```
import numpy as np # Import the numpy library for numerical operations
from sklearn.datasets import load iris # Import the load iris function from scikit-learn datasets
from sklearn.preprocessing import StandardScaler # Import StandardScaler for data
standardization
from sklearn.decomposition import PCA # Import PCA for principal component analysis
from sklearn.cluster import KMeans # Import KMeans for k-means clustering
# Step 1: Load the Iris dataset
iris = load _iris # Load the Iris dataset into the variable iris
X = iris.data # Extract the features from the Iris dataset
# Step 2: Standardize the data
scaler = StandardScaler # Create a StandardScaler object for data standardization
X_scaled = scaler.fit _transform(X) # Standardize the features and store them in X_ scaled
# Step 3: Perform PCA
pca = PCAn_ components=2) # Create a PCA object to reduce the data to 2 principal
components
X_pea = pca.fit_transform(X _scaled) # Apply PCA on the standardized data and store the
transformed data in
X_pca
# Step 4: Print principal component details
print("Principal Component Details:") # Print the header for principal component details
print"inExplained Variance Ratio:", pca.explained_variance_ratio_) # Print the explained variance
ratio of each
principal component
print"inPrincipal Components:") # Print the header for principal components
print(pca.components_) # Print the principal components
# Step 5: Apply K-means clustering on PCA-reduced data
kmeans = KMeans(n_clusters=3, random_ state-42, n_ init=10) # Create a KMeans object with 3
clusters,
random state for reproducibility, and 10 initializations
kmeans.fit(X_pca) # Fit the KMeans algorithm on the PCA-reduced data
y_kmeans = kmeans.predict(X_pca) # Predict the cluster for each data point and store the
cluster labels in
y_kmeans
# Step 6: Print cluster centers and cluster sizes
print("inCluster Centers (in PCA-reduced space):") # Print the header for cluster centers
for i, center in enumerate(kmeans.cluster _centers ): # Iterate over the cluster centers
```

```
printf'Cluster (i+1}: {center)") # Print the center of each cluster
print("inCluster Sizes:") # Print the header for cluster sizes
for i, size in enumerate(np.bincount(y_kmeans)): # Iterate over the sizes of each cluster
print(f"Cluster {i+1}: {size)")
```