Algorithm 1: Principal Component Analysis (PCA)

```
Input: Data matrix $X$ of dimensions $N \times M$ (rows are observations, columns are
       features); number of principal components $q$
Output: Matrix $Z$ of dimensions $N \times q$ containing the $q$ principal components
1 Center the data by subtracting the mean of each column (feature) from the corresponding
  column in $X$
2 Compute the covariance matrix $\Sigma = \frac{1}{N} X^T X$
3 Perform eigenvalue decomposition on $\Sigma$ to obtain eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ and
  eigenvectors $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_M$
4 Sort the eigenvectors by decreasing eigenvalue magnitudes
5 Select the first $q$ eigenvectors $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_q$
6 Form the projection matrix $\boldsymbol{\Gamma}_q$ using the selected eigenvectors as columns
7 Compute the principal components $Z = X\boldsymbol{\Gamma}_q$
8 return $Z$
```

```python
import numpy as np

def PCA(X, q):
    """
    Perform Principal Component Analysis (PCA) on the input data matrix X.

    Parameters:
    X (numpy.ndarray): The input data matrix of shape (N, M), where N is the number of samples
                       and M is the number of features.
    q (int): The number of principal components to return.

    Returns:
    numpy.ndarray: A matrix Z of dimensions (N, q) containing the q principal components.
    """
    # Step 1: Center the data by subtracting the mean of each column (feature)
    N, M = X.shape
    mean = np.mean(X, axis=0)
    X_centered = X - mean

    # Step 2: Compute the covariance matrix
    covariance_matrix = X_centered.T @ X_centered / N

    # Step 3: Perform eigenvalue decomposition on the covariance matrix
    eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)

    # Step 4: Sort the eigenvectors by decreasing eigenvalue magnitudes
    sorted_indices = np.argsort(-eigenvalues)
    eigenvectors = eigenvectors[:, sorted_indices]
    eigenvalues = eigenvalues[sorted_indices]

    # Step 5: Select the first q eigenvectors
    gamma_q = eigenvectors[:, :q]

    # Step 6: Compute the principal components Z
    Z = X_centered @ gamma_q

    # Return the principal components matrix Z
    return Z
```

```python

# ==============
# Implementation
# ==============

# Example usage
X = np.random.randn(100, 5)  # Generate random data (100 samples, 5 features)
q = 3  # Number of principal components to return

# Perform PCA and obtain the first q principal components
Z = PCA(X, q)

# Print the resulting matrix of principal components Z
print("Principal Components (Z):")
print("==========================\n")
print(Z)
```

Listing 1: PCA Python Implementation