**1. Transpose Matrix**

**Question:**  
Given a matrix, return its transpose. The transpose of a matrix flips it over its main diagonal so that the row becomes the column and vice versa.

**Sample Input:**

matrix = [[1, 2, 3],  
 [4, 5, 6]]

**Expected Output:**

[[1, 4],  
 [2, 5],  
 [3, 6]]

**Solution:**

def transpose(matrix):  
 rows = len(matrix)  
 cols = len(matrix[0])  
 result = []  
 for j in range(cols):  
 new\_row = []  
 for i in range(rows):  
 new\_row.append(matrix[i][j])  
 result.append(new\_row)  
 return result  
  
# Example run:  
matrix = [[1, 2, 3], [4, 5, 6]]  
print("Transposed Matrix:", transpose(matrix))

**2. Matrix Addition**

**Question:**  
Given two matrices A and B of the same dimensions, return a new matrix which is the element-wise sum of A and B.

**Sample Input:**

A = [[1, 2],  
 [3, 4]]  
B = [[5, 6],  
 [7, 8]]

**Expected Output:**

[[6, 8],  
 [10, 12]]

**Solution:**

def matrix\_add(A, B):  
 result = []  
 for i in range(len(A)):  
 new\_row = []  
 for j in range(len(A[0])):  
 new\_row.append(A[i][j] + B[i][j])  
 result.append(new\_row)  
 return result  
  
# Example run:  
A = [[1, 2], [3, 4]]  
B = [[5, 6], [7, 8]]  
print("Sum of matrices:", matrix\_add(A, B))

**3. Scalar Multiplication**

**Question:**  
Given a matrix and a scalar number, return a matrix where each element is multiplied by the scalar.

**Sample Input:**

matrix = [[1, 2],  
 [3, 4]]  
scalar = 3

**Expected Output:**

[[3, 6],  
 [9, 12]]

**Solution:**

def scalar\_multiply(matrix, scalar):  
 result = []  
 for i in range(len(matrix)):  
 new\_row = []  
 for j in range(len(matrix[0])):  
 new\_row.append(matrix[i][j] \* scalar)  
 result.append(new\_row)  
 return result  
  
# Example run:  
matrix = [[1, 2], [3, 4]]  
print("Scalar multiplied matrix:", scalar\_multiply(matrix, 3))

**4. Trace of a Matrix**

**Question:**  
Given a square matrix, return the sum of the elements on its main diagonal.

**Sample Input:**

matrix = [[1, 2, 3],  
 [4, 5, 6],  
 [7, 8, 9]]

**Expected Output:**

15 # (1 + 5 + 9)

**Solution:**

def is\_square(matrix):  
 if len(matrix) == len(matrix[0]):  
 return True  
 else:  
 return False  
  
def trace(matrix):  
 if not is\_square(matrix):  
 return None  
 total = 0  
 for i in range(len(matrix)):  
 total += matrix[i][i]  
 return total  
  
# Example run:  
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
print("Trace of matrix:", trace(matrix))

**5. Check if a Matrix is Square**

**Question:**  
Given a matrix, determine if it is square. A square matrix has the same number of rows and columns.

**Sample Input 1:**

matrix = [[1, 2],  
 [3, 4]]

**Expected Output 1:**

True

**Sample Input 2:**

matrix = [[1, 2, 3],  
 [4, 5, 6]]

**Expected Output 2:**

False

**Solution:**

def is\_square(matrix):  
 if len(matrix) == len(matrix[0]):  
 return True  
 else:  
 return False  
  
# Example runs:  
square\_matrix = [[1, 2], [3, 4]]  
non\_square\_matrix = [[1, 2, 3], [4, 5, 6]]  
print("Is square\_matrix square?", is\_square(square\_matrix))  
print("Is non\_square\_matrix square?", is\_square(non\_square\_matrix))

Each question illustrates the problem clearly, contains input and expected output examples, and provides a simple loop-based program solution suited for coding practice and interviews.

Summary table of relevant LeetCode program numbers:

| **Operation** | **LeetCode Problem Number** | **Notes** |
| --- | --- | --- |
| Transpose Matrix | #867 | Exact problem on LeetCode |
| Matrix Addition | None | Basic operation, no direct problem |
| Scalar Multiplication | None | Basic operation, no direct problem |
| Trace of a Matrix | None | Basic matrix operation, no problem found |
| Check if Square | None | Basic check, no specific problem on LeetCode |