In [1]:
```
import pandas as pd
```

In [4]:
```
data=pd.read_csv("Admission_Predict.csv")
data
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

In [3]:
```
data["Chance of Admit "]
```

Out[3]:
```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       ...
395    0.82
396    0.84
397    0.91
398    0.67
399    0.95
Name: Chance of Admit , Length: 400, dtype: float64
```

In [5]:
```
from sklearn.preprocessing import Binarizer
```

In [6]:
```
limit = Binarizer(threshold=0.75)
```

In [7]:
```
data["Chance of Admit "] = limit.transform(data[["Chance of Admit "]])
```

In [8]:
```
data
```

Out[8]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	1.0
1	2	324	107	4	4.0	4.5	8.87	1	1.0
2	3	316	104	3	3.0	3.5	8.00	1	0.0
3	4	322	110	3	3.5	2.5	8.67	1	1.0
4	5	314	103	2	2.0	3.0	8.21	0	0.0
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	1.0
396	397	325	107	3	3.0	3.5	9.11	1	1.0
397	398	330	116	4	5.0	4.5	9.45	1	1.0
398	399	312	103	3	3.5	4.0	8.78	0	0.0
399	400	333	117	4	5.0	4.0	9.66	1	1.0

400 rows × 9 columns

In [9]:
```
inputdata=data.drop(["Chance of Admit "],axis=1)
```

In [10]:
```
inputdata
```

Out[10]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	1	337	118	4	4.5	4.5	9.65	1
1	2	324	107	4	4.0	4.5	8.87	1
2	3	316	104	3	3.0	3.5	8.00	1
3	4	322	110	3	3.5	2.5	8.67	1
4	5	314	103	2	2.0	3.0	8.21	0
...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1
396	397	325	107	3	3.0	3.5	9.11	1
397	398	330	116	4	5.0	4.5	9.45	1
398	399	312	103	3	3.5	4.0	8.78	0
399	400	333	117	4	5.0	4.0	9.66	1

400 rows × 8 columns

In [16]:
```
outputdata=data["Chance of Admit "]
```

In [18]:
```
outputdata.astype(int)
```

Out[18]:
```
0      1
1      1
2      0
3      1
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Chance of Admit , Length: 400, dtype: int32
```

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
xtrain, ytrain, xtest, ytest = train_test_split(inputdata, outputdata, random_state=0, test_size=0.2
```

In [21]:
```
xtrain
```

Out[21]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
250	251	320	104	3	3.0	2.5	8.57	1
63	64	315	107	2	4.0	3.0	8.50	1
312	313	311	107	4	4.5	4.5	9.00	1
159	160	297	100	1	1.5	2.0	7.90	0
283	284	321	111	3	2.5	3.0	8.90	1
...	...	...	...	...	...	...	...	...
323	324	305	102	2	2.0	2.5	8.18	0
192	193	322	114	5	4.5	4.0	8.94	1
117	118	290	104	4	2.0	2.5	7.46	0
47	48	339	119	5	4.5	4.0	9.70	0
172	173	322	110	4	4.0	5.0	9.13	1

300 rows × 8 columns

In [22]:
```
xtest.shape
```

Out[22]:
```
(300,)
```

In [23]:
```
ytrain
```

Out[23]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
132	133	309	105	5	3.5	3.5	8.56	0
309	310	308	110	4	3.5	3.0	8.60	0
341	342	326	110	3	3.5	3.5	8.76	1
196	197	306	105	2	3.0	2.5	8.26	0
246	247	316	105	3	3.0	3.5	8.73	0
...	...	...	...	...	...	...	...	...
146	147	315	105	3	2.0	2.5	8.48	0
135	136	314	109	4	3.5	4.0	8.77	1
390	391	314	102	2	2.0	2.5	8.24	0
264	265	325	110	2	3.0	2.5	8.76	1
364	365	313	102	3	3.5	4.0	8.90	1

100 rows × 8 columns

In [24]:
```
ytest.shape
```

Out[24]:
```
(100,)
```

In [25]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [26]:
```
classifier = DecisionTreeClassifier(random_state=0)
```

In [27]:
```
classifier.fit(xtrain,xtest)
```

Out[27]:
```
DecisionTreeClassifier(random_state=0)
```

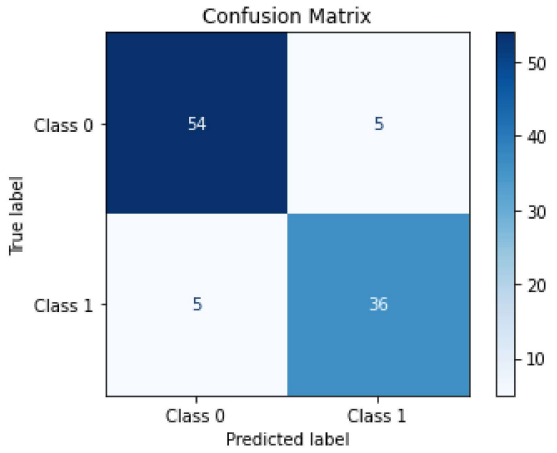
In [30]:
```
predicted=classifier.predict(ytrain)
predicted
```

Out[30]:
```
array([0., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0.,
       1., 0., 1., 1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
       1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0.,
       1., 0., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0.,
       1., 0., 0., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0.,
       1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1.])
```

In [34]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

In [35]:
```
cm = confusion_matrix(ytest,predicted)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Class 0', 'Class 1'])
disp.plot(cmap=plt.cm.Blues, values_format='d')
plt.title("Confusion Matrix")
plt.show()
```



In [36]:
```
from sklearn.metrics import classification_report
```

In [37]:
```
print(classification_report(ytest,predicted))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.92      0.92        59
         1.0       0.88      0.88      0.88        41

    accuracy                           0.90       100
   macro avg       0.90      0.90      0.90       100
weighted avg       0.90      0.90      0.90       100
```

In [38]:
```
pd.DataFrame({"Actual":ytest,"Predicted":predicted})
```

Out[38]:

	Actual	Predicted
132	0.0	0.0
309	0.0	0.0
341	1.0	1.0
196	0.0	0.0
246	0.0	1.0
...	...	...
146	0.0	0.0
135	1.0	1.0
390	0.0	0.0
264	0.0	0.0
364	1.0	1.0

100 rows × 2 columns

In [ ]: