Normal Standardization

Given a matrix $X \in \mathbb{R}^{m \times n}$, do the following:

Transform X to a matrix Y of the same size so that each column of Y has a sample mean of zero and the unit standard deviation.

How to Calculate

For each column j of matrix X, perform the following steps:

Step 1: Calculate the sample mean of column j:

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} X_{i,j}$$

Step 2: Calculate the sample standard deviation of column j:

$$\sigma_j = \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (X_{i,j} - \mu_j)^2}$$

Step 3: Standardize each element in column j:

$$Y_{i,j} = \frac{X_{i,j} - \mu_j}{\sigma_j}$$

After this transformation, each column of Y will have:

- Mean: $\frac{1}{m} \sum_{i=1}^{m} Y_{i,j} = 0$
- Standard deviation: $\sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (Y_{i,j})^2} = 1$

Input Format

- A 2d numpy array of size $m \times n$ containing real numbers representing matrix X

Output Format

Output the resulting $m \times n$ matrix Y where each column has a sample mean of zero and unit standard deviation.

Constraints

- $X \in \mathbb{R}^{m \times n}$
- $1 \leq m, n \leq 1000$
- All input values are real numbers in the range $[-1000, 1000]$
- Each column of X has at least 2 distinct values (to ensure non-zero standard deviation)

Sample Input

```
X = [[1.0, 2.0, 3.0],
     [4.0, 5.0, 6.0],
     [7.0, 8.0, 9.0]]
```

Sample Output

```
[[-1.22474487, -1.22474487, -1.22474487],
 [ 0.        ,  0.        ,  0.        ],
 [ 1.22474487,  1.22474487,  1.22474487]]
```

Implementation

Goal: Fill in the following function:

```
def column_standardization(X):
    ...
    return ... # Return the standardized matrix
exec("\n".join(iter(input, "#Exit"))) # Don't remove this line
```