In [34]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [35]:
```
df=pd.read_csv("Iris.csv")
df
```

Out[35]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

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

Out[36]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [37]:
```
df.shape
```

Out[37]:
```
(150, 6)
```

In [38]:
```
df.isnull().sum()
```

Out[38]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [39]:
```
df.describe()
```

Out[39]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

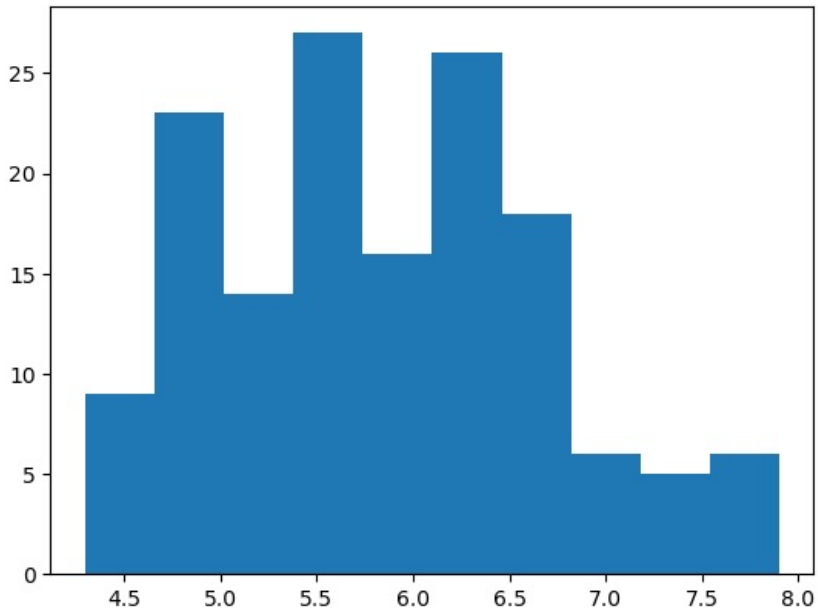
In [40]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

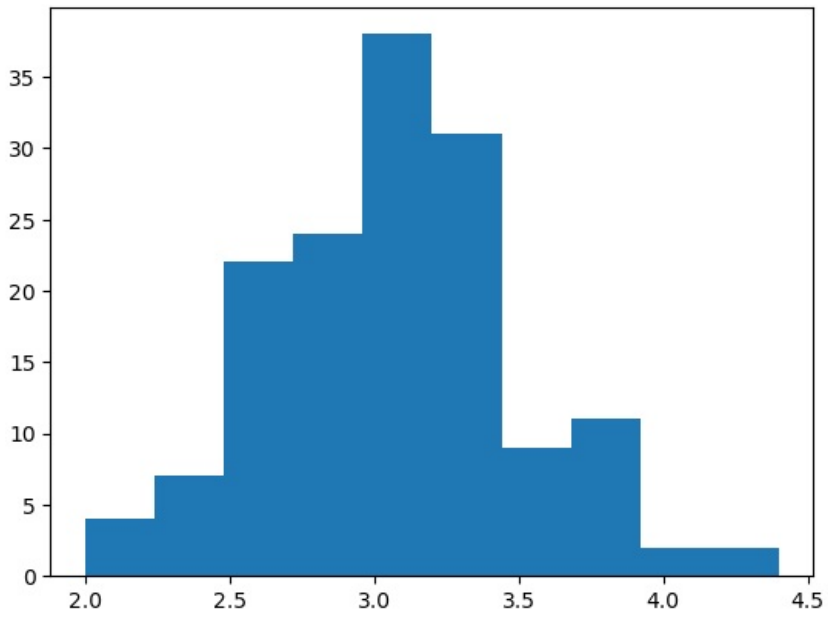
In [41]:
```
df["Species"].value_counts()
```

Out[41]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

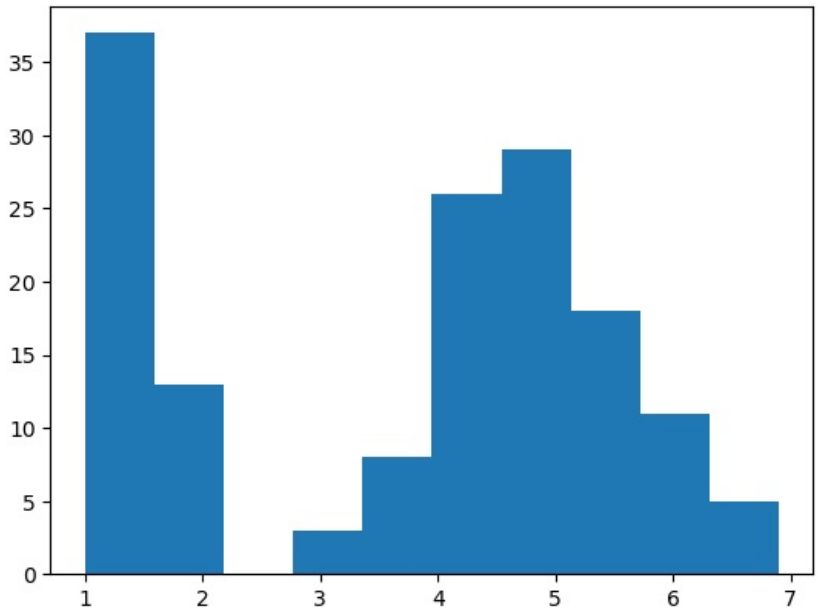
In [42]:
```
df["SepalLengthCm"].hist(grid=0)
plt.show()
```



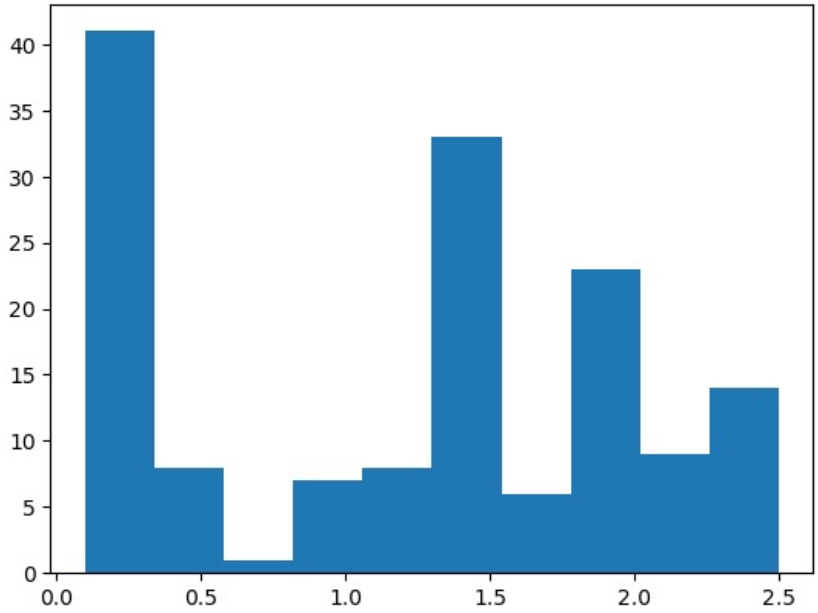
In [43]:
```
df["SepalWidthCm"].hist(grid=0)
plt.show()
```



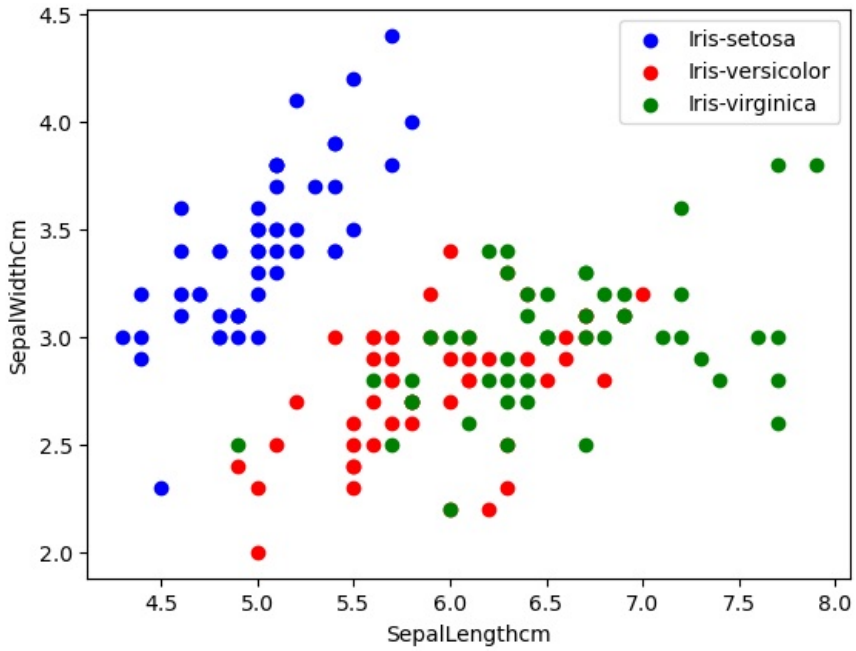
In [44]:
```
df["PetalLengthCm"].hist(grid=0)
plt.show()
```



