In [1]:

```
#Python program to add two Matrices
# Function to add two matrices
def add_matrices(mat1, mat2):
    if len(mat1) != len(mat2) or len(mat1[0]) != len(mat2[0]):
        raise ValueError("Matrices must have the same dimensions for addition.")

    result = []
    for i in range(len(mat1)):
        row = []
        for j in range(len(mat1[0])):
            row.append(mat1[i][j] + mat2[i][j])
        result.append(row)

    return result

# Example matrices
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matrix2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]

# Call the add_matrices function to add the matrices
result_matrix = add_matrices(matrix1, matrix2)

# Display the result
for row in result_matrix:
    print(row)
```

```
[10, 10, 10]
[10, 10, 10]
[10, 10, 10]
```

In [ ]:

In [2]:

```
#Python program to multiply two matrices
# Function to multiply two matrices
def multiply_matrices(mat1, mat2):
    if len(mat1[0]) != len(mat2):
        raise ValueError("Number of columns in the first matrix must be equal to th

    result = []
    for i in range(len(mat1)):
        row = []
        for j in range(len(mat2[0])):
            sum = 0
            for k in range(len(mat2)):
                sum += mat1[i][k] * mat2[k][j]
            row.append(sum)
        result.append(row)

    return result

# Example matrices
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matrix2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]

# Call the multiply_matrices function to multiply the matrices
result_matrix = multiply_matrices(matrix1, matrix2)

# Display the result
for row in result_matrix:
    print(row)
```

```
[30, 24, 18]
[84, 69, 54]
[138, 114, 90]
```

In [ ]:

In [3]:

```
#Python program for Matrix Product
import numpy as np

# Define the matrices
matrix1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matrix2 = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

# Calculate the matrix product
result_matrix = np.dot(matrix1, matrix2)

# Display the result
print("Matrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("\nMatrix Product:")
print(result_matrix)
```

```
Matrix 1:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Matrix 2:
[[9 8 7]
 [6 5 4]
 [3 2 1]]

Matrix Product:
[[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]
```

In [ ]:

In [4]:

```
#Adding and Subtracting Matrices in Python
# Function to add two matrices
def add_matrices(mat1, mat2):
    if len(mat1) != len(mat2) or len(mat1[0]) != len(mat2[0]):
        raise ValueError("Matrices must have the same dimensions for addition.")

    result = []
    for i in range(len(mat1)):
        row = []
        for j in range(len(mat1[0])):
            row.append(mat1[i][j] + mat2[i][j])
        result.append(row)

    return result

# Function to subtract two matrices
def subtract_matrices(mat1, mat2):
    if len(mat1) != len(mat2) or len(mat1[0]) != len(mat2[0]):
        raise ValueError("Matrices must have the same dimensions for subtraction.")

    result = []
    for i in range(len(mat1)):
```

```python
        row = []
        for j in range(len(mat1[0])):
            row.append(mat1[i][j] - mat2[i][j])
        result.append(row)

    return result

# Example matrices
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matrix2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]

# Call the add_matrices function to add the matrices
result_addition = add_matrices(matrix1, matrix2)

# Call the subtract_matrices function to subtract the matrices
result_subtraction = subtract_matrices(matrix1, matrix2)

# Display the results
print("Matrix 1:")
for row in matrix1:
    print(row)
print("\nMatrix 2:")
for row in matrix2:
    print(row)
print("\nMatrix Addition:")
for row in result_addition:
    print(row)
print("\nMatrix Subtraction:")
for row in result_subtraction:
    print(row)
```

```
Matrix 1:
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]

Matrix 2:
[9, 8, 7]
[6, 5, 4]
[3, 2, 1]

Matrix Addition:
[10, 10, 10]
[10, 10, 10]
[10, 10, 10]

Matrix Subtraction:
[-8, -6, -4]
[-2, 0, 2]
[4, 6, 8]
```

In [ ]:

In [5]:

```python
#Transpose a matrix in Single line in Python
import numpy as np

# Example matrix
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Transpose the matrix in a single line
transposed_matrix = np.transpose(matrix)

print("Original matrix:")
print(matrix)
print("\nTransposed matrix:")
print(transposed_matrix)
```

```
Original matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Transposed matrix:
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

In [ ]:

In [6]:

```
#Python | Matrix creation of n*n
n = 3  # Change this to your desired matrix size (e.g., 3 for a 3x3 matrix)

# Create an n x n matrix filled with zeros using list comprehension
matrix = [[0 for _ in range(n)] for _ in range(n)]

# Print the matrix
for row in matrix:
    print(row)
```

```
[0, 0, 0]
[0, 0, 0]
[0, 0, 0]
```

In [ ]:

In [7]:

```
#Python | Get Kth Column of Matrix
# Define a matrix (example)
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Specify the column number you want to extract (1-based index)
k = 2  # Change this to the desired column number

# Use list comprehension to extract the Kth column
kth_column = [row[k - 1] for row in matrix]

# Print the extracted column
print(f"Kth column ({k}th column):", kth_column)
```

```
Kth column (2th column): [2, 5, 8]
```

In [ ]:

In [8]:

```
#Python – Vertical Concatenation in Matrix
import numpy as np

# Example matrices
matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
matrix2 = np.array([[7, 8, 9], [10, 11, 12]])

# Vertically concatenate the matrices using np.vstack()
concatenated_matrix = np.vstack((matrix1, matrix2))

# Print the concatenated matrix
```

```
print("Matrix 1:")
print(matrix1)

print("\nMatrix 2:")
print(matrix2)

print("\nVertically concatenated matrix:")
print(concatenated_matrix)
```

```
Matrix 1:
[[1 2 3]
 [4 5 6]]

Matrix 2:
[[ 7  8  9]
 [10 11 12]]

Vertically concatenated matrix:
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

In [ ]:

In [9]:

```
#Python program to check if a string is palindrome or not
def is_palindrome(input_string):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    cleaned_string = input_string.replace(" ", "").lower()
    # Check if the cleaned string is equal to its reverse
    return cleaned_string == cleaned_string[::-1]

# Input string
input_string = "A man a plan a canal Panama"

# Check if the input string is a palindrome
if is_palindrome(input_string):
    print(f"'{input_string}' is a palindrome.")
else:
    print(f"'{input_string}' is not a palindrome.")
```

```
'A man a plan a canal Panama' is a palindrome.
```

In [ ]:

In [10]:

```
#Python program to check whether the string is Symmetrical or Palindrome
def is_symmetrical(input_string):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    cleaned_string = input_string.replace(" ", "").lower()
    # Check if the cleaned string is equal to its reverse
    return cleaned_string == cleaned_string[::-1]

# Input string
input_string = "A man a plan a canal Panama"

# Check if the input string is symmetrical and a palindrome
if is_symmetrical(input_string):
    print(f"'{input_string}' is both symmetrical and a palindrome.")
else:
    print(f"'{input_string}' is not both symmetrical and a palindrome.")
```

```
'A man a plan a canal Panama' is both symmetrical and a palindrome.
```

In [ ]:

```
```