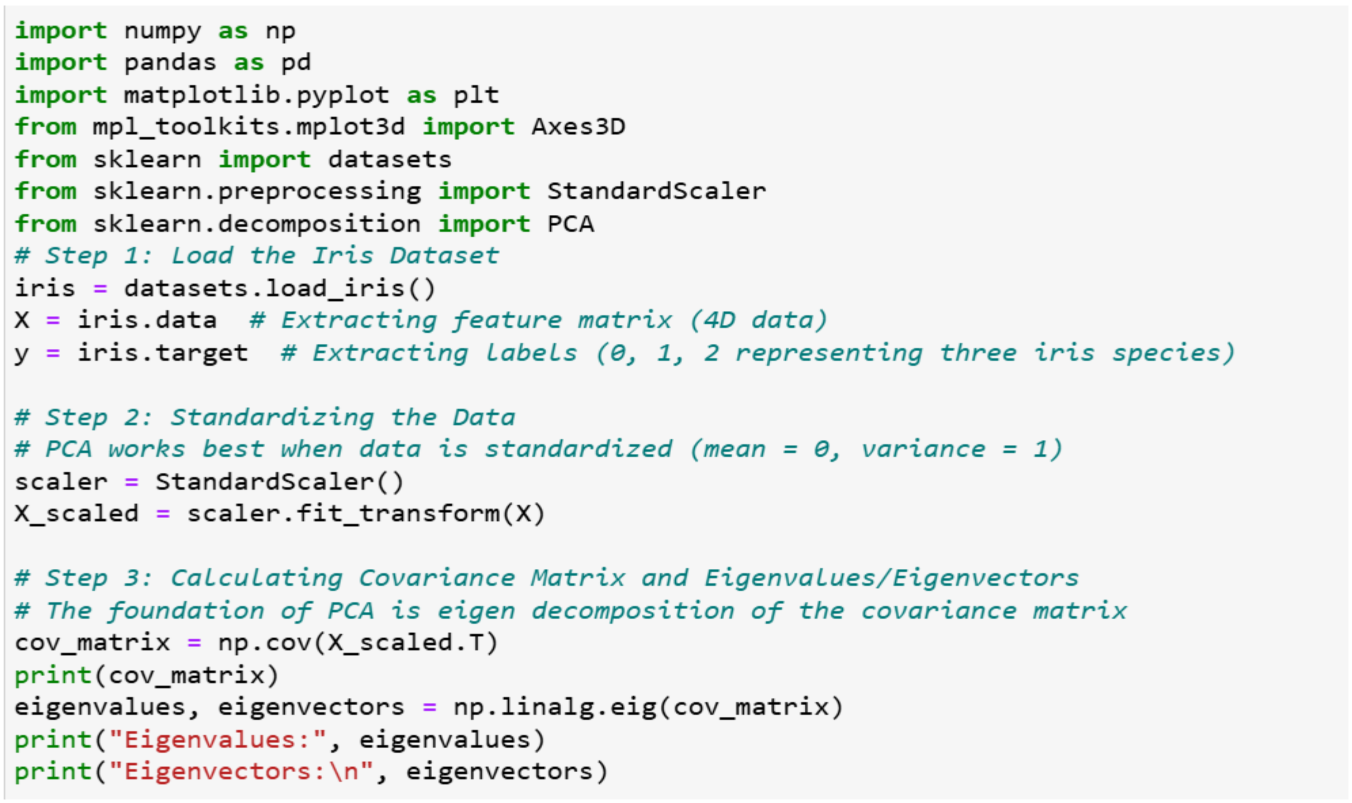
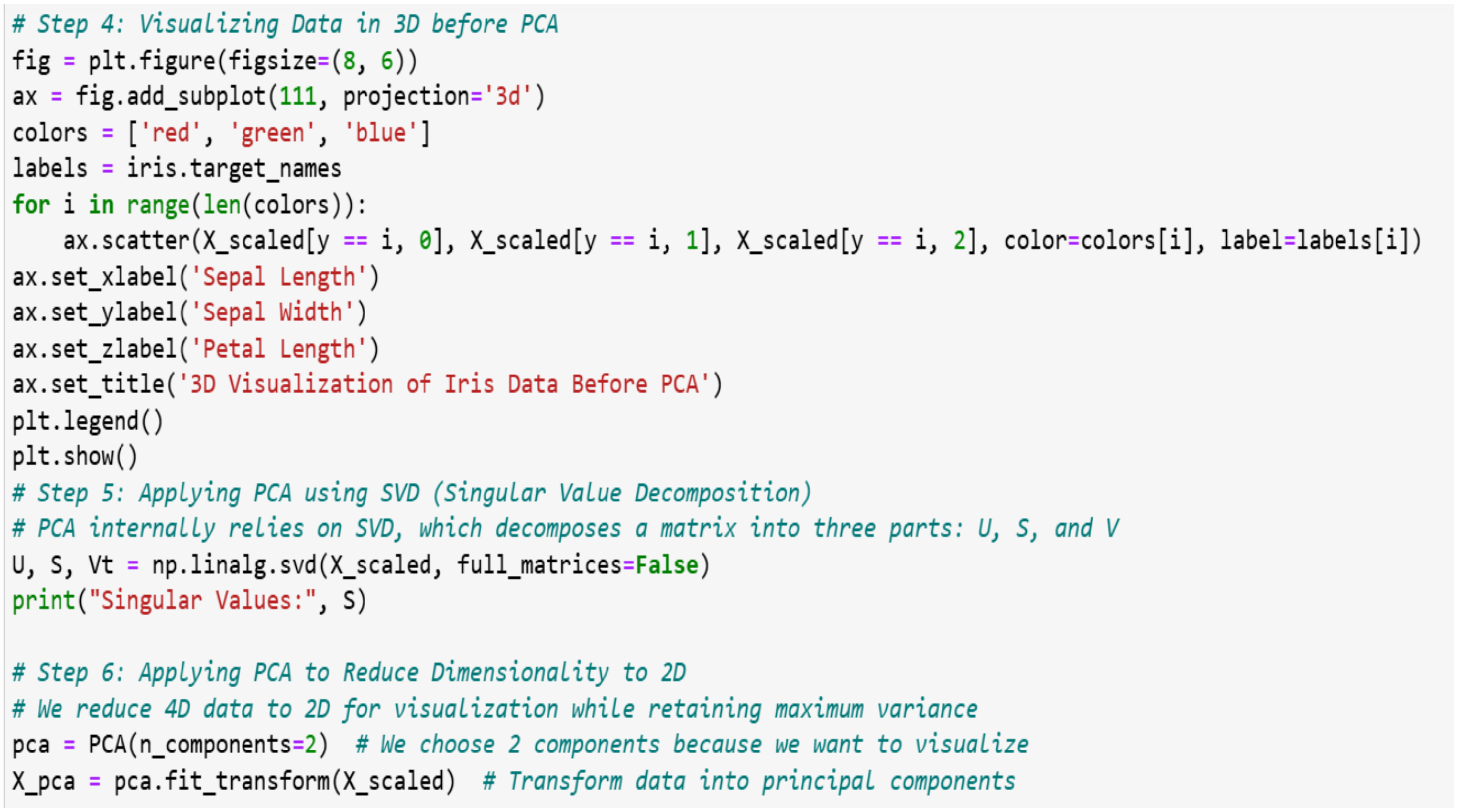
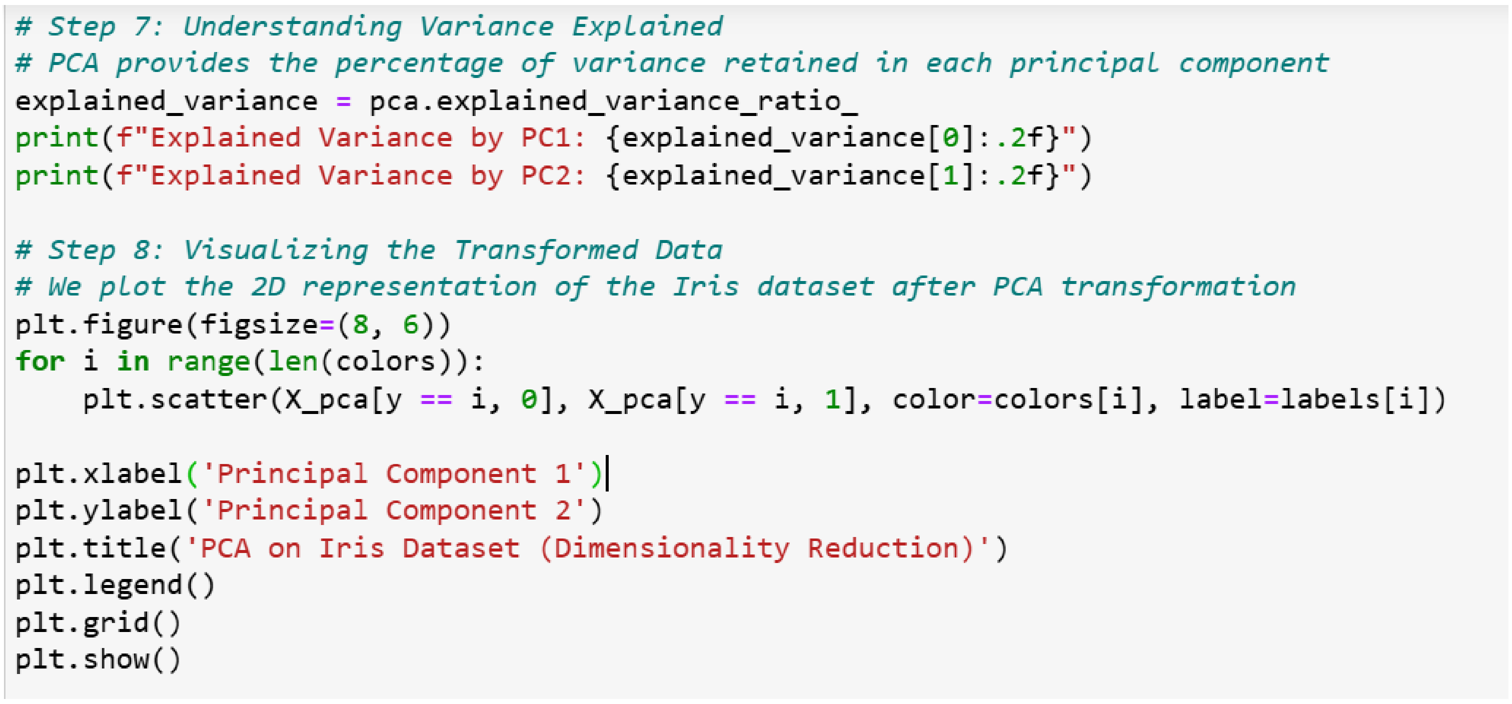
3. Develop a program to implement Principal Component Analysis (PCA) for reducing the dimensionality of the Iris dataset from 4 features to 2



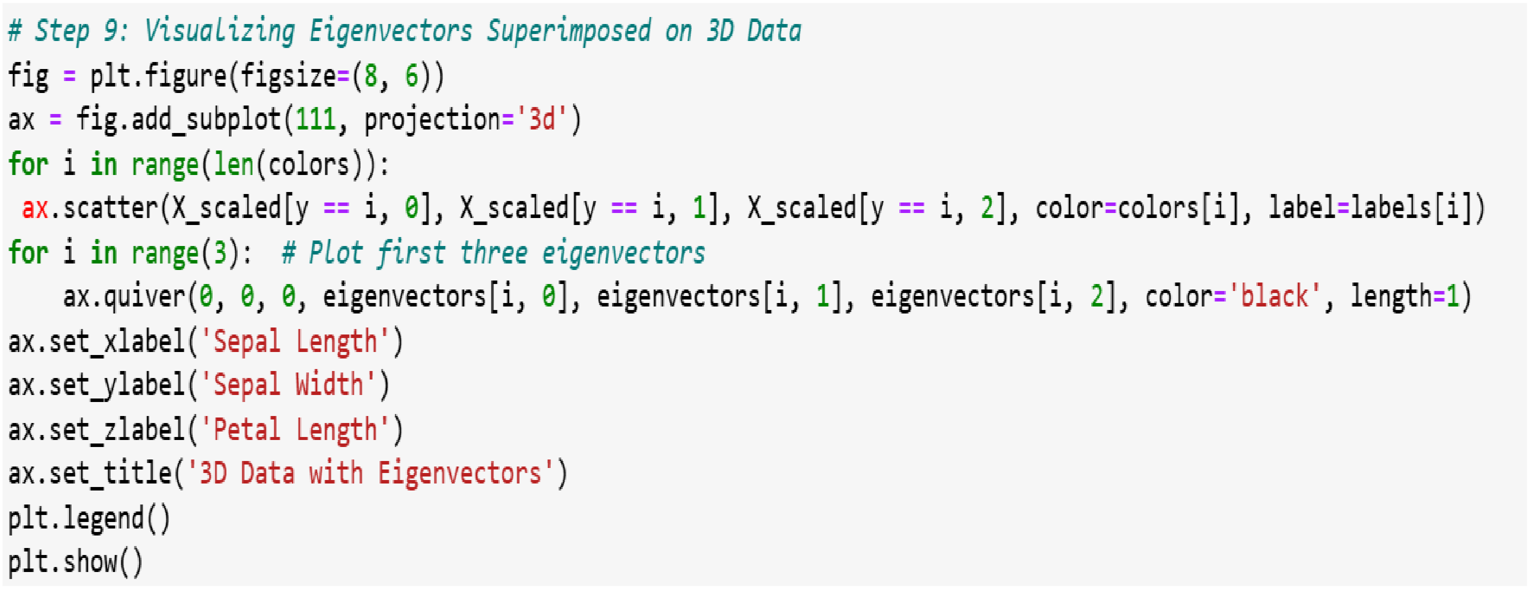
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# Step 1: Load the Iris Dataset
iris = datasets.load_iris()
X = iris.data  # Extracting feature matrix (4D data)
y = iris.target  # Extracting labels (0, 1, 2 representing three iris species)

# Step 2: Standardizing the Data
# PCA works best when data is standardized (mean = 0, variance = 1)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Calculating Covariance Matrix and Eigenvalues/Eigenvectors
# The foundation of PCA is eigen decomposition of the covariance matrix
cov_matrix = np.cov(X_scaled.T)
print(cov_matrix)
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)
print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)
```



```python
# Step 4: Visualizing Data in 3D before PCA
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
colors = ['red', 'green', 'blue']
labels = iris.target_names
for i in range(len(colors)):
    ax.scatter(X_scaled[y == i, 0], X_scaled[y == i, 1], X_scaled[y == i, 2], color=colors[i], label=labels[i])
ax.set_xlabel('Sepal Length')
ax.set_ylabel('Sepal Width')
ax.set_zlabel('Petal Length')
ax.set_title('3D Visualization of Iris Data Before PCA')
plt.legend()
plt.show()
# Step 5: Applying PCA using SVD (Singular Value Decomposition)
# PCA internally relies on SVD, which decomposes a matrix into three parts: U, S, and V
U, S, Vt = np.linalg.svd(X_scaled, full_matrices=False)
print("Singular Values:", S)

# Step 6: Applying PCA to Reduce Dimensionality to 2D
# We reduce 4D data to 2D for visualization while retaining maximum variance
pca = PCA(n_components=2)  # We choose 2 components because we want to visualize
X_pca = pca.fit_transform(X_scaled)  # Transform data into principal components
```



```python
# Step 7: Understanding Variance Explained
# PCA provides the percentage of variance retained in each principal component
explained_variance = pca.explained_variance_ratio_
print(f"Explained Variance by PC1: {explained_variance[0]:.2f}")
print(f"Explained Variance by PC2: {explained_variance[1]:.2f}")

# Step 8: Visualizing the Transformed Data
# We plot the 2D representation of the Iris dataset after PCA transformation
plt.figure(figsize=(8, 6))
for i in range(len(colors)):
    plt.scatter(X_pca[y == i, 0], X_pca[y == i, 1], color=colors[i], label=labels[i])

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA on Iris Dataset (Dimensionality Reduction)')
plt.legend()
plt.grid()
plt.show()
```



```python
# Step 9: Visualizing Eigenvectors Superimposed on 3D Data
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
for i in range(len(colors)):
 ax.scatter(X_scaled[y == i, 0], X_scaled[y == i, 1], X_scaled[y == i, 2], color=colors[i], label=labels[i])
for i in range(3):  # Plot first three eigenvectors
    ax.quiver(0, 0, 0, eigenvectors[i, 0], eigenvectors[i, 1], eigenvectors[i, 2], color='black', length=1)
ax.set_xlabel('Sepal Length')
ax.set_ylabel('Sepal Width')
ax.set_zlabel('Petal Length')
ax.set_title('3D Data with Eigenvectors')
plt.legend()
plt.show()
```