In [3]:
```
#Import Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
#Load the Dataset
df = pd.read_csv("glass.csv")
print(df.head())
```

```
        RI     Na    Mg    Al     Si     K    Ca   Ba   Fe  Type
0  1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.0  0.0     1
1  1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.0  0.0     1
2  1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.0  0.0     1
3  1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.0  0.0     1
4  1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.0  0.0     1
```

In [5]:
```
#Check for Missing Values
print(df.isnull().sum())
```

```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

In [6]:
```
df = df.fillna(df.mean())
```

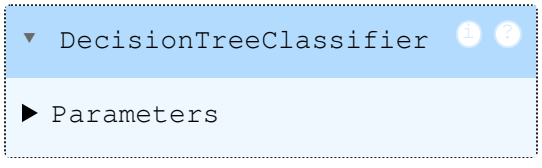
In [7]:
```
#Split Features and Target
X = df.drop("Type", axis=1)    # input features
y = df["Type"]                 # target variable
#If the target variable is not numeric (e.g., "building_windows_float"), use LabelEncoder:
le = LabelEncoder()
y = le.fit_transform(y)
```

In [8]:
```
#Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

In [9]:
```
#Data Scaling (Optional)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [10]:
```
#Train the Model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
```

Out[10]:

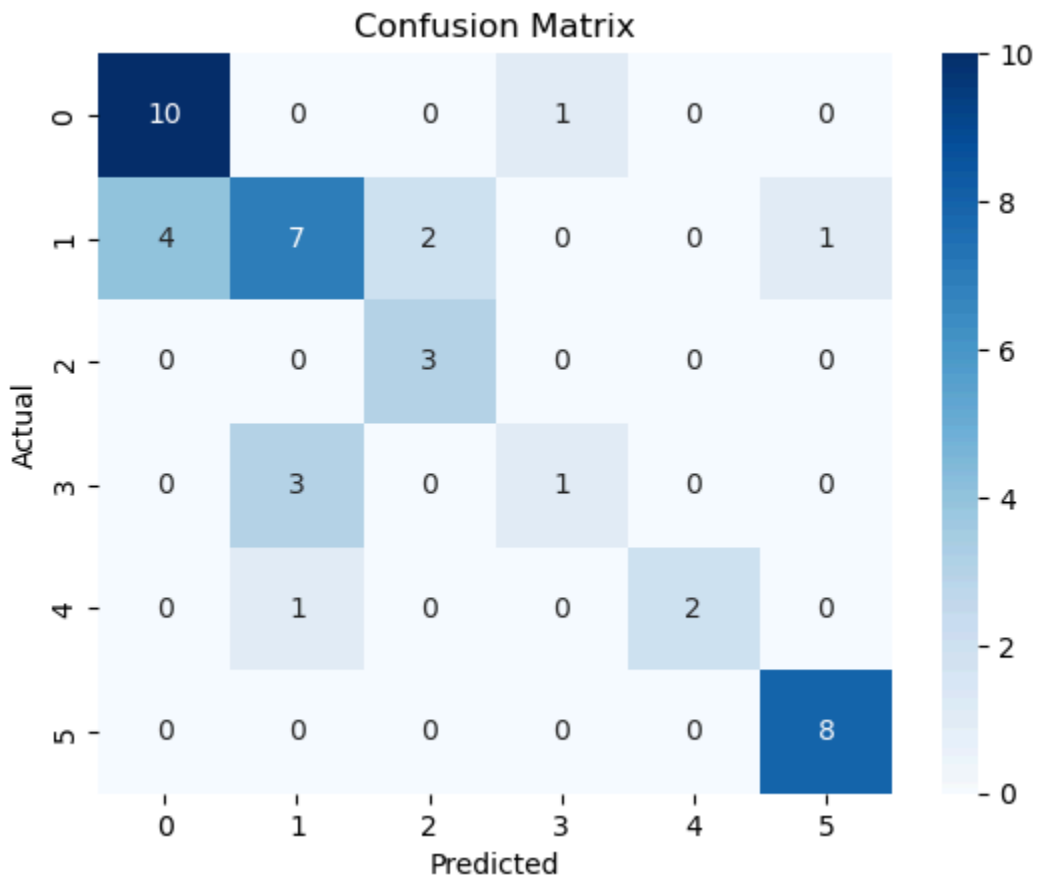


In [11]:
```
#Make Predictions
y_pred = model.predict(X_test)
```

In [12]:
```
#Evaluate the Model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7209302325581395

Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.91      0.80        11
           1       0.64      0.50      0.56        14
           2       0.60      1.00      0.75         3
           3       0.50      0.25      0.33         4
           4       1.00      0.67      0.80         3
           5       0.89      1.00      0.94         8

    accuracy                           0.72        43
   macro avg       0.72      0.72      0.70        43
weighted avg       0.71      0.72      0.70        43
```

In [13]:
```
#Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```


Confusion Matrix
Actual
Predicted
0
1
2
3
4
5
10 0 0 1 0 0
4 7 2 0 0 1
0 0 3 0 0 0
0 3 0 1 0 0
0 1 0 0 2 0
0 0 0 0 0 8
10
8
6
4
2
0

In [ ]: