DAAN 862: Assignment 2

Name: Jerum Hubbert **User ID:** jlh7781
Date: January 25, 2026

Question 1

Perform the following actions:

- Use the `randn` function to create an array with a dimension of **5x5**.
- Use a `for` loop to calculate the **sum of all elements in the diagonal** of the array. *(25 points)*
- Choose any **three functions** to apply to this array. *(25 points)*

In [2]:
```
import numpy as np

# 1. Create a 5x5 array using the randn function
# This generates numbers from a standard normal distribution (mean=0, stdev=
np.random.seed(42) # Optional: ensures you get the same numbers every time y
arr_1 = np.random.randn(5, 5)

print("--- 5x5 Array ---")
print(arr_1)

# 2. Use a for loop to calculate the sum of all elements in the diagonal
# Note: We access the diagonal where the row index (i) equals the column inc
diagonal_sum = 0
for i in range(5):
    diagonal_sum += arr_1[i, i]

print(f"\nSum of the diagonal elements = {diagonal_sum}")

# 3. Choose any three functions to apply to this array
# We'll use Mean, Standard Deviation, and Sum of the entire array
overall_mean = np.mean(arr_1)
overall_std  = np.std(arr_1)
overall_min  = np.min(arr_1)

print("\n--- Additional Functions Applied ---")
print(f"1. Mean = {overall_mean}")
print(f"2. Standard Deviation = {overall_std}")
print(f"3. Minimum Value in Array = {overall_min}")
```

```
--- 5x5 Array ---
[[ 0.49671415 -0.1382643   0.64768854  1.52302986 -0.23415337]
 [-0.23413696  1.57921282  0.76743473 -0.46947439  0.54256004]
 [-0.46341769 -0.46572975  0.24196227 -1.91328024 -1.72491783]
 [-0.56228753 -1.01283112  0.31424733 -0.90802408 -1.4123037 ]
 [ 1.46564877 -0.2257763   0.0675282  -1.42474819 -0.54438272]]

Sum of the diagonal elements = 0.865482440038265

--- Additional Functions Applied ---
1. Mean = -0.16350805865809107
2. Standard Deviation = 0.9372267579693156
3. Minimum Value in Array = -1.913280244657798
```

Question 2

Perform the following actions:

- Use `x = np.random.randint(0, 1000, size = (10, 10))` to generate a **10x10 array**.
- Use a `for` loop to find out **how many even numbers** are in it. *(25 points)*
- Randomly generate an **8x9 array** from a normal distribution with `mean = 1` and `sigma = 0.5`.
- Calculate the **mean of elements** whose indexes have a relation of `(i+j)%5 == 0` (where `i` is the row index and `j` is the column index). *(25 points)*

In [3]:
```python
import numpy as np

# --- Part 1: Counting Even Numbers ---
# Generate a 10x10 array of random integers between 0 and 1000
x = np.random.randint(0, 1000, size=(10, 10))

even_count = 0

# Use a nested for loop to iterate through rows and columns
for row in x:
    for value in row:
        if value % 2 == 0:
            even_count += 1

print("--- 10x10 Integer Array Analysis ---")
print(f"Number of even integers found: {even_count}")


# --- Part 2: Normal Distribution and Index Logic ---
# Generate 8x9 array from normal distribution (mean=1, sigma=0.5)
# 'loc' is mean, 'scale' is sigma
arr_norm = np.random.normal(loc=1, scale=0.5, size=(8, 9))

# List to store elements that meet the index condition (i+j)%5 == 0
special_elements = []

rows, cols = arr_norm.shape
```

```python

# Iterate through indices to check the condition
for i in range(rows):
    for j in range(cols):
        if (i + j) % 5 == 0:
            special_elements.append(arr_norm[i, j])

# Calculate the mean of those specific elements
if len(special_elements) > 0:
    special_mean = np.mean(special_elements)
else:
    special_mean = 0

print("\n--- 8x9 Normal Distribution Analysis ---")
print(f"Condition: (i+j) % 5 == 0")
print(f"Number of elements matching condition = {len(special_elements)}")
print(f"Mean of those elements = {special_mean}")
```

```
--- 10x10 Integer Array Analysis ---
Number of even integers found: 47

--- 8x9 Normal Distribution Analysis ---
Condition: (i+j) % 5 == 0
Number of elements matching condition = 14
Mean of those elements = 0.9116382596726015
```