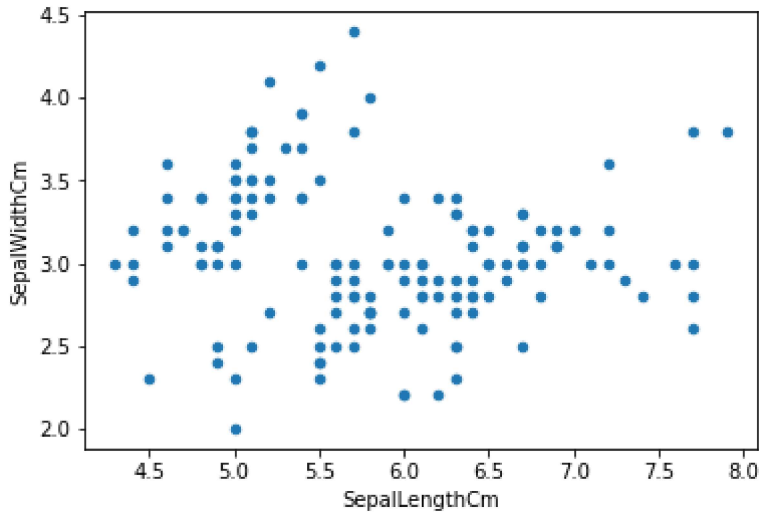
In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
iris = pd.read_csv("C:/Users/tyler/OneDrive/Desktop/Tyler stuff/Predictive Modeling/Iris.csv") #Iris.csv is now a pandas dataframe
print(iris.head()) #prints first 5 values
print(iris.describe()) #prints some basic statistical details like percentile, mean, std etc. of the data frame
```

```
   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
               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 [33]:
```
#visualizing data

iris.plot(kind="scatter", x="SepalLengthCm", y="SepalWidthCm")
plt.show()

#Preprocessing the dataset :
#Using an inbuilt library called 'train_test_split', which divides our data set into a ratio of 80:20. 80% will be used for training, #evaluating, and selection among our models and 20% will be held back as a validation dataset.

#Splitting the dataset into the Training set and Test set
```



In [4]:
```
from sklearn.model_selection import train_test_split
x = iris.iloc[:, :-1].values #last column values excluded
y = iris.iloc[:, -1].values #last column value
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_stat
e=0)
```

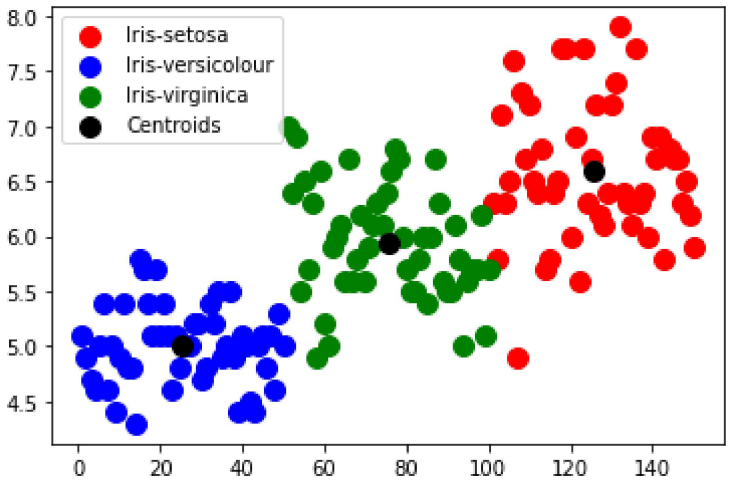
In [20]:
```
#Using decision tree on Iris dataset :

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_r
eport
classifier = DecisionTreeClassifier()
classifier.fit(x_train, y_train) #training the classifier
y_pred = classifier.predict(x_test) #making precdictions
```

In [22]:
```
print('accuracy is',accuracy_score(y_pred,y_test))
#Accuracy score
print(classification_report(y_test, y_pred))
#Summary of the predictions made by the classifier
print(confusion_matrix(y_test, y_pred))
#to evaluate the quality of the output
```

```
accuracy is 0.9666666666666667
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       0.93      1.00      0.96        13
 Iris-virginica       1.00      0.83      0.91         6

       accuracy                           0.97        30
      macro avg       0.98      0.94      0.96        30
   weighted avg       0.97      0.97      0.97        30

[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
```

In [18]:
```
#Using K-means clustering on Iris dataset:

from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
iris_data=load_iris() #loading iris dataset from sklearn.datasets
iris_df = pd.DataFrame(iris_data.data, columns = iris_data.feature_names) #cre
ating dataframe
kmeans = KMeans(n_clusters=3,init = 'k-means++', max_iter = 100, n_init = 10,
random_state = 0) #Applying Kmeans classifier
y_kmeans = kmeans.fit_predict(x)
print(kmeans.cluster_centers_) #display cluster centers
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],s = 100, c = 'red', label
= 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],s = 100, c = 'blue', labe
l = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],s = 100, c = 'green', lab
el = 'Iris-virginica') #Visualising the clusters - On the first two columns
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],s = 10
0, c = 'black', label = 'Centroids') #plotting the centroids of the clusters
plt.legend()
plt.show()
```

```
[[125.5     6.588   2.974   5.552   2.026]
 [ 25.5     5.006   3.418   1.464   0.244]
 [ 75.5     5.936   2.77    4.26    1.326]]
```



In [ ]: