Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in machine learning and data analysis. It's a powerful tool for simplifying complex datasets, identifying patterns, and visualizing high-dimensional data.

What is PCA?
PCA is a linear transformation technique that reduces the dimensionality of a dataset by projecting it onto a new set of orthogonal axes, called principal components. These principal components are chosen such that they capture the maximum amount of variance in the data.

Logic Behind PCA
The logic behind PCA is based on the following ideas:

1. Variance: PCA aims to capture the maximum amount of variance in the data. Variance is a measure of how spread out the data is.
2. Orthogonality: PCA projects the data onto a new set of orthogonal axes, which means that the new axes are perpendicular to each other.
3. Linear Transformation: PCA is a linear transformation technique, which means that it uses linear combinations of the original variables to create the new principal components.

Math Behind PCA
Let's dive into the math behind PCA!

Given a dataset X with n samples and p features, we can represent it as a matrix:

$X = [x_1, x_2, ..., x_n]^T$

where x_i is the i-th sample and ^{T} denotes the transpose.

The goal of PCA is to find a new set of orthogonal axes, called principal components, that capture the maximum amount of variance in the data.

To do this, we need to find the eigenvectors and eigenvalues of the covariance matrix Σ of the data:

$\Sigma = (1/n) * X^T X$

The eigenvectors of Σ represent the directions of the new principal components, while the eigenvalues represent the amount of variance captured by each component.

We can sort the eigenvectors and eigenvalues in descending order of the eigenvalues, and select the top k eigenvectors to form the new principal components.

The new principal components can be represented as:

$Y = X * W$

where W is the matrix of eigenvectors and Y is the transformed data.

Applying PCA
To apply PCA, you can follow these steps:

1. Standardize the data: Standardize the data by subtracting the mean and dividing by the standard deviation for each feature.
2. Compute the covariance matrix: Compute the covariance matrix Σ of the standardized data.
3. Compute the eigenvectors and eigenvalues: Compute the eigenvectors and eigenvalues of the covariance matrix Σ.
4. Select the top k eigenvectors: Select the top k eigenvectors to form the new principal components.
5. Transform the data: Transform the original data using the selected eigenvectors.

Example in Python
Here's an example of applying PCA using Python and the scikit-learn library:

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()
X = iris.data

# Standardize the data
X_std = (X - X.mean(axis=0)) / X.std(axis=0)

# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_std)
```

This code applies PCA to the iris dataset and reduces the dimensionality from 4 features to 2 features. The explained_variance_ratio_ attribute of the PCA object shows the proportion of variance explained by each principal component.