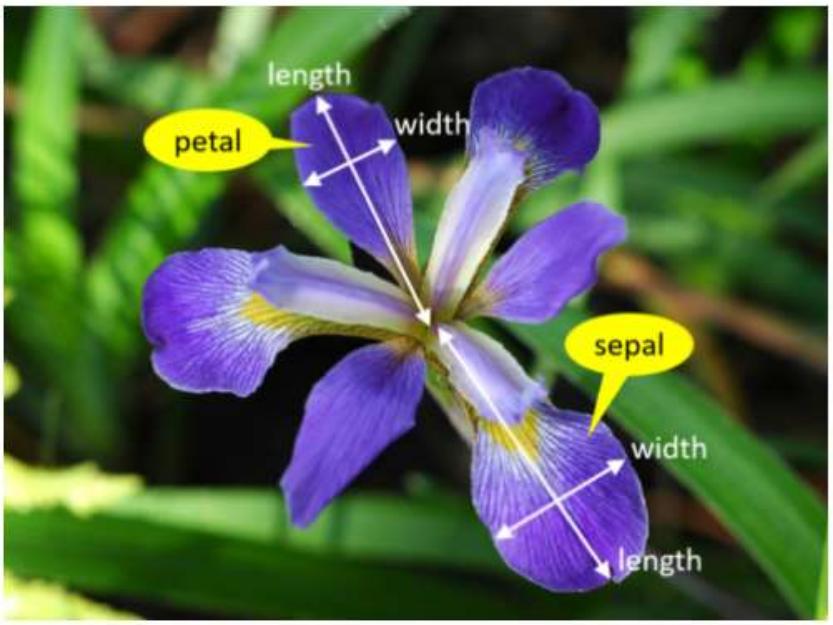
IRIS FLOWER DATASET ANALYSIS

In [2]:
```
from PIL import Image
import matplotlib.pyplot as plt
a=Image.open("D:\Data Analytics Project\Iris Flower dataset analysis/1727763313291.jpg")
plt.imshow(a)
plt.axis('off')
plt.show()
```



Load the dataset

In [3]:
```
import pandas as pd
```

In [4]:
```
data = pd.read_csv("D:\Data Analytics Project\Iris Flower dataset analysis/IRIS.csv")
```

Data Preprocessing

In [5]:
```
# Data preview of first few rows
data.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [6]:
```
# Data preview of last few rows
data.tail(5)
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

In [7]:
```
# Checking Null Values
data.isnull().sum()
```

Out[7]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [8]:
```
data_species = data['species'].value_counts()
data_species
```

Out[8]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

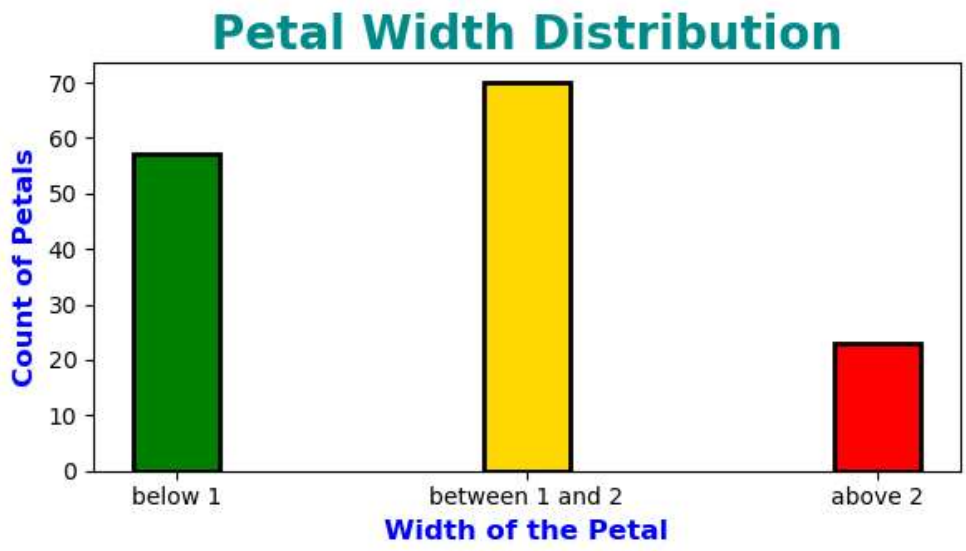
Exploratory Data Analysis (EDA) and Data Visualization

Petal Width Distribution

In [27]:
```
import matplotlib.pyplot as plt
```

In [41]:
```
data_below_1 = data[data['petal_width'] <= 1].value_counts().sum()
data_1_2 = data[(data['petal_width'] > 1) & (data['petal_width'] <= 2)].value_counts().sum()
data_above_3 = data[data['petal_width'] >2].value_counts().sum()
total_petals = data['petal_width'].value_counts().sum()
print('Width below 1 :',data_below_1,'petals')
print('Width between 1 and 2 :',data_1_2,'petals')
print('Width above 2 :',data_above_3,'petals')
print('Total No.of Petals :',total_petals)

# Plot
counts = [data_below_1,data_1_2,data_above_3]
categories = ['below 1','between 1 and 2','above 2']
plt.figure(figsize=(6,3.5))
plt.bar(categories,counts,color=['green','gold','red'],edgecolor='black',width=0.25,linewidth=2)
plt.title('Petal Width Distribution',fontsize=20,fontweight='bold',color='darkcyan')
plt.xlabel('Width of the Petal',fontsize=12,fontweight='bold',color='blue')
plt.ylabel('Count of Petals',fontsize=12,fontweight='bold',color='blue')
plt.tight_layout()
plt.show()
```

```
Width below 1 : 57 petals
Width between 1 and 2 : 70 petals
Width above 2 : 23 petals
Total No.of Petals : 150
```

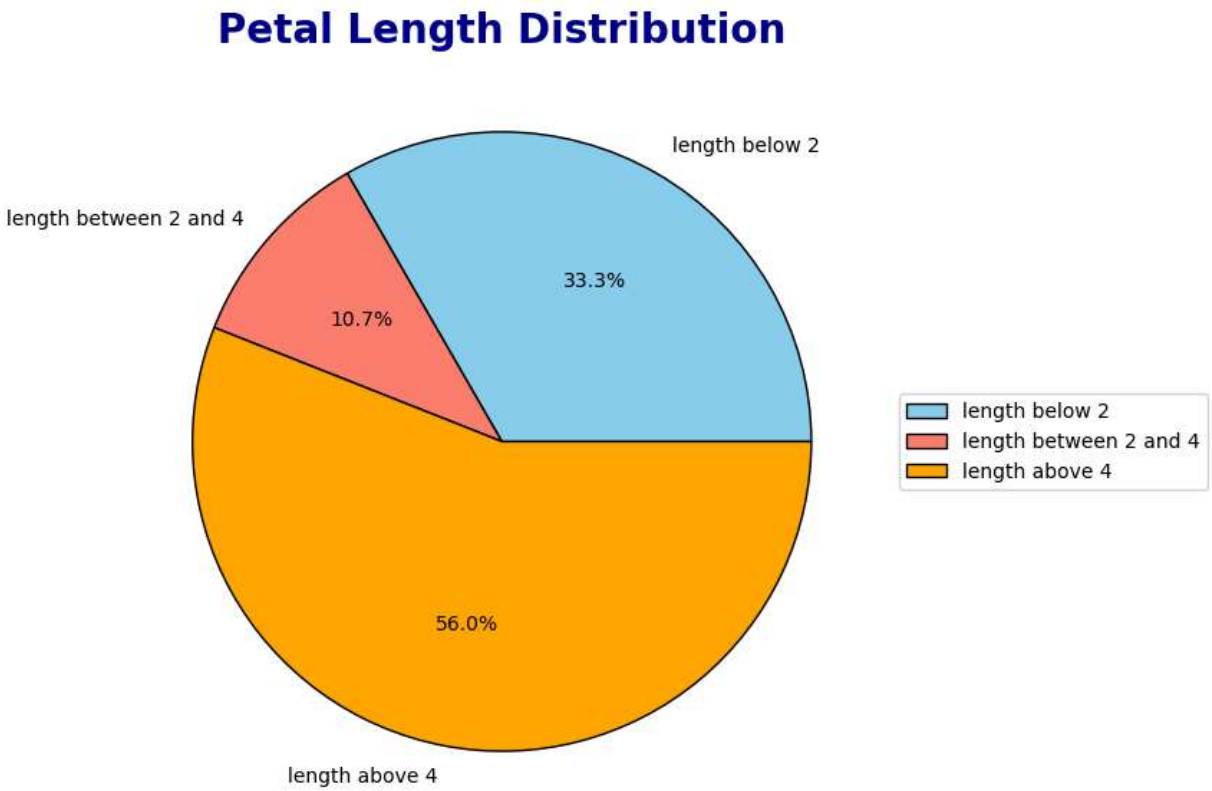


Petal Length Distribution

In [40]:
```
petal_length_below_2 = data[data['petal_length'] <=2 ].value_counts().sum()
petal_length_2_4 = data[(data['petal_length'] >2) & (data['petal_length'] <=4 )].value_counts().s
petal_length_above_4 = data[data['petal_length'] >4 ].value_counts().sum()
print('Length below 2 :',petal_length_below_2,'petals')
print('Length between 2 and 4 :',petal_length_2_4,'petals')
print('Length above 4 :',petal_length_above_4,'petals')
print('Totals No. of Petals :',total_petals)
```

```
Length below 2 : 50 petals
Length between 2 and 4 : 16 petals
Length above 4 : 84 petals
Totals No. of Petals : 150
```

In [46]:
```
# Plot
categories = ['length below 2','length between 2 and 4','length above 4']
counts = [petal_length_below_2,petal_length_2_4,petal_length_above_4]
plt.figure(figsize=(7,7))
plt.pie(counts,labels=categories,autopct='%1.1f%%',colors=['skyblue','salmon','orange'],wedgeprop
plt.title('Petal Length Distribution',fontsize=20,fontweight='bold',color='darkblue')
plt.legend(loc='center left',bbox_to_anchor=(1,0.5))
plt.show()
```

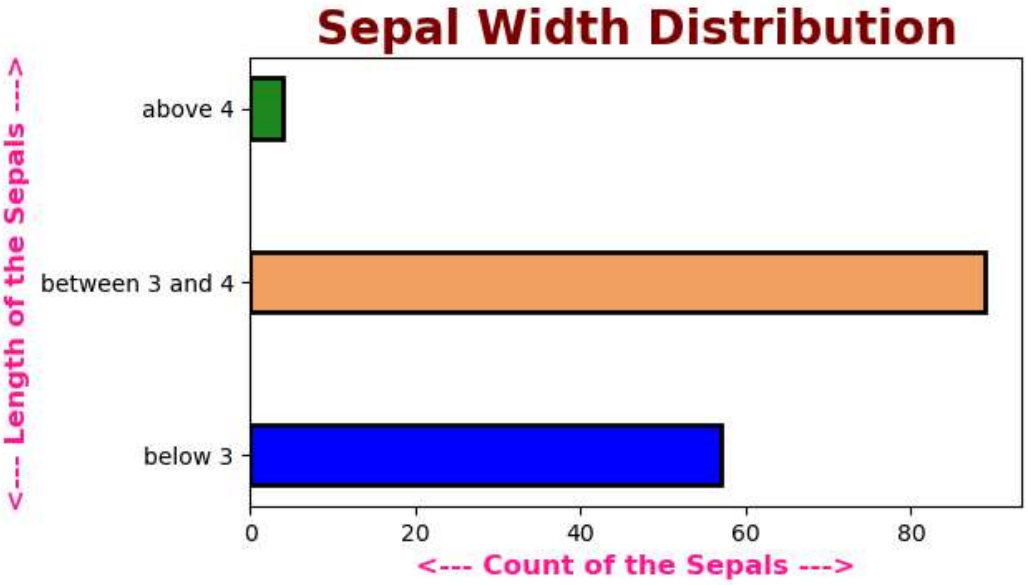


Sepal Width Distribution

In [53]:
```
sepal_width_below_3 = data[data['sepal_width'] < 3].value_counts().sum()
sepal_width_bet_3_4 = data[(data['sepal_width'] >=3) & (data['sepal_width'] <4)].value_counts().s
sepal_width_above_4 = data[data['sepal_width'] >=4].value_counts().sum()
total_sepals = data['sepal_width'].value_counts().sum()
print('Width below 3 :',sepal_width_below_3,'sepals')
print('Width between 3 and 4 :',sepal_width_bet_3_4,'sepals')
print('Width above 4 :',sepal_width_above_4,'sepals')
print('Total No. of Sepals :',total_sepals)
```

```
Width below 3 : 57 sepals
Width between 3 and 4 : 89 sepals
Width above 4 : 4 sepals
Total No. of Sepals : 150
```

In [68]:
```
# Plot
categories = ['below 3','between 3 and 4','above 4']
counts = [sepal_width_below_3,sepal_width_bet_3_4,sepal_width_above_4]
plt.figure(figsize=(6,3.5))
plt.barh(categories,counts,color=['blue','sandybrown','forestgreen'],edgecolor='black',height=0.3
plt.title('Sepal Width Distribution',fontsize=20,fontweight='bold',color='maroon')
plt.ylabel('<--- Length of the Sepals --->',fontsize=12,fontweight='bold',color='deeppink')
plt.xlabel('<--- Count of the Sepals --->',fontsize=12,fontweight='bold',color='deeppink')
plt.show()
```

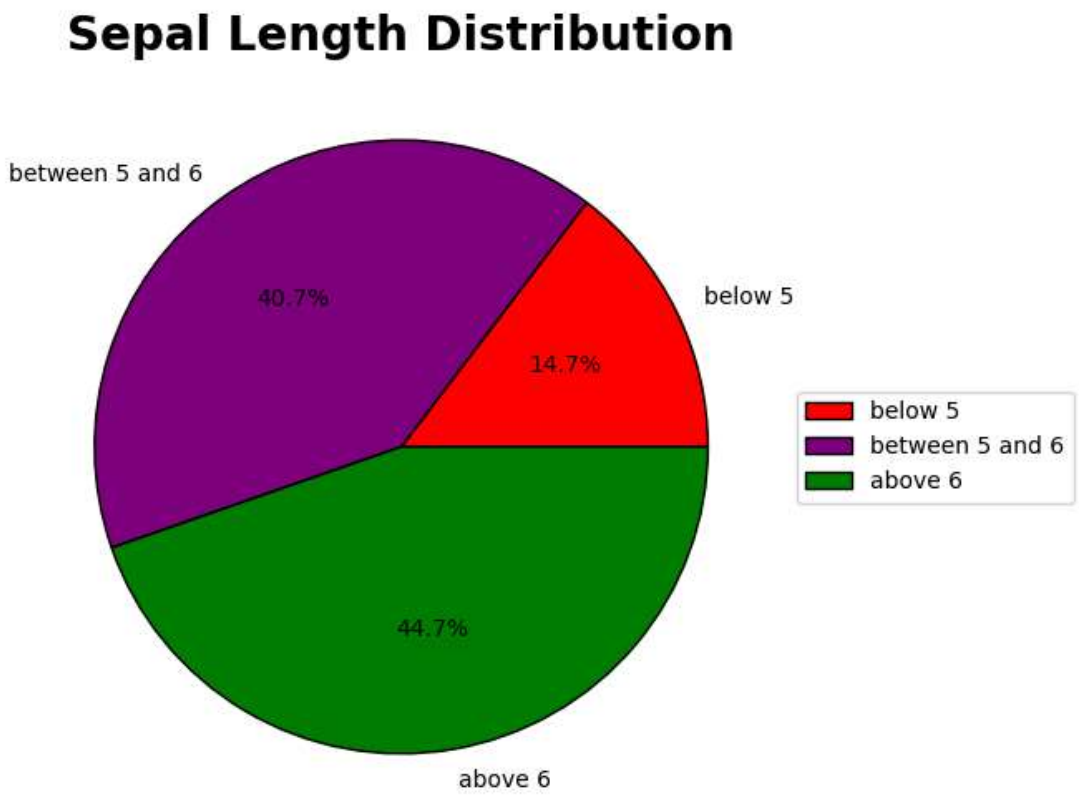


Sepal Length Distribution

In [75]:
```
sepal_length_below_5 = data[data['sepal_length'] < 5].value_counts().sum()
sepal_length_bet_5_6 = data[(data['sepal_length'] >= 5) & (data['sepal_length'] < 6)].value_count
sepal_length_above_6 = data[data['sepal_length'] >= 6].value_counts().sum()
print('Length below 5 :',sepal_length_below_5,'sepals')
print('Length between 5 and 6 :',sepal_length_bet_5_6,'sepals')
print('Length above 6 :',sepal_length_above_6,'sepals')
print('Total No. of Sepals :',total_sepals)
```

```
Length below 5 : 22 sepals
Length between 5 and 6 : 61 sepals
Length above 6 : 67 sepals
Total No. of Sepals : 150
```

