Concepcion, Ryan O.
2ECE-A ECE2112 9/3/2024

Programming Assignment #2

NORMALIZATION PROBLEM: Normalization is one of the most basic preprocessing techniques in data analytics. This involves centering and scaling process. Centering means subtracting the data from the mean and scaling means dividing with its standard deviation. Mathematically, normalization can be expressed as:

$$Z = \frac{X - \bar{x}}{\sigma}$$

Solution:

In [4]:
```
import numpy as np

#Generates 5x5 matrix filled with random float numbers between 0 and 1.
#Each element in the matrix is randomly chosen from a uniform distribution.
X = np.random.rand(5, 5)

#Print random 5x5 array
print("Random 5x5 Array")
print(X)

#Create variables that calculates the mean and standard deviation.
X_mean = X.mean()
X_std = X.std()

#Normalize by subtracting the mean from each element and dividing by the standard deviation, resulting in X_normalized.
X_normalized = (X - X_mean) / X_std
print("Normalized")
print(X_normalized)

#Save the normalized array as X_normalized.npy
#Allows you to store the array and load it later without recalculating or regenerating the data.
np.save('X_normalized.npy', X_normalized)
```

```
Random 5x5 Array
[[0.38091187 0.48938746 0.96763568 0.46620257 0.94388637]
 [0.50993217 0.68868718 0.17701088 0.92808885 0.65975692]
 [0.42006992 0.35819386 0.17555785 0.37906374 0.295499  ]
 [0.47883256 0.92480671 0.23809706 0.19757525 0.92598166]
 [0.55393675 0.50290627 0.20230583 0.47661023 0.14197483]]
Normalized
[[-0.45261909 -0.03795506  1.79021968 -0.12658276  1.69943441]
 [ 0.04058012  0.72389768 -1.23206088  1.63904604  0.61330746]
 [-0.30293165 -0.53946207 -1.23761531 -0.45968383 -0.77912247]
 [-0.07830273  1.62649958 -0.99854988 -1.15345052  1.63099102]
 [ 0.20879418  0.0137226  -1.13536719 -0.08679793 -1.36599139]]
```

DIVISIBLE BY 3 PROBLEM: Create the following 10 x 10 ndarray.

$$A = \begin{bmatrix} 1 & 4 & \cdots & 81 & 100 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 8281 & 8464 & \cdots & 9801 & 10000 \end{bmatrix}$$

which are the squares of the first 100 positive integers.

From this ndarray, determine all the elements that are divisible by 3. Save the result as *div_by_3.npy*

Solution:

In [5]:
```
#Create a 10x10 array from 1 to 100 , and exponentiate each number to 2.
#Reshapes the array into a 10x10 matrix.
A = np.arange(1, 101).reshape(10, 10) ** 2

#Selects all elements from the matrix that are divisible by 3.
#Creates a new array which contains only those elements for which the remainder when divided by 3 is zero.
div_by_3 = A[A % 3 == 0]

#Print the 10x10 matrix from 1 to 100, and also print the list of elements divisible by 3 from that matrix.
print("Array is: ")
print(A)
print("Number of elements divisible by 3:")
print(div_by_3)


#Save the result as div_by_3.npy, and load it without regenerating the data.
np.save('div_by_3.npy', div_by_3)
```

```
Array is:
[[    1     4     9    16    25    36    49    64    81   100]
 [  121   144   169   196   225   256   289   324   361   400]
 [  441   484   529   576   625   676   729   784   841   900]
 [  961  1024  1089  1156  1225  1296  1369  1444  1521  1600]
 [ 1681  1764  1849  1936  2025  2116  2209  2304  2401  2500]
 [ 2601  2704  2809  2916  3025  3136  3249  3364  3481  3600]
 [ 3721  3844  3969  4096  4225  4356  4489  4624  4761  4900]
 [ 5041  5184  5329  5476  5625  5776  5929  6084  6241  6400]
 [ 6561  6724  6889  7056  7225  7396  7569  7744  7921  8100]
 [ 8281  8464  8649  8836  9025  9216  9409  9604  9801 10000]]
Number of elements divisible by 3:
[   9   36   81  144  225  324  441  576  729  900 1089 1296 1521 1764
 2025 2304 2601 2916 3249 3600 3969 4356 4761 5184 5625 6084 6561 7056
 7569 8100 8649 9216 9801]
```