

**Experiment No.: 3**

**Aim**

To implement Matrix operations (using vectorization), transformation using python and SVD using Python

**Questions**

(a) Matrix operations (using vectorization),

(b) transformation using python and

(c) SVD using Python.

**Program and Output**

1. 1. import numpy as np

a = np.array([1, 2, 3])

print("type: %s" %type(a))

print("shape: %s" %a.shape)

print(a[0], a[1], a[2])

a[0] = 5

print(a)

b = np.array([[1,2,3],[4,5,6]])

print("\n shape of b:",b.shape)

print(b[0, 0], b[0, 1], b[1, 0]

a = np.zeros((2,2))

print("All zeros matrix:\n  %s" %a)

b = np.ones((1,2))

print("\nAll ones matrix:\n  %s" %b)

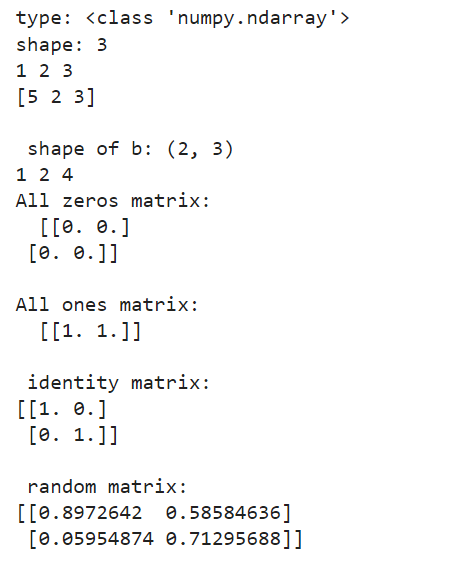
d = np.eye(2)

print("\n identity matrix: \n%s"%d)

e = np.random.random((2,2))

print("\n random matrix: \n%s"%e)

Output



2. #vectorized sum

print("Vectorized sum example\n")

x = np.array([[1,2],[3,4]])

print("x:\n %s" %x)

print("sum: %s"%np.sum(x))

print("sum axis = 0: %s" %np.sum(x, axis=0))

print(" sum axis = 1: %s" %np.sum(x, axis=1))

#matrix dot product

a = np.arange(10000)

b = np.arange(10000)

dp = np.dot(a,b)

print("Dot product: %s\n" %dp)

#outer product

op = np.outer(a,b)

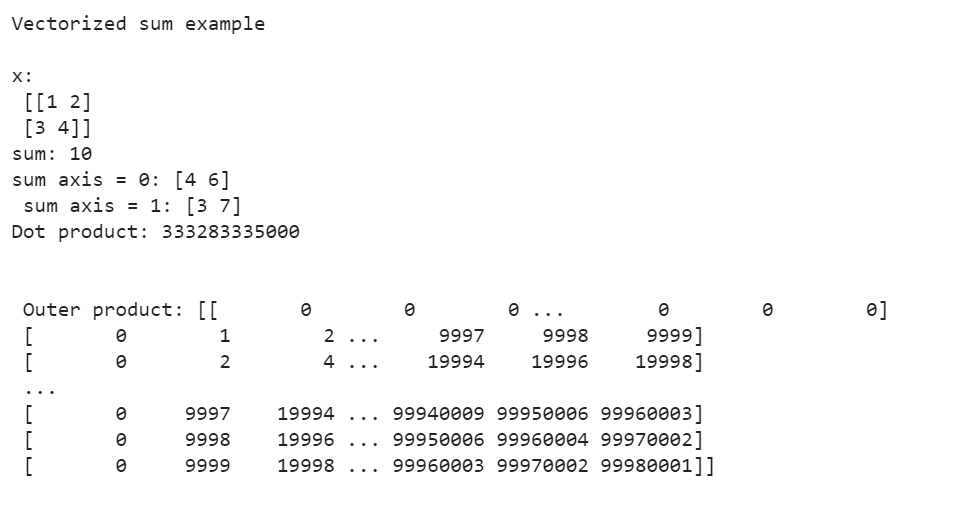
print("\n Outer product: %s\n" %op)

#elementwise product

ep = np.multiply(a, b)

print("\n Element Wise product: %s \n" %ep)

Output



1. #transformation using python

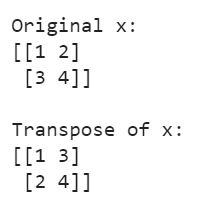
import numpy as np

x = np.array([[1,2], [3,4]])

print("Original x: \n%s " %x)

print("\nTranspose of x: \n%s" %x.T)

Output



1. # Singular-value decomposition

from numpy import array

from scipy.linalg import svd

# define a matrix

A = array([[1, 2], [3, 4], [5, 6]])

print("A: \n%s" %A)

# SVD

U, s, VT = svd(A)

print("\nU: \n%s" %U)

print("\ns: \n %s" %s)

print("\nV^T: \n %s" %VT)

output