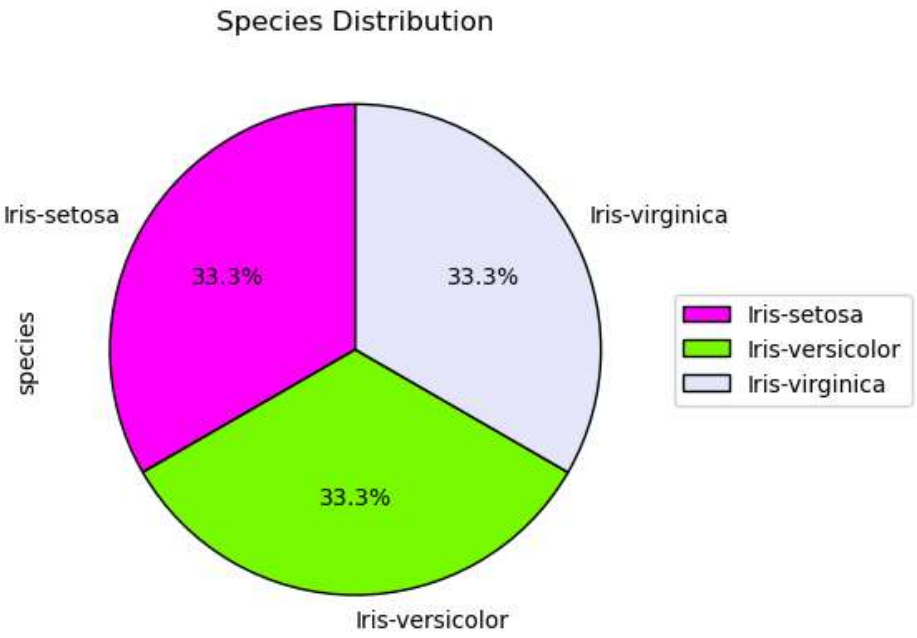
In [87]:
```
# Plot
categories = ['below 5','between 5 and 6','above 6']
counts = [sepal_length_below_5,sepal_length_bet_5_6,sepal_length_above_6]
plt.figure(figsize=(6,6))
plt.pie(counts,labels=categories,colors=['red','purple','green'],autopct='%1.1f%%',wedgeprops={'e
plt.title('Sepal Length Distribution',fontsize=20,fontweight='bold')
plt.legend(loc='center left',bbox_to_anchor=(1,0.5))
plt.show()
```



Species Distribution

In [104]:
```
print('Distribution of Species :')
print(data_species)
print('Total No. of Species :',data_species.sum())
data_species.plot.pie(autopct='%1.1f%%',startangle=90,wedgeprops={'edgecolor':'black'},
                      colors=['magenta','lawngreen','lavender'])
plt.title('Species Distribution')
plt.legend(loc='center left',bbox_to_anchor=(1,0.5))
plt.show()
```

```
Distribution of Species :
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
Total No. of Species : 150
```



CLASSIFICATION MODEL

In [112]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [126]:
```
# Separate features (x) and target (y)
x = data[['sepal_length','sepal_width','petal_length','petal_width']]
y = data['species']

# Split the data into training and testing sets
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=42,test_size=0.2)
```

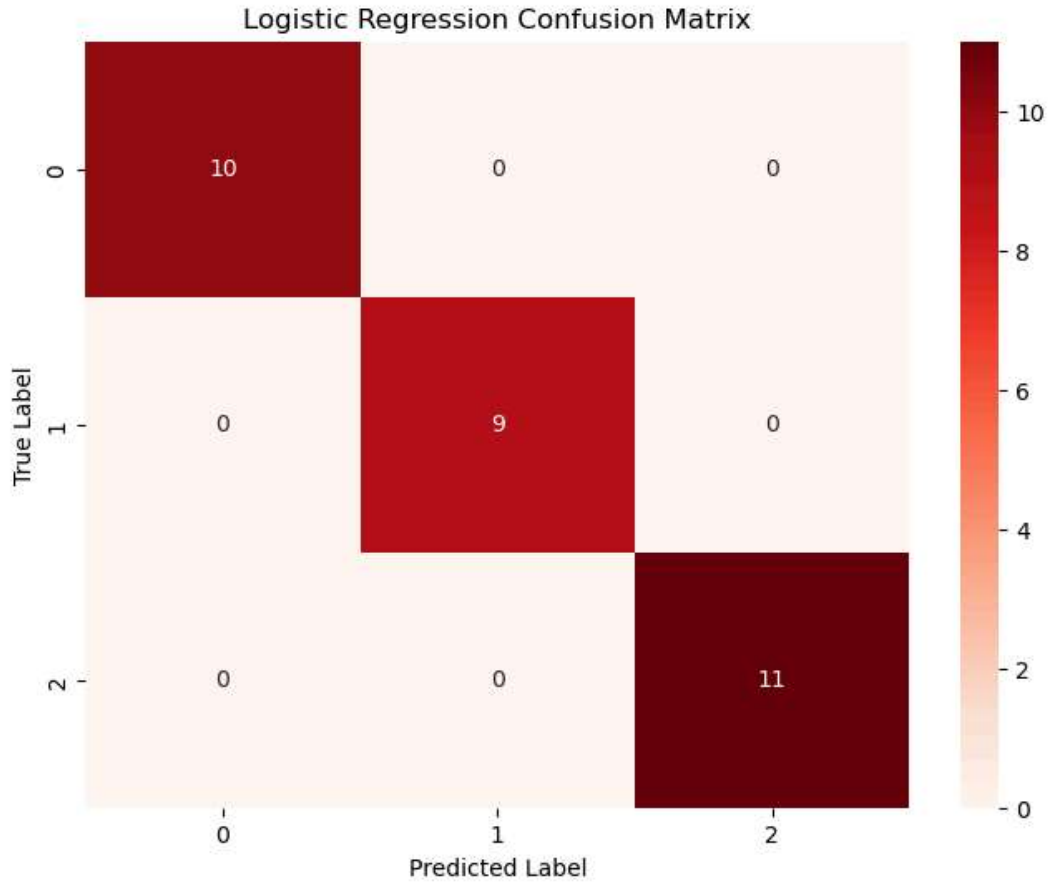
In [ ]:
```

```

1. Logistic Regression

In [139]:
```
lr = LogisticRegression(max_iter=200)
lr.fit(x_train,y_train)

# Model Prediction
y_pred = lr.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
classi_rep = classification_report(y_test,y_pred)
conf_matrix = confusion_matrix(y_test,y_pred)
print("Logistic Regression Accuracy:",accuracy*100,'%')
print('\nClassification Report:')
print(classi_rep)
print('Confusion Matrix:')
print(conf_matrix)
```

```
Logistic Regression Accuracy: 100.0 %

Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       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

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [140]:
```
import seaborn as sns
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix,annot=True,fmt='d',cmap='Reds')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Logistic Regression Confusion Matrix')
plt.show()
```

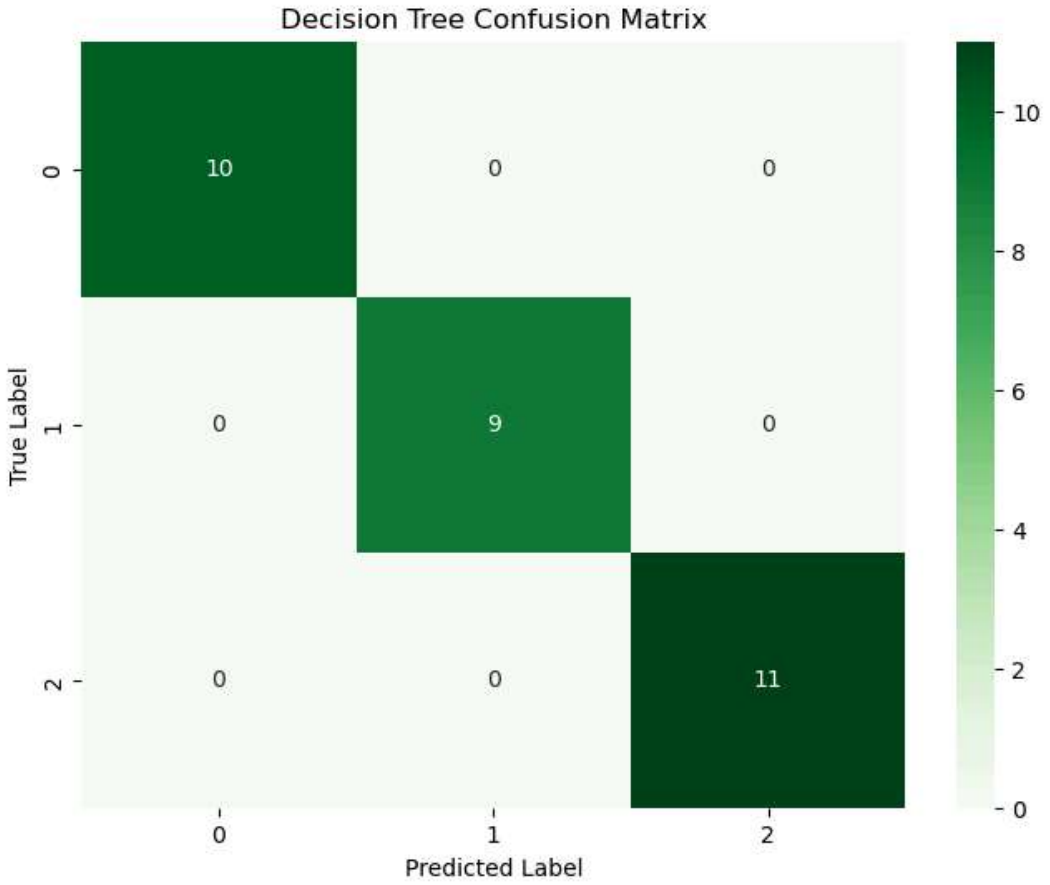


2. Decision Tree Classifier

In [145]:
```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)

