In [ ]:

```
#Write Python Code to demonstrate implementation of
#Decision Trees Using Python. Use IRIS Dataset.
#OR
#Write Python/R Programming Code to demonstrate
#calculate popular attribute selection measures (ASM) like Information Gain, Gain Ra
#and Gini Index etc.
```

In [1]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

```

In [2]:

```
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
```

In [3]:

```
# Import Library for splitting data
from sklearn.model_selection import train_test_split

```

In [4]:

```
# Creating Train and Test datasets
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state = 50,
                                                    test_size = 0.25)
```

In [5]:

```
# Creating Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0, criterion='gini')
clf.fit(X_train,y_train)
```

Out[5]:

```
DecisionTreeClassifier(random_state=0)
```

In [6]:

```
# Predict Accuracy Score
y_pred = clf.predict(X_test)
```

In [7]:

```
print("Test data accuracy:",accuracy_score(y_test,y_pred))
```

```
Test data accuracy: 0.9473684210526315
```

In [9]:

```
# Create the decision tree classifier using entropy
clf1 = DecisionTreeClassifier(random_state=0, criterion='entropy')
clf1.fit(X_train,y_train)
```

Out[9]:

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [12]:

```
# Predict Accuracy Score
y_pred1 = clf1.predict(X_test)
```

In [13]:

```
print("Test data accuracy:",accuracy_score(y_test,y_pred1))
```

```
Test data accuracy: 0.9473684210526315
```

In [14]:

```
# Get the feature importances (which include the gain ratio)
importances = clf.feature_importances_

```

In [15]:

```
gain_ratios = {}
# Print the gain ratio for each feature
for i, feature in enumerate(iris.feature_names):
    gain = importances[i]
    split = clf.tree_.impurity[i]
    gain_ratio = gain / (split + 1e-7)
    print(f'Gain ratio for {feature}: {gain_ratio:.3f}')

    gain_ratios[feature] = gain_ratio
```

```
Gain ratio for sepal length (cm): 0.030
Gain ratio for sepal width (cm): 201101.269
Gain ratio for petal length (cm): 0.120
Gain ratio for petal width (cm): 6.040
```

In [16]:

```
# Find the feature with the highest gain ratio
best_feature = max(gain_ratios, key=gain_ratios.get)
```

In [17]:

```
print("Best feature for splitting")
print(best_feature)
```

```
Best feature for splitting
sepal width (cm)
```

In [ ]:

```
1
```