In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [2]:

```python
data_file=pd.read_csv("sales data file.csv")
```

In [4]:

```python
data_file.head(15)
```

Out[4]:

	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
5	8.7	48.9	75.0	7.2
6	57.5	32.8	23.5	11.8
7	120.2	19.6	11.6	13.2
8	8.6	2.1	1.0	4.8
9	199.8	2.6	21.2	15.6
10	66.1	5.8	24.2	12.6
11	214.7	24.0	4.0	17.4
12	23.8	35.1	65.9	9.2
13	97.5	7.6	7.2	13.7
14	204.1	32.9	46.0	19.0

In [5]:

```python
data_file.shape
```

Out[5]:

```
(200, 4)
```

In [13]:

```python
data_file.describe()
```

Out[13]:

	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

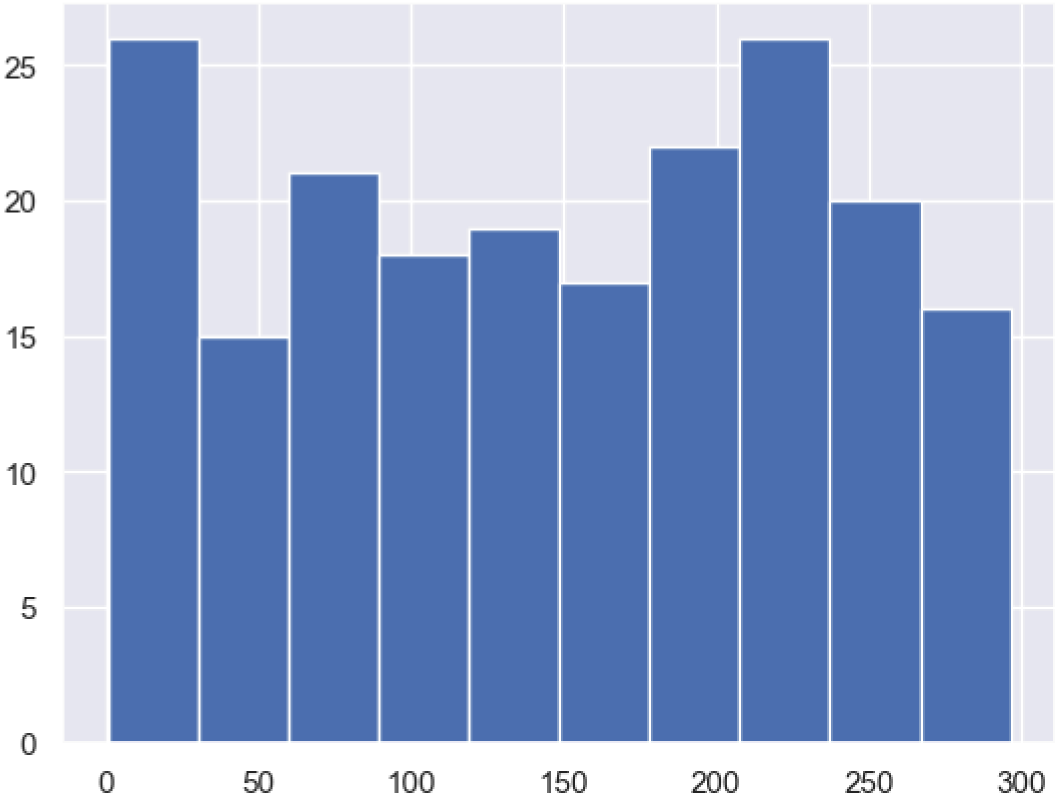
In [14]:
```
data_file.isnull().sum()
```

Out[14]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [15]:
```
sns.set()
```

In [37]:
```
data_file['TV'].hist()
```

Out[37]:
```
<AxesSubplot:>
```

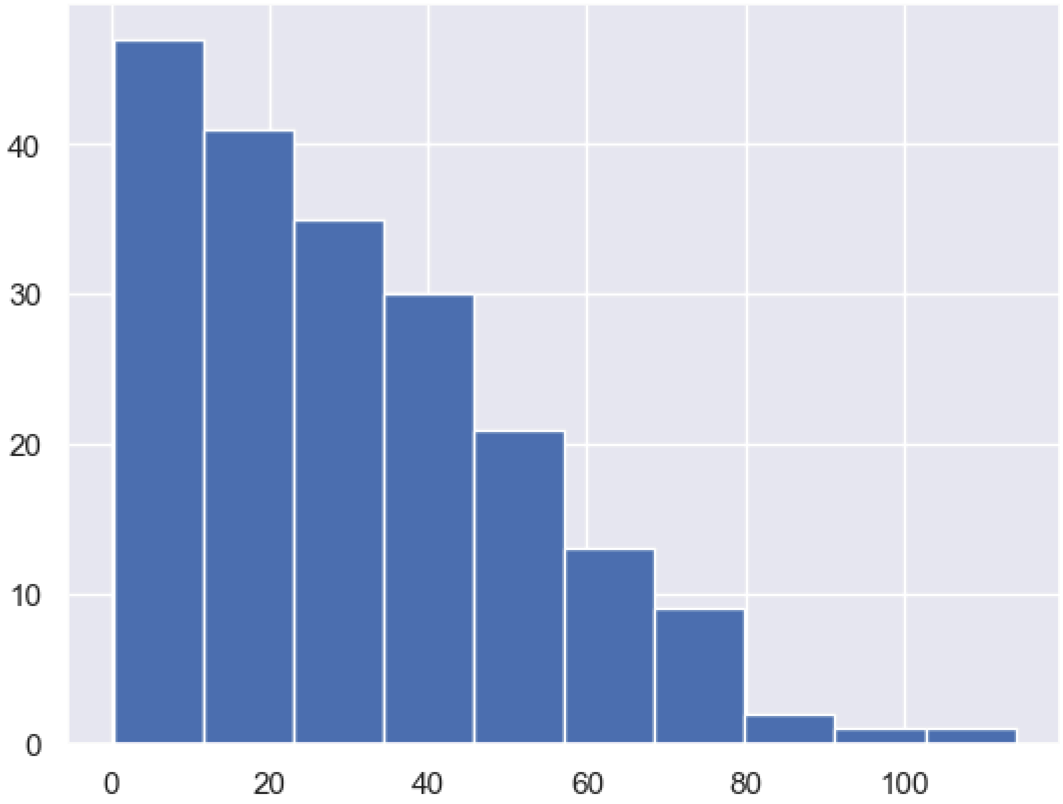


In [38]:

```
data_file['Newspaper'].hist()
```

Out[38]:

```
<AxesSubplot:>
```

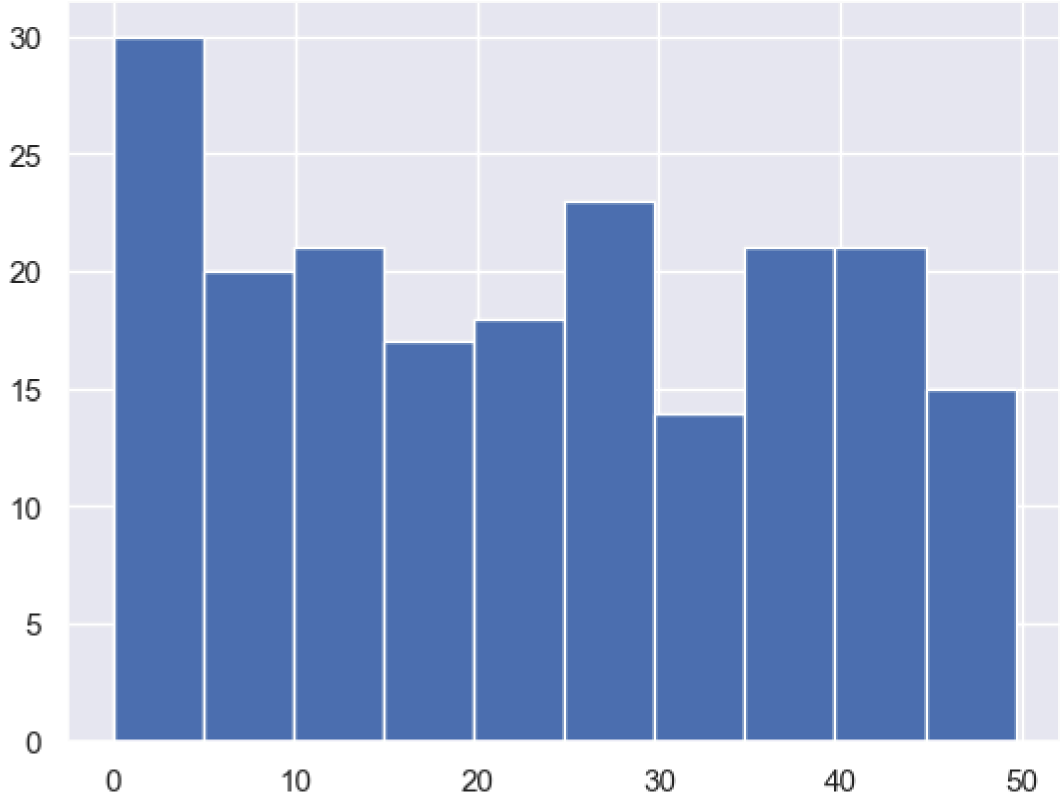


In [39]:

```
data_file['Radio'].hist()
```

Out[39]:

```
<AxesSubplot:>
```



```
In [50]:  X=data_file.drop(columns='Sales')
```

```
In [51]:  Y=data_file['Sales']
```

```
In [52]:  X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_
```

```
In [53]:  model=LinearRegression()
```

```
In [54]:  model.fit(X_train,Y_train)
```

```
Out[54]:  LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [55]:  prediction=model.predict(X_test)
```

```
In [56]:  prediction
```

```
Out[56]:  array([17.94221632, 11.28731032, 19.36406753, 15.25309499,  8.85035488,
                 11.08345095, 24.54827272, 10.72184726, 18.64190205, 17.03877174,
                 14.71887065, 13.30204368, 19.10529921, 11.4654086 , 13.82417942,
                 14.56139355, 16.86156735, 17.27369971, 17.78634747, 21.28201581,
                 19.1397699 , 11.05346066,  9.93276334, 11.49854807,  8.5309559 ,
                 13.26073545, 21.75566382, 16.96066432, 24.25791572, 11.92392893,
                 16.40376866, 21.96064207,  9.51770237, 10.16209996, 10.08141197,
                 10.45644324, 15.54919097,  9.92133897, 13.83425453, 12.54320065,
                 14.5093965 , 12.61758414,  6.46804914, 20.25656292, 23.16303373,
                 24.65508581, 15.20817964,  9.27513655, 18.72004324, 18.16217728,
                 12.73063894, 16.65175796, 15.79776032,  8.36188762, 21.22771856,
                  9.52094834, 23.88078008, 23.29062902, 19.6930198 , 16.7646752
          2])
```

```
In [57]:  model.intercept_
```

```
Out[57]:  5.022730805826264
```

```
In [58]:  model.coef_
```

```
Out[58]:  array([ 0.05223455,  0.10672463, -0.00120158])
```

```
In [59]:  accuracy_score=model.score(X_test,Y_test)*100
```

```
In [60]:  print(f"Accuracy of model: {accuracy_score}%")
```

```
          Accuracy of model: 88.77675297095178%
```