Out[16]:
```
count    140.000000
mean      15.005714
std        5.608264
min        1.600000
25%       10.800000
50%       15.500000
75%       19.300000
max       27.000000
Name: Sales, dtype: float64
```

In [17]:
```
X_train.head()
```

Out[17]:
```
74     213.4
3      151.5
185    205.0
26     142.9
90     134.3
Name: TV, dtype: float64
```

In [18]:
```
X_test.head()
```

Out[18]:
```
126      7.8
104    238.2
99     135.2
92     217.7
111    241.7
Name: TV, dtype: float64
```

In [19]:
```
 y_train.head()
```

```
Out[19]: 74     17.0
         3      16.5
         185    22.6
         26     15.0
         90     14.0
         Name: Sales, dtype: float64
```

```
In [20]:  y_test.head()
```

```
Out[20]: 126     6.6
         104    20.7
         99     17.2
         92     19.4
         111    21.8
         Name: Sales, dtype: float64
```

```
In [21]: import statsmodels.api as sm
```

```
In [22]: X_train_sm = sm.add_constant(X_train)
```

```
In [23]: lr = sm.OLS(y_train, X_train_sm).fit()
```

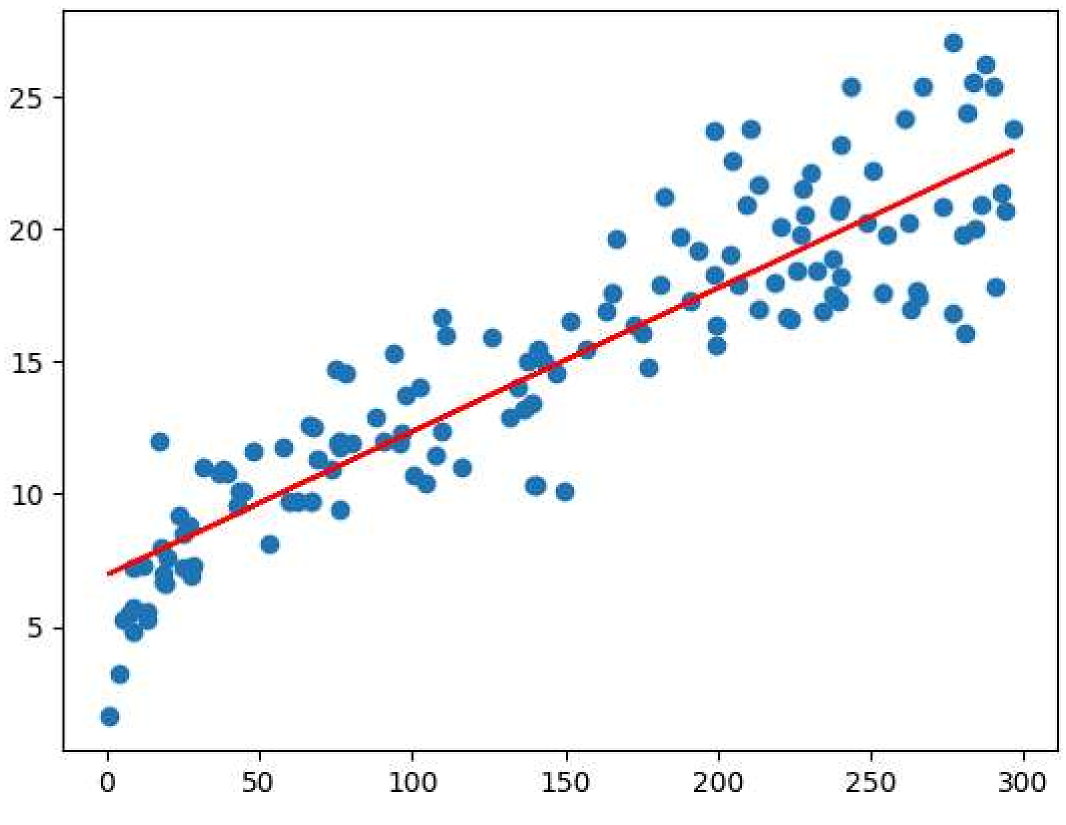
```
In [24]: lr.params
```

```
Out[24]: const    6.948683
         TV       0.054546
         dtype: float64
```

```
In [25]: 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:                Thu, 08 Aug 2024   Prob (F-statistic):           1.52e-52
Time:                        16:27:13   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 spe
cified.
```