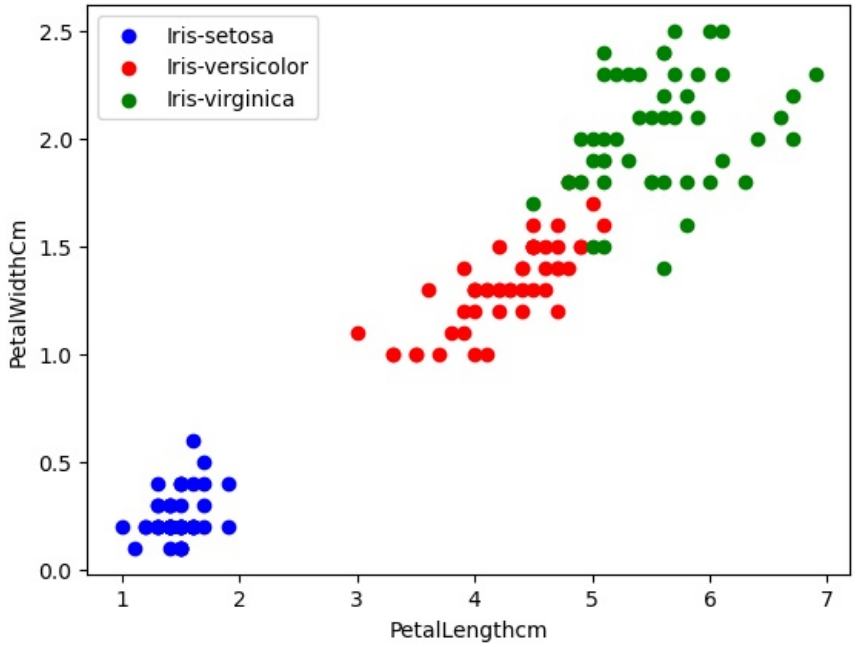
In [45]:
```
df["PetalWidthCm"].hist(grid=0)
plt.show()
```



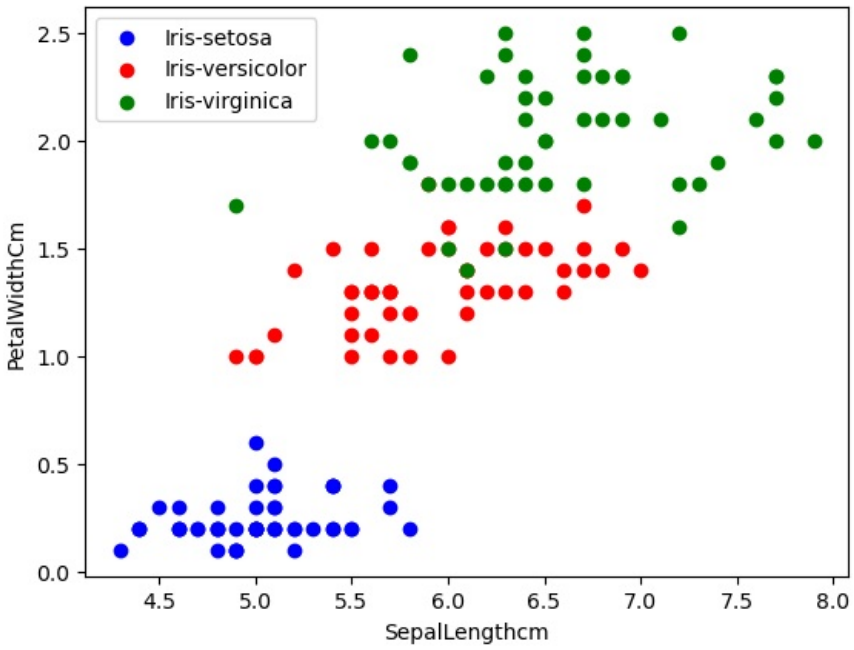
In [46]:
```
colors=['blue','red','green']
species=['Iris-setosa','Iris-versicolor','Iris-virginica']
for i in range(3):
    x=df[df["Species"]==species[i]]
    plt.scatter(x['SepalLengthCm'],x['SepalWidthCm'],c=colors[i],label=species[i])
plt.xlabel('SepalLengthcm')
plt.ylabel('SepalWidthCm')
plt.legend()
plt.show()
```



