```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from numpy import array
from numpy import diag
from numpy import dot
from numpy import zeros
from sklearn.decomposition import PCA
from numpy import dot
from numpy.linalg import inv
from sklearn.metrics import mean_squared_error
```

Generating data

```
X = np.array([3,2,1,2,4,5,1,2,3,0,2,5])
X = np.reshape(X,(4,3))
print(X)
```

```
[[3 2 1]
 [2 4 5]
 [1 2 3]
 [0 2 5]]
```

Finding sample mean

```
mean = np.mean(X,axis = 0, dtype='float64')
print(f"Mean of the columns of X {mean}")
```

```
Mean of the columns of X [1.5 2.5 3.5]
```

Zero centering of the samples

```
X = X - mean
print(X)
```

```
[[ 1.5 -0.5 -2.5]
 [ 0.5  1.5  1.5]
 [-0.5 -0.5 -0.5]
 [-1.5 -0.5  1.5]]
```

PCA by Eigen value decomposition of covariance matrix

```
from numpy import cov
from numpy.linalg import eig
V = cov(X.T)
Evalues, Evectors = eig(V)
print(f"Eigen vectors = \n {Evectors}")
print(f"Eigen values = \n {Evalues}")
```

```
Eigen vectors =
 [[-0.45056922 -0.66677184 -0.59363515]
 [ 0.19247228 -0.72187235  0.66472154]
 [ 0.87174641 -0.18524476 -0.45358856]]
Eigen values =
 [4.74888619 1.56450706 0.01994008]
```

```
# Converting Evalues to a diagonal matrix
Evalues_diag = zeros((X.shape[0], X.shape[1]))
Evalues_diag[:X.shape[1], :X.shape[1]] = diag(Evalues)
print(f"Diagonalized form of Evalues = \n {Evalues_diag}")
```

```
Diagonalized form of Evalues =
 [[4.74888619 0.          0.        ]
 [0.         1.56450706 0.        ]
 [0.         0.         0.01994008]
 [0.         0.         0.        ]]
```

Projecting X using the Eigen vectors

```
k = 2
Evectors_k = Evectors[:,0:k]
proj = X.dot(Evectors_k)
print(f"Selecting 2 principal axes = \n {Evectors_k}")
print("\n")
print(f"Projected X using the above principal axes = \n {proj}")
```

```
Selecting 2 principal axes =
 [[-0.45056922 -0.66677184]
 [ 0.19247228 -0.72187235]
 [ 0.87174641 -0.18524476]]


Projected X using the above principal axes =
 [[-2.95145599 -0.17610969]
 [ 1.37104342 -1.69406159]
 [-0.30682473  0.78694448]
 [ 1.8872373   1.0832268 ]]
```

Reconstruction of X

```
recon = proj.dot(Evectors_k.T)+mean
print(f"Reconstruction of X using the new basis = \n {recon}")
```

```
Reconstruction of X using the new basis =
 [[ 2.94726021  2.05905526  0.95970224]
 [ 2.0118026   3.98678407  5.0090182 ]
 [ 1.11353336  1.87287129  3.0867493 ]
 [-0.07259617  2.08128939  4.94453025]]
```

Reconstruction error

```
print(mean_squared_error(X+mean, recon))
```

```
0.004985020477602166
```

An alternate way to perform PCA: PCA by singular value decomposition of data

```
U, S, VT = np.linalg.svd(X)

print(f"Unitary matrix = \n {U}")
print(f"Shape of unitary matrix = {U.shape}")
print("\n")
print(f"Diagonal matrix of singular values = \n {S}")
print(f"Shape of this matrix = {S.shape}")
print("\n")
print(f"Matrix of principal axes = \n {VT}")
print(f"Shape of this matrix = {VT.shape}")
print("\n")

# Converting S to a diagonal matrix
S_diag = zeros((X.shape[0], X.shape[1]))
S_diag[:X.shape[1], :X.shape[1]] = diag(S)
print(f"Diagonalized form of S = \n {S_diag}")
```

```
Unitary matrix = 
 [[-0.78195148 -0.08128939  0.36324086  0.5       ]
 [ 0.36324086 -0.78195148 -0.08128939  0.5       ]
 [-0.08128939  0.36324086 -0.78195148  0.5       ]
 [ 0.5         0.5         0.5         0.5       ]]
Shape of unitary matrix = (4, 4)


Diagonal matrix of singular values = 
 [3.77447461 2.1664536  0.24458178]
Shape of this matrix = (3,)


Matrix of principal axes = 
 [[-0.45056922  0.19247228  0.87174641]
 [-0.66677184 -0.72187235 -0.18524476]
 [ 0.59363515 -0.66472154  0.45358856]]
Shape of this matrix = (3, 3)


Diagonalized form of S = 
 [[3.77447461 0.         0.        ]
 [0.         2.1664536  0.        ]
 [0.         0.         0.24458178]
 [0.         0.         0.        ]]
```

- Eigen vectors are the columns of 'V' or rows of 'VT'.
- Eigen values are present in the diagonal matrix of S.

Reconstruction with minimum loss using first 2 components (k=2)

```
k = 2
```

```
U_k = U.T[0:k][0:k]
U_kT = U_k.T
S_diag_k = S_diag[0:2,0:2]
VT_k = VT[0:k]
```

PC scores or Projected X

```
projectedX = U_kT.dot(S_diag_k)
print(projectedX)
```

```
[[-2.95145599 -0.17610969]
 [ 1.37104342 -1.69406159]
 [-0.30682473  0.78694448]
 [ 1.8872373   1.0832268 ]]
```

```
reconstruct_k = U_kT.dot(S_diag_k.dot(VT_k))
reconstruct_k = reconstruct_k + mean
print(reconstruct_k)
```

```
[[ 2.94726021  2.05905526  0.95970224]
 [ 2.0118026   3.98678407  5.0090182 ]
 [ 1.11353336  1.87287129  3.0867493 ]
 [-0.07259617  2.08128939  4.94453025]]
```

Reconstruction error

```
print(mean_squared_error(X+mean, reconstruct_k))
```

```
0.0049850204776021615
```