In [4]:

```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

In [5]:

```
iris.feature_names
```

Out[5]:

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [6]:

```
iris.target_names
```

Out[6]:

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [11]:

```
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

Out[11]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [14]:

```
df['target']=iris.target
df.head()
```

Out[14]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [15]:

```
df[df.target==1].head()
```

Out[15]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

In [16]:

```
df[df.target==2].head()
```

Out[16]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

In [17]:

```
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df.head()
```

Out[17]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

In [18]:

```
df[45:55]
```

Out[18]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
45	4.8	3.0	1.4	0.3	0	setosa
46	5.1	3.8	1.6	0.2	0	setosa
47	4.6	3.2	1.4	0.2	0	setosa
48	5.3	3.7	1.5	0.2	0	setosa
49	5.0	3.3	1.4	0.2	0	setosa
50	7.0	3.2	4.7	1.4	1	versicolor
51	6.4	3.2	4.5	1.5	1	versicolor
52	6.9	3.1	4.9	1.5	1	versicolor
53	5.5	2.3	4.0	1.3	1	versicolor
54	6.5	2.8	4.6	1.5	1	versicolor

In [20]:

```
df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
```

In [21]:

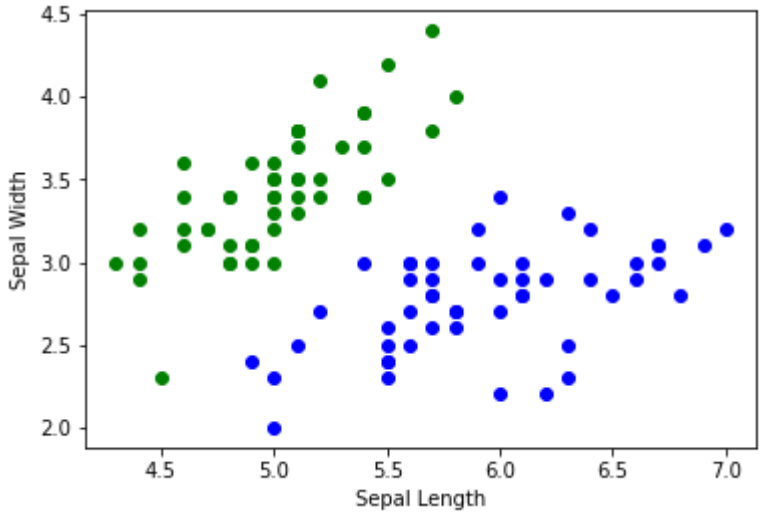
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [26]:

```
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green")
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue")
```

Out[26]:

```
<matplotlib.collections.PathCollection at 0x1e5c0d65cd0>
```

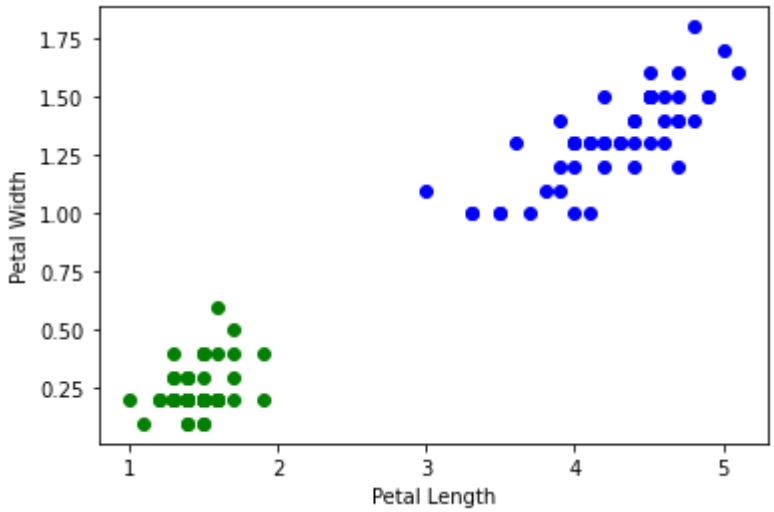


In [27]:

```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green")
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue")
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x1e5c0e77cd0>
```



In [28]:

```
from sklearn.model_selection import train_test_split
```

In [29]:

```
x = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

In [32]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y ,test_size=0.2,random_state=42)
```

In [33]:

```
len(x_train)
```

Out[33]:

```
120
```

In [34]:

```
len(x_test)
```

Out[34]:

```
30
```

In [44]:

```
#KNN CLASSIFIER WITH K=9


from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=9)
```

In [37]:

```
knn.fit(x_train, y_train)
```

Out[37]:

```
KNeighborsClassifier(n_neighbors=9)
```

In [38]:

```
knn.score(x_test, y_test)
```

Out[38]:

```
1.0
```

In [40]:

```
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[40]:

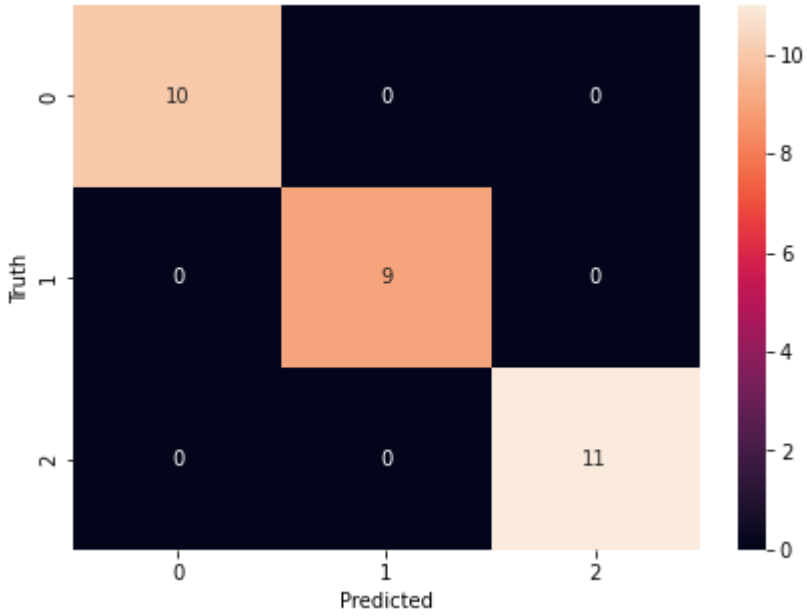
```
array([[10,  0,  0],
       [ 0,  9,  0],
       [ 0,  0, 11]], dtype=int64)
```

In [42]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[42]:

```
Text(42.0, 0.5, 'Truth')
```



In [43]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```