# Predict the model
dtc_y_pred = dtc.predict(x_test)
dtc_accuracy = accuracy_score(y_test,dtc_y_pred)
dtc_classi_rep = classification_report(y_test,dtc_y_pred)
dtc_conf_matrix = confusion_matrix(y_test,dtc_y_pred)
print('Decision Tree Accuracy :',dtc_accuracy*100,'%')
print('\nClassification Report:')
print(dtc_classi_rep)
print('Confusion Matrix :')
print(dtc_conf_matrix)
```

```
Decision Tree Accuracy : 100.0 %

Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       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

Confusion Matrix :
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [146]:
```
plt.figure(figsize=(8,6))
sns.heatmap(dtc_conf_matrix,annot=True,fmt='d',cmap='Greens')
plt.title('Decision Tree Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

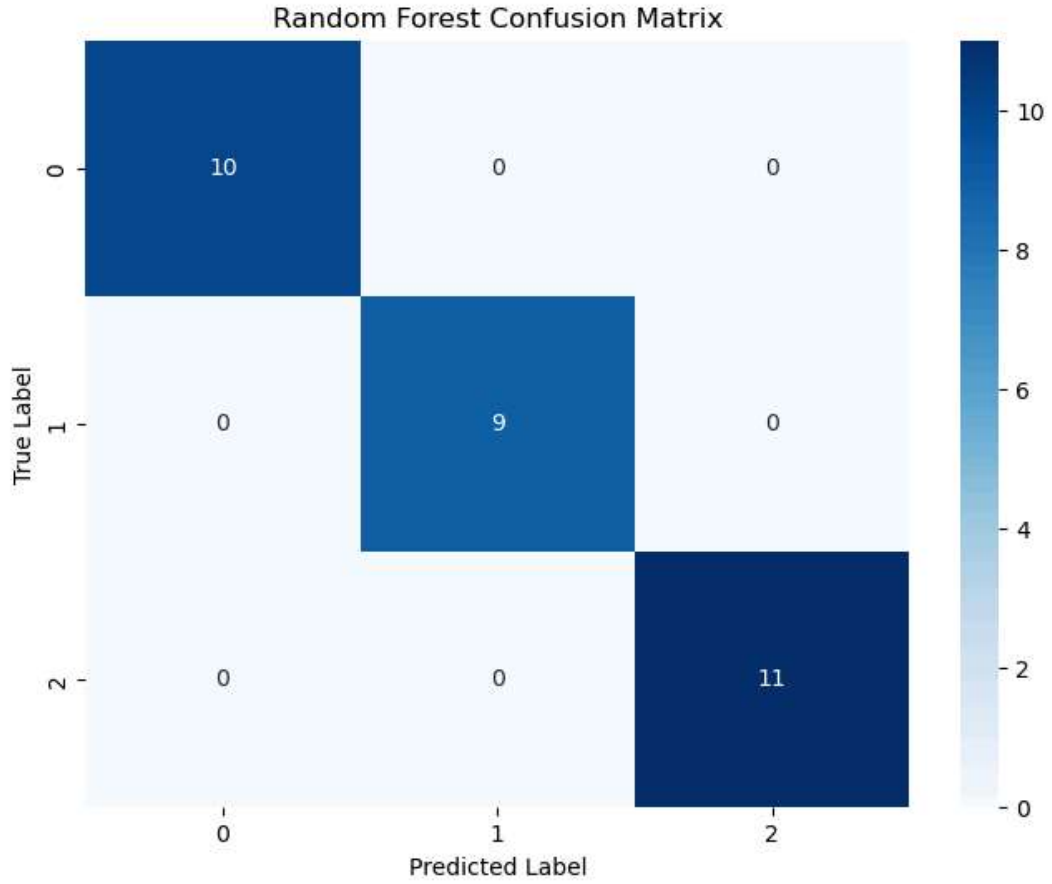


3. Random Forest Classifier

In [147]:
```
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)

# Predit the model
rfc_y_pred = rfc.predict(x_test)
rfc_accuracy = accuracy_score(y_test,rfc_y_pred)
rfc_classi_rep = classification_report(y_test,rfc_y_pred)
rfc_conf_matrix = confusion_matrix(y_test,rfc_y_pred)
print('Random Forest Accuracy :',rfc_accuracy*100,'%')
print('\nClassification Report :')
print(rfc_classi_rep)
print('Confusion Matrix :')
print(rfc_conf_matrix)
```

