Step 1 | Data Pre-Processing

Importing the Libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [2]:
```
dataset = pd.read_csv('Logistic Regression Dataset.csv')
```

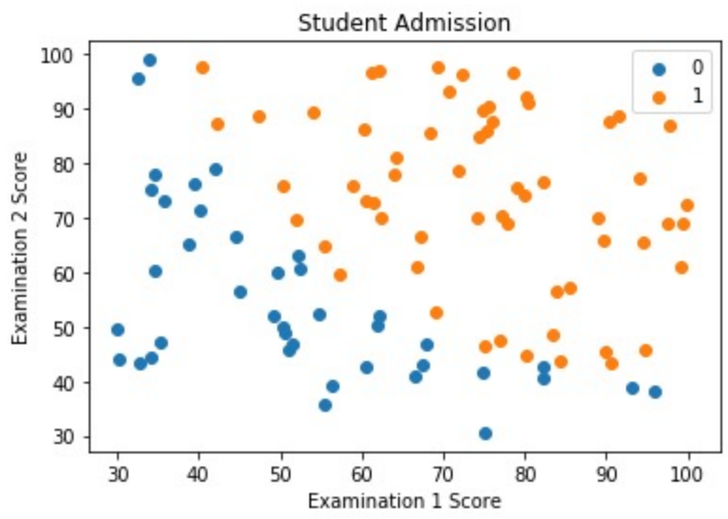
In [3]:
```
#separate two column
x = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, 2].values
```

In [4]:
```
# Admission Status (0 for Not Admitted and 1 for Admitted
admissiones = [0, 1]

# Dataset parameters that we will take into account.
x_axis = 'Examination 1 Score'
y_axis = 'Examination 2 Score'

# Scatter the data on the plot for each validity class separatelly.
for admission in admissiones:
    plt.scatter(
        dataset[x_axis][dataset['Status'] == admission],
        dataset[y_axis][dataset['Status'] == admission],
        label=admission
    )

# Plot the data.
plt.xlabel(x_axis)
plt.ylabel(y_axis)
plt.title('Student Admission')
plt.legend()
plt.show()
```



Splitting the dataset into the Training set and Test set

In [5]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
```

Feature Scaling

Step 2 | Logistic Regression Model

Fitting Logistic Regression to the Training set

In [6]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(solver='liblinear')
classifier.fit(X_train, y_train)
```

Out[6]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
          tol=0.0001, verbose=0, warm_start=False)
```

Step 3 | Predection

Predicting the Test set results

In [7]:
```
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[1 1 0 0 1 1 1 1 0 1 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1]
```

Step 4 | Evaluating The Predection

Making the Confusion Matrix

In [8]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[ 8  3]
 [ 0 14]]
```

Classfication Report

In [9]:
```
from sklearn.metrics import classification_report

print('classification reprot is == >\n')
print(classification_report(y_test,y_pred))
```

```
classification reprot is == >

              precision    recall  f1-score   support

           0       1.00      0.73      0.84        11
           1       0.82      1.00      0.90        14

   micro avg       0.88      0.88      0.88        25
   macro avg       0.91      0.86      0.87        25
weighted avg       0.90      0.88      0.88        25
```