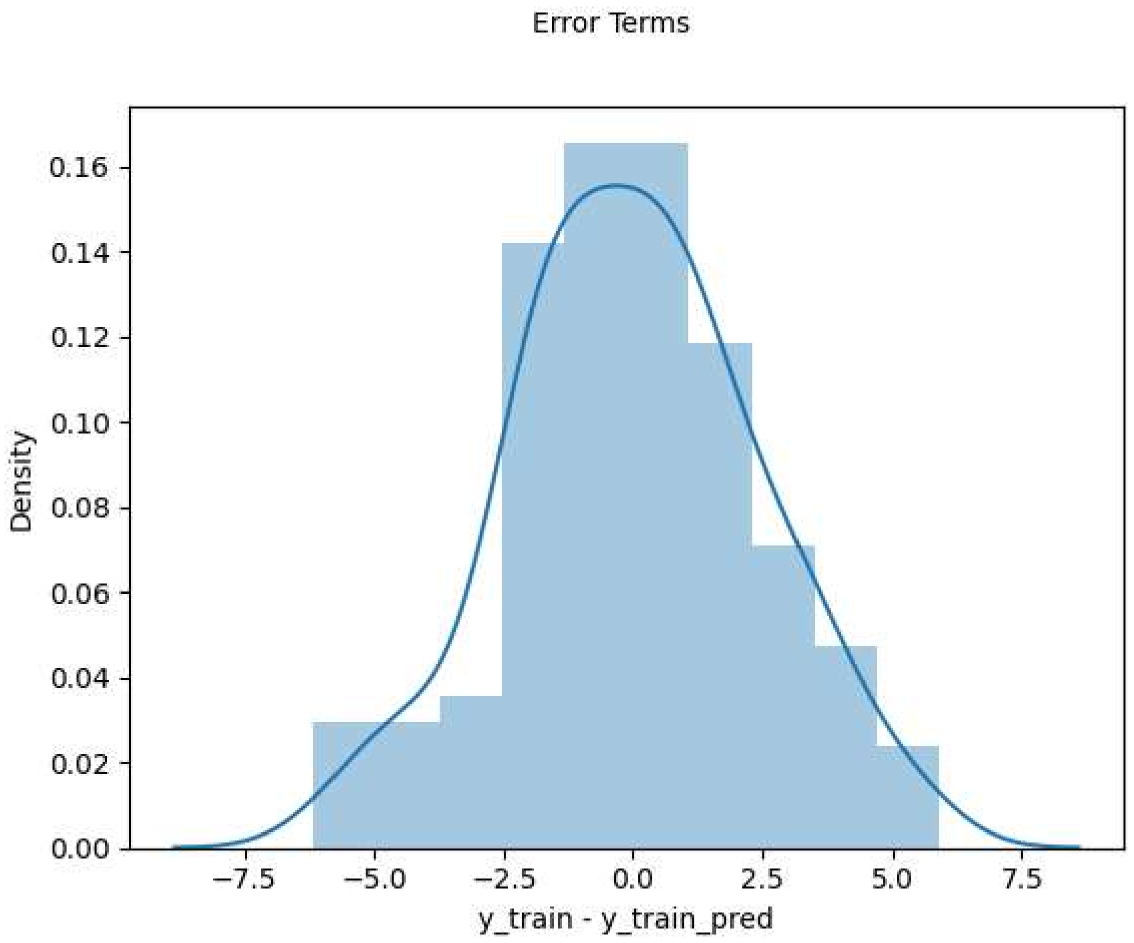
In [26]:
```python
plt.scatter(X_train, y_train)
plt.plot(X_train, 6.948 + 0.054*X_train, 'r')
plt.show()
```



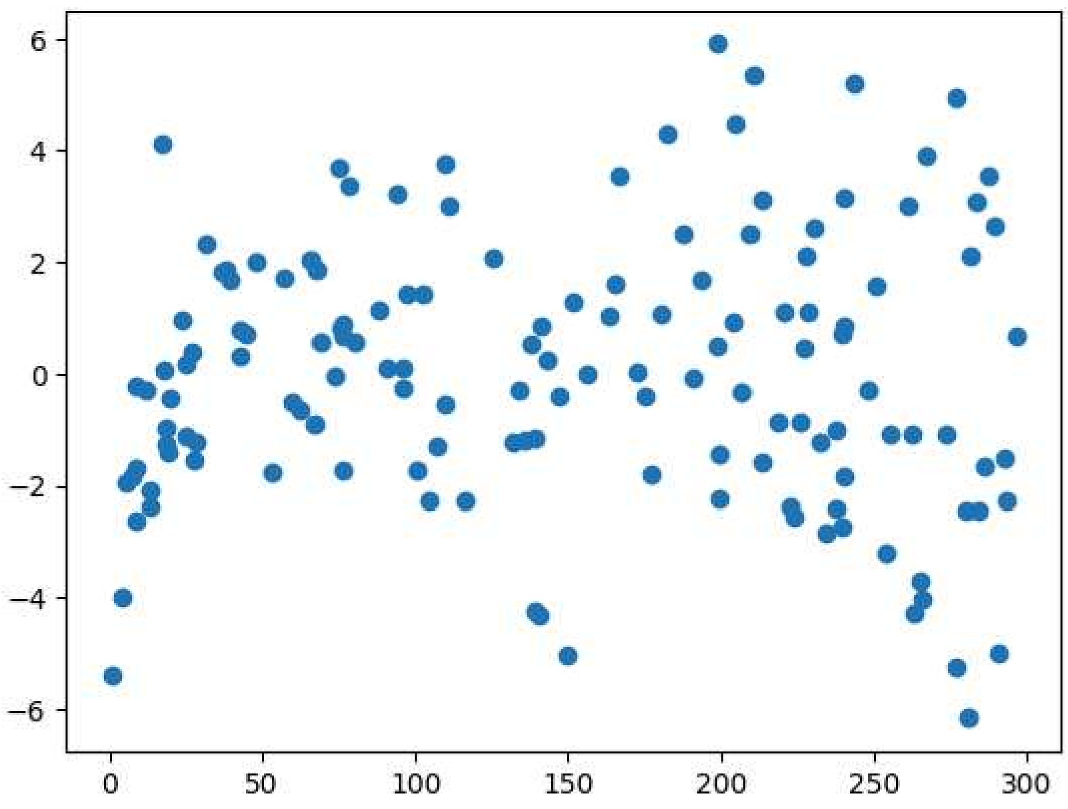
In [32]:
```
y_train_pred = lr.predict(X_train_sm)
res = (y_train - y_train_pred)
```

In [34]:
```
fig = plt.figure()
sns.distplot(res, bins = 10)
fig.suptitle('Error Terms', fontsize = 10)                  # P
plt.xlabel('y_train - y_train_pred', fontsize = 10)
plt.show()
```

```
C:\Users\rushi\AppData\Local\Temp\ipykernel_11376\931037861.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 = 10)
```



In [30]:
```
plt.scatter(X_train,res)
plt.show()
```



```
In [35]: X_test_sm = sm.add_constant(X_test)
         y_pred = lr.predict(X_test_sm)
```

```
In [36]: y_pred.head()
```

```
Out[36]: 126     7.374140
         104    19.941482
         99     14.323269
         92     18.823294
         111    20.132392
         dtype: float64
```

```
In [37]: from sklearn.metrics import mean_squared_error
         from sklearn.metrics import r2_score
```

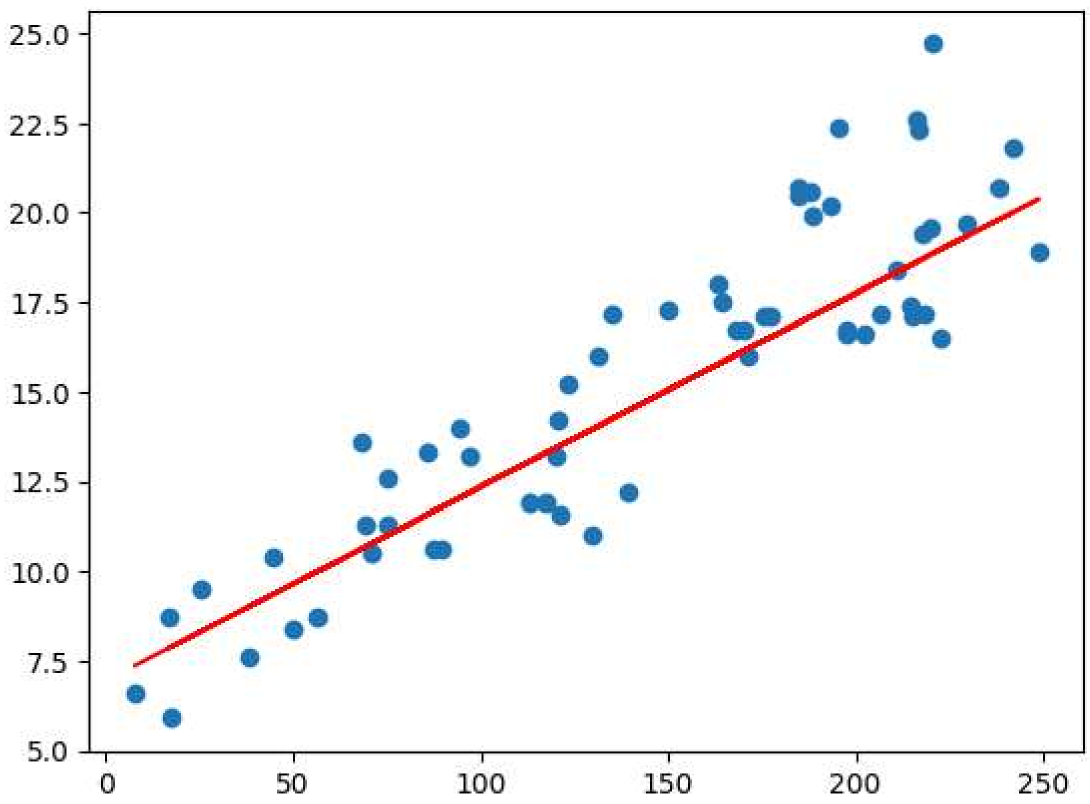
```
In [38]: np.sqrt(mean_squared_error(y_test, y_pred))
```

```
Out[38]: 2.019296008966232
```

```
In [39]: r_squared = r2_score(y_test, y_pred)
         r_squared
```

```
Out[39]: 0.792103160124566
```

```
In [40]: plt.scatter(X_test, y_test)
         plt.plot(X_test, 6.948 + 0.054 * X_test, 'r')
         plt.show()
```



```
In [43]:
```

```
In [ ]:
```