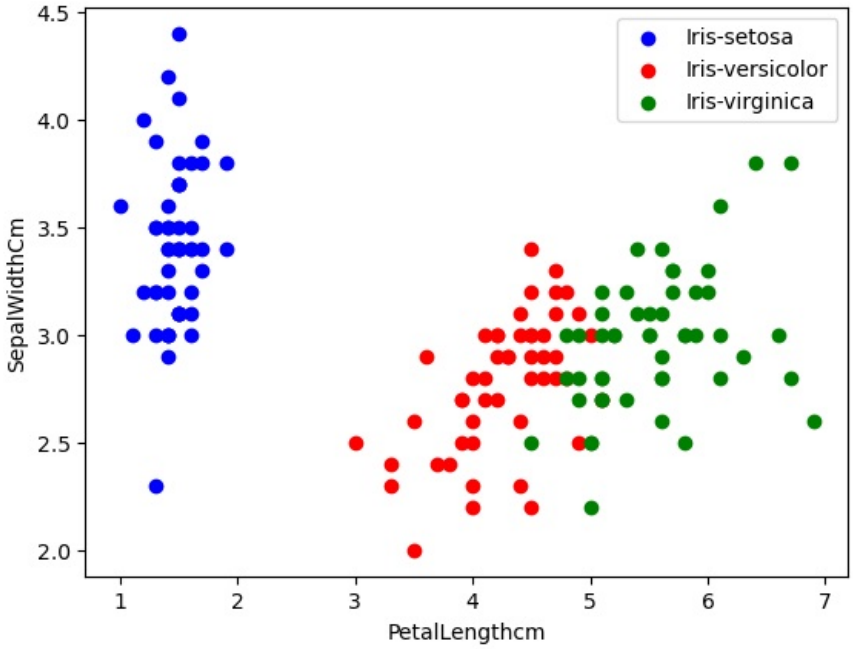
In [47]:
```
colors=['blue','red','green']
species=['Iris-setosa','Iris-versicolor','Iris-virginica']
for i in range(3):
    x=df[df["Species"]==species[i]]
    plt.scatter(x['PetalLengthCm'],x['PetalWidthCm'],c=colors[i],label=species[i])
plt.xlabel('PetalLengthcm')
plt.ylabel('PetalWidthCm')
plt.legend()
plt.show()
```



In [50]:
```
colors=['blue','red','green']
species=['Iris-setosa','Iris-versicolor','Iris-virginica']
for i in range(3):
    x=df[df["Species"]==species[i]]
    plt.scatter(x['SepalLengthCm'],x['PetalWidthCm'],c=colors[i],label=species[i])
plt.xlabel('SepalLengthcm')
plt.ylabel('PetalWidthCm')
plt.legend()
plt.show()
```



In [51]:
```
colors=['blue','red','green']
species=['Iris-setosa','Iris-versicolor','Iris-virginica']
for i in range(3):
    x=df[df["Species"]==species[i]]
    plt.scatter(x['PetalLengthCm'],x['SepalWidthCm'],c=colors[i],label=species[i])
plt.xlabel('PetalLengthcm')
plt.ylabel('SepalWidthCm')
plt.legend()
plt.show()
```



In [52]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [53]:
```
df['Species']=le.fit_transform(df['Species'])
df
```

Out[53]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	2
146	147	6.3	2.5	5.0	1.9	2
147	148	6.5	3.0	5.2	2.0	2
148	149	6.2	3.4	5.4	2.3	2
149	150	5.9	3.0	5.1	1.8	2

150 rows × 6 columns

In [54]:
```
from sklearn.model_selection import train_test_split
X=df.drop(columns=['Species'])
Y=df['Species']
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.30)
```

In [55]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [56]:
```
model.fit(x_train,y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs fai
led to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[56]:

▾ LogisticRegression

LogisticRegression()

In [57]:
```
model.score(x_test,y_test)*100
```

Out[57]:
```
100.0
```

In [ ]: