1. **Python program to add two Matrices**

import numpy as np

  X = [[1,2,3],

    [4 ,5,6],

    [7 ,8,9]]

  Y = [[9,8,7],

    [6,5,4],

    [3,2,1]]

 result = np.array(X) + np.array(Y)

 print(result)

**Output:** [[10 10 10]

[10 10 10]

[10 10 10]]

1. **Python program to multiply two matrices**

# take a 3x3 matrix

A = [[12, 7, 3],

  [4, 5, 6],

  [7, 8, 9]]

# take a 3x4 matrix

B = [[5, 8, 1, 2],

  [6, 7, 3, 0],

  [4, 5, 9, 1]]

result = [[0, 0, 0, 0],

    [0, 0, 0, 0],

    [0, 0, 0, 0]]

# iterating by row of A

for i in range(len(A)):

  # iterating by column by B

  for j in range(len(B[0])):

    # iterating by rows of B

    for k in range(len(B)):

      result[i][j] += A[i][k] \* B[k][j]

for r in result: print(r)

**Output:** [114, 160, 60, 27]

[74, 97, 73, 14]

[119, 157, 112, 23]

1. **Python program for Matrix Product**

# initializing list

test\_list = [[1, 4, 5], [7, 3], [4], [46, 7, 3]]

# printing original list

print("The original list : " + str(test\_list))

res = 1

for i in test\_list:

  for j in i:

    res \*= j

# print result

print("The total element product in lists is : " + str(res))

**Output:** The original list: [[1, 4, 5], [7, 3], [4], [46, 7, 3]]

The total element product in lists is: 1622880

1. **Adding and Subtracting Matrices in Python**

# importing numpy as np

import numpy as np

# creating first matrix

A = np.array([[1, 2], [3, 4]])

# creating second matrix

B = np.array([[4, 5], [6, 7]])

print("Printing elements of first matrix")

print(A)

print("Printing elements of second matrix")

print(B)

# subtracting two matrix

print("Subtraction of two matrix")

print(np.subtract(A, B))

**Output:** Printing elements of first matrix

[[1 2]

[3 4]]

Printing elements of second matrix

[[4 5]

[6 7]]

Subtraction of two matrix

[[-3 -3]

[-3 -3]]

1. **Transpose a matrix in Single line in Python**

import numpy

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

print(matrix)

print("\n")

print(numpy.transpose(matrix))

**output:** [[1, 2, 3], [4, 5, 6]]

[[1 4]

[2 5]

[3 6]]

1. **Python | Matrix creation of n\*n**

import itertools

# initializing N

N = 4

# printing dimension

print("The dimension : " + str(N))

# using next() + itertools.count()

# matrix creation of n \* n

temp = itertools.count(1)

res = [[next(temp) for i in range(N)] for i in range(N)]

# print result

print("The created matrix of N \* N: " + str(res))

**Output:** The dimension : 4

The created matrix of N \* N: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]

1. **Python | Get Kth Column of Matrix**

# initialize list

test\_list = [[4, 5, 6], [8, 1, 10], [7, 12, 5]]

# printing original list

print("The original list is : " + str(test\_list))

# initialize K

K = 2

# Get Kth Column of Matrix

# using list comprehension

res = [sub[K] for sub in test\_list]

# printing result

print("The Kth column of matrix is : " + str(res))

**Output:** The original list is : [[4, 5, 6], [8, 1, 10], [7, 12, 5]]

The Kth column of matrix is : [6, 10, 5]

1. **Python – Vertical Concatenation in Matrix**

# Python3 code to demonstrate working of

# Vertical Concatenation in Matrix

# Using join() + list comprehension + zip\_longest()

from itertools import zip\_longest

# initializing lists

test\_list = [["Gfg", "good"], ["is", "for"], ["Best"]]

# printing original list

print("The original list : " + str(test\_list))

# using join to concaternate, zip\_longest filling values using

# "fill"

res = ["".join(ele) for ele in zip\_longest(\*test\_list, fillvalue ="")]

# printing result

print("List after column Concatenation : " + str(res))

**Output:** The original list : [['Gfg', 'good'], ['is', 'for'], ['Best']]

List after column Concatenation : ['GfgisBest', 'goodfor']

1. **Python program to check if a string is palindrome or not**

# function which return reverse of a string

def isPalindrome(s):

  return s == s[::-1]

# Driver code

s = "malayalam"

ans = isPalindrome(s)

if ans:

  print("Yes")

else:

  print("No")

**Output:** Yes

1. **Python program to check whether the string is Symmetrical or Palindrome**

string = 'amaama'

half = int(len(string) / 2)

if len(string) % 2 == 0: # even

  first\_str = string[:half]

  second\_str = string[half:]

else: # odd

  first\_str = string[:half]

  second\_str = string[half+1:]

# symmetric

if first\_str == second\_str:

  print(string, 'string is symmetrical')

else:

  print(string, 'string is not symmetrical')

# palindrome

if first\_str == second\_str[::-1]: # ''.join(reversed(second\_str)) [slower]

  print(string, 'string is palindrome')

else:

  print(string, 'string is not palindrome')

**output:** amaama string is symmetrical

amaama string is palindrome