```
Random Forest Accuracy : 100.0 %

Classification Report :
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       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

Confusion Matrix :
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [148]:
```
plt.figure(figsize=(8,6))
sns.heatmap(rfc_conf_matrix,annot=True,fmt='d',cmap='Blues')
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

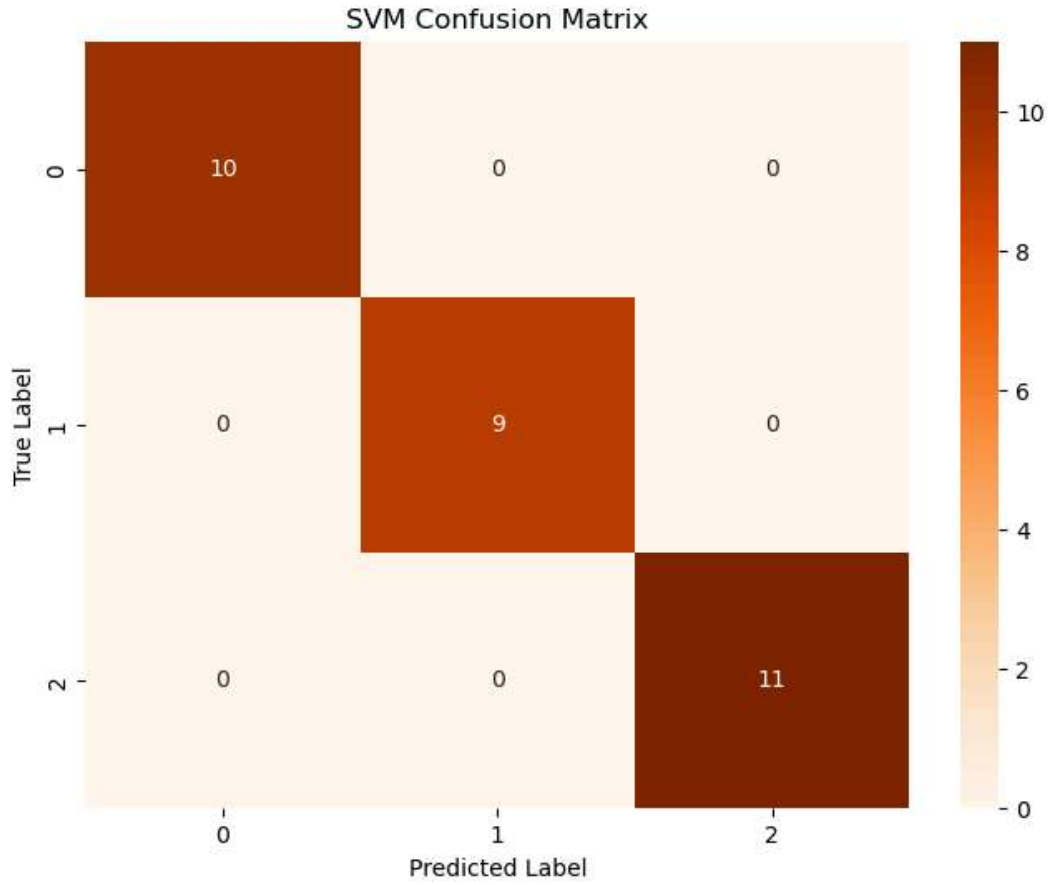


4. Support Vector Machine

In [149]:
```
svm = SVC()
svm.fit(x_train,y_train)

# Predict the Model
svm_y_pred = svm.predict(x_test)
svm_accuracy = accuracy_score(y_test,svm_y_pred)
svm_classi_rep = classification_report(y_test,svm_y_pred)
svm_conf_matrix = confusion_matrix(y_test,svm_y_pred)
print('SVM Accuracy :',svm_accuracy*100,'%')
print('Classification Report :')
print(svm_classi_rep)
print('Confusion Matrix :')
print(svm_conf_matrix)
```

```
SVM Accuracy : 100.0 %
Classification Report :
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       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

Confusion Matrix :
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [157]:
```
plt.figure(figsize=(8,6))
sns.heatmap(svm_conf_matrix,annot=True,fmt='d',cmap='Oranges')
plt.title('SVM Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [162]:
```
plt.figure()
plt.text(0.5,0.6,'Thank You',fontsize=30)
plt.text(0.5,0.4,'CodSoft',fontsize=40,fontweight='bold',color='Blue')
plt.axis('off')
plt.show()
```