Practice Exercise 2: Column Normalization

Problem: Normalize Matrix Columns

```
import numpy as np

# Problem: Normalize each column of a matrix to range [0, 1]
# Formula: (X - min) / (max - min)

X = np.array([[1, 2],
              [3, 4],
              [5, 6]])

print("Original matrix:")
print(X)
print("\nColumn mins:", X.min(axis=0))
print("Column maxs:", X.max(axis=0))
```

Solution: Vectorized Normalization

```
def normalize_columns(X):
    min_vals = X.min(axis=0)
    max_vals = X.max(axis=0)
    return (X - min_vals) / (max_vals - min_vals)

normalized = normalize_columns(X)
print("\nNormalized matrix:")
print(normalized)

# Verify normalization
print("\nVerification:")
print("Column mins:", normalized.min(axis=0))
print("Column maxs:", normalized.max(axis=0))
```

Row-wise Normalization

```
def normalize_rows(X):
    """Normalize each row to [0, 1]"""
    min_vals = X.min(axis=1, keepdims=True)
    max_vals = X.max(axis=1, keepdims=True)
    return (X - min_vals) / (max_vals - min_vals)

X_row_norm = normalize_rows(X)
print("Row-normalized matrix:")
print(X_row_norm)

print("\nRow mins:", X_row_norm.min(axis=1))
print("Row maxs:", X_row_norm.max(axis=1))
```

Z-score Normalization

```
def standardize_columns(X):
    """Standardize to mean=0, std=1"""
    mean_vals = X.mean(axis=0)
    std_vals = X.std(axis=0)
    return (X - mean_vals) / std_vals
```

```
standardized = standardize_columns(X)
print("Standardized matrix:")
print(standardized)

print("\nColumn means:", standardized.mean(axis=0))
print("Column stds:", standardized.std(axis=0))
```

Robust Normalization (Handles Outliers)

```
def robust_normalize(X):
    """Use median and IQR instead of mean and std"""
    median = np.median(X, axis=0)
    q75 = np.percentile(X, 75, axis=0)
    q25 = np.percentile(X, 25, axis=0)
    iqr = q75 - q25
    return (X - median) / iqr

# Add some outliers
X_outliers = np.array([[1, 2],
                       [3, 4],
                       [5, 6],
                       [100, 8]])  # Outlier in first column

robust_norm = robust_normalize(X_outliers)
print("Robust normalized (with outlier):")
print(robust_norm)

# Compare with standard normalization
std_norm = standardize_columns(X_outliers)
print("\nStandard normalized (affected by outlier):")
print(std_norm)
```