In [31]:
```
### check fit of the model
new_data = {
    'income': [1000980],
     'expenses': [100000],
}
```

In [32]:
```
# Convert new data to DataFrame
new_df = pd.DataFrame(new_data)
# Predict the budget class for the new data
predicted_budgetclass = model.predict(new_df)
# Display the predicted budget class
print(f"predicted_budgetclass: {predicted_budgetclass[0]}")
```

```
predicted_budgetclass: notexceeding
```

Result

a python program to decide whether the budget of a company is exceeding or not with decision trees,

with a sample dataset was developed and executed successfully

VIVA

In [1]:
```
### import libraries
#### pandas - functions for analyzing, cleaning, exploring, and manipulating data
import pandas as pd
```

In [2]:
```
### import the data
data = pd.read_csv('consumer data.csv')
```

In [3]:
```
### visualize the data
data
```

Out[3]:

	Customer ID	Age	Dept
0	1	43	Marketing
1	2	32	Sales
2	3	39	Marketing
3	4	37	Tech_Support
4	5	47	R&D
5	6	34	Sales
6	7	35	HR
7	8	40	Tech_Support
8	9	42	Marketing
9	10	45	R&D

In [6]:
```
data.columns = data.columns.str.strip()
```

In [8]:
```
features = ['Age']
print(features)
```

```
['Age']
```

In [10]:
```
### import libraries
#### decision tree - to solve classification problems and categorize objects depending on their le
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
features = ['Age']
target_attribute = 'Dept'
```

In [11]:
```
### creating and training a decision tree classifier
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels = train_test_split(data[features], data[target_at
                                                                    test_size=0.2, random_state=42
```

In [12]:
```
# Create and train the decision tree model
model = DecisionTreeClassifier()
model.fit(train_data, train_labels)
```

Out[12]:
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [13]:
```
# Predict on the testing set
test_predictions = model.predict(test_data)
```

In [14]:
```
# Evaluate the model
accuracy = accuracy_score(test_labels, test_predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 100.00%
```

In [17]:
```
### check fit of the model
new_data = {
    'Age': [35]
}
```

In [18]:
```
# Convert new data to DataFrame
new_df = pd.DataFrame(new_data)
# Predict the budget class for the new data
predicted_budgetclass = model.predict(new_df)
# Display the predicted budget class
print(f"predicted_budgetclass: {predicted_budgetclass[0]}")
```

```
predicted_budgetclass: HR
```

In [ ]: