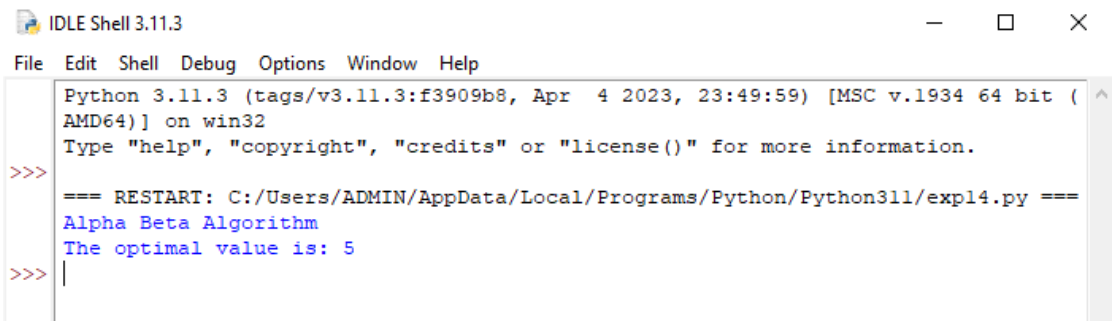
AIM: Write the python program to implement Apha & Beta pruning algorithm for gaming

code:

```
print("Alpha Beta Algorithm")

MAX, MIN = 1000, -1000


def minimax(depth, nodeIndex, maximizingPlayer, values, alpha, beta):
    # Base case: if we reach the maximum depth, return the value at this node
    if depth == 3:
        return values[nodeIndex]


    if maximizingPlayer:
        best = MIN
        # Explore left and right children
        for i in range(2):
            val = minimax(depth + 1, nodeIndex * 2 + i, False, values, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)
            # Alpha-Beta Pruning
            if beta <= alpha:
                break
        return best
    else:
        best = MAX
        for i in range(2):
            val = minimax(depth + 1, nodeIndex * 2 + i, True, values, alpha, beta)
            best = min(best, val)
            beta = min(beta, best)
```

```
        # Alpha-Beta Pruning
        if beta <= alpha:
            break
    return best

if __name__ == "__main__":
    # Example leaf node values
    values = [3, 5, 6, 9, 1, 2, 0, -1]
    optimal_value = minimax(0, 0, True, values, MIN, MAX)
    print("The optimal value is:", optimal_value)
```

OUTPUT:



AIM: Write the python program to implement Decision Tree

CODE:

```
# Import necessary libraries
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the iris dataset as an example
iris = datasets.load_iris()
X = iris.data
```

```
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Decision Tree classifier
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Evaluate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

OUTPUT: