In [2]:
```
# import Necessary libraries. In this exercise we will be using sklearn inherent
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
```

In [3]:
```
# identify the features of the dataset
iris.feature_names
```

Out[3]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [4]:
```
# identify the different kinds on flowers in dataframe
iris.target_names
```

Out[4]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [5]:
```
# get details of petals and sepals for each flower
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

Out[5]:

	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 [6]:
```
# Adding a new column called as Target in the data frame
df['target'] = iris.target
df.head()
```

Out[6]:

	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 [9]:
```
# How many rows in the dataframe as 1 in target
df[df.target==1].head()
```

Out[9]:

	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 [10]:
```
# How many rows in the dataframe as 2 in target
df[df.target==2].head()
```

Out[10]:

	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 [12]:
```
# Adding flower name column in the dataset with the name of the flower
df['flower_name'] =df.target.apply(lambda x: iris.target_names[x])
df.head()
```

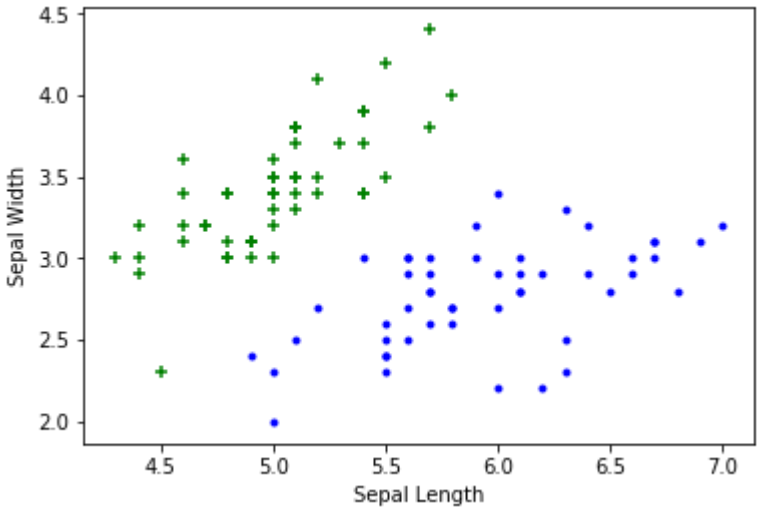
Out[12]:

	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 [13]:
```
# do a plotting of the flower type

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

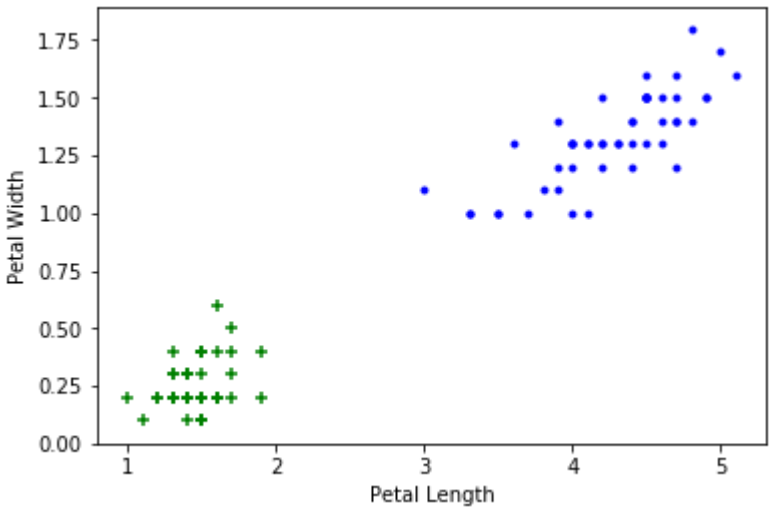
In [18]:
```
# plot the flowers with df0(setosa),df1(versicolor), df2(virginica)
# currently we are plotting only df0 and df1

df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marke
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x21add00fb70>
```



In [19]:
```
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marke
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x21add4cf198>
```



In [20]:
```
# Train Using Support Vector Machine (SVM)
from sklearn.model_selection import train_test_split
```

In [22]:
```
# drop the columns target and flower name from the dataset
X = df.drop(['target','flower_name'], axis='columns')
y = df.target
```

In [23]:
```
# split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [25]:
```
# identify number of records in train dataset
len(X_train)
```

Out[25]:
```
120
```

In [27]:
```
# identify number of records in test dataset
len(X_test)
```

Out[27]:
```
30
```

In [28]:
```
# import SVM from sklearn and create a model object
from sklearn.svm import SVC
model = SVC()
```

In [29]:
```
# fit the model
model.fit(X_train, y_train)
```

Out[29]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [30]:
```
# identify the accuracy of the model
model.score(X_test, y_test)
```

Out[30]:
```
0.9666666666666667
```

In [31]:
```
model.predict([[4.8,3.0,1.5,0.3]])
```

Out[31]:
```
array([0])
```

In [32]:
```
# Evaluate Tune parameters
# 1. Regularization (C)

model_C = SVC(C=1)
model_C.fit(X_train, y_train)
model_C.score(X_test, y_test)
```

Out[32]:
```
0.9666666666666667
```

In [33]:
```
# C value when C = 10

model_C = SVC(C=10)
model_C.fit(X_train, y_train)
model_C.score(X_test, y_test)
```

Out[33]:
```
1.0
```

In [36]:
```
# Evaluate Tune parameters
# 2. Gamma (C)
model_g = SVC(gamma=10)
model_g.fit(X_train, y_train)
model_g.score(X_test, y_test)
```

Out[36]:
```
0.9
```

In [38]:
```
# Evaluate Tune parameters
# 3. Kernel

model_linear_kernal = SVC(kernel='linear')
model_linear_kernal.fit(X_train, y_train)
model_linear_kernal.score(X_test, y_test)
```

Out[38]:
```
0.9666666666666667
```