# CYCLE 2 – PART 2

1. Create a square matrix with random integer values(use randint()) and use appropriate functions to find:

i) inverse

ii) rank of matrix

iii) Determinant

iv) transform matrix into 1D array

v) eigen values and vectors

**Code:**

import numpy as np

print("Matrix:")

ar = np.random.randint(10, size=(3, 3))

print(ar)

print("\n Inverse Matrix:")

print(np.linalg.inv(ar))

print("\n Rank of Matrix:")

print(np.linalg.matrix\_rank(ar))

print("\n Determinant:")

print(np.linalg.det(ar))

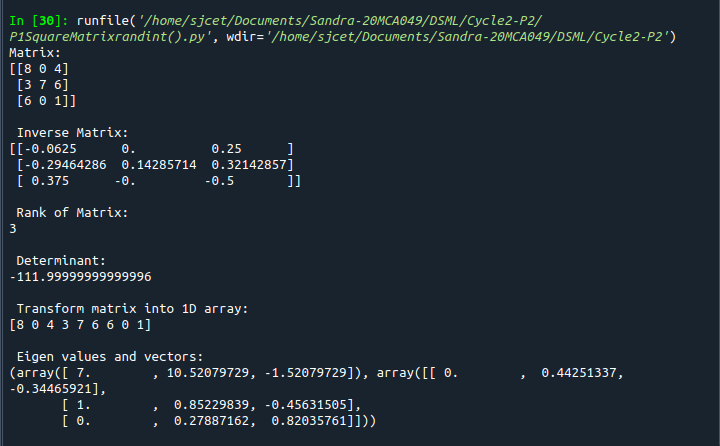
print("\n Transform matrix into 1D array:")

print(np.ravel(ar))

print("\n Eigen values and vectors:")

print(np.linalg.eig(ar))

**Output:**



2. Create a matrix X with suitable rows and columns

i) Display the cube of each element of the matrix using different methods

(use multiply(), \*, power(),\*\*)

ii) Display identity matrix of the given square matrix.

iii) Display each element of the matrix to different powers.

iv) Create a matrix Y with same dimension as X and perform the operation X2+2Y

**Code:**

import numpy as np

x=np.arange(1,10).reshape(3,3)

print("Matrix:\n ",x)

print("Cube using multiply():\n",np.multiply(x,3))

print("Cube using \* :\n ",x\*x\*x)

print("Cube using power():\n",np.power(x,3))

print("Cube using \*\* :\n",x\*\*3)

print("Identity matrix of the given square matrix:")

print(np.identity(3,dtype= int))

print(" Display each element of the matrix to different powers.")

print(np.power(x,x))

print("Create a matrix Y with same dimension as X and perform the operation X2+2Y:",)

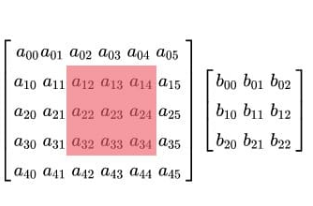
y = np.arange(11,20).reshape(3,3)

print(np.add((np.power(x,2)),(np.multiply(y,2))))

**Output:**



1. Multiply a matrix with a submatrix of another matrix and replace the same in larger matrix.



**Code:**

import numpy as np;

x = np.arange(1,31).reshape(5,6)

print("\n x-> big matrix :\n",x)

#x[column,row]

y=x[1:4,2:5]

#print("y-> sub matrix of x : \n",y)

z = np.arange(2,11).reshape(3,3)

#print("matrix z :\n",z)

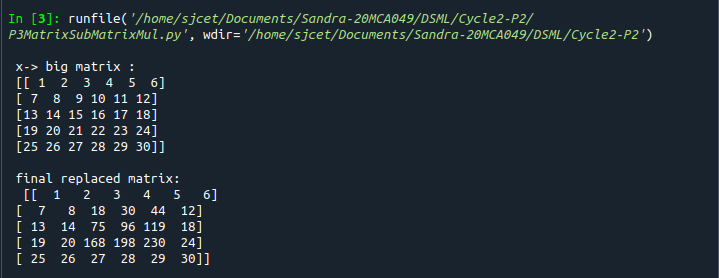
y = np.multiply(y,z)

#print("multiplied matrix : \n",y)

x[1:4,2:5] = y

print("\n final replaced matrix:\n ",x)

**Output:**



1. Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of the 3 matrices.

**Code:**

import numpy as np

A = ([1, 1, 1],[1 ,1, 1],[1, 1, 1])

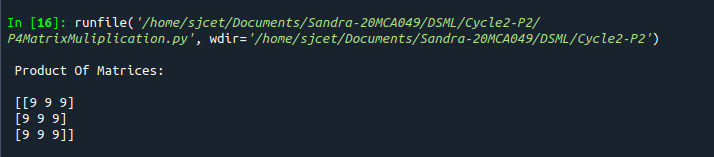
B = ([1, 1, 1],[1 ,1, 1],[1, 1, 1])

C = ([1, 1, 1],[1 ,1, 1],[1, 1, 1])

res = np.dot(A,np.dot(B,C))

print("\n Product Of Matrices: \n\n", res)

**Output:**



1. Write a program to check whether given matrix is symmetric or Skew Symmetric.

**Code:**

import numpy as np;

x = np.arange(1,10).reshape(3,3)

print("Matrix x: \n", x)

# Transposing the Matrix M

y=x.transpose();

print("Transpose of x :\n ",y)

# -1(Transpose)

skew=np.multiply(y,-1);

print("negative of transpose :\n ",skew)

if x.all() == y.all():

print("Matrix is symmetric")

else:

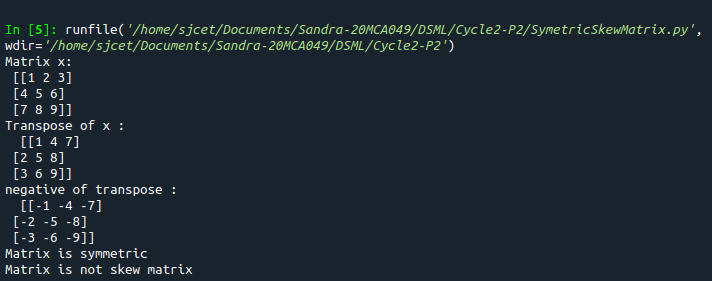
print("Matrix is not symmetric")

if x.all() == ~y.all():

print("Matrix is skew symmetric")

else:

print("Matrix is not skew matrix")

**Output:**

1. Write a program to find out the value of X using **solve(),** given **A** and **b** as above

**Code:**

import numpy as np;

A=np.array([[2,1,-2],[3,0,1],[1,1,-1]])

b=np.array([[-3],[5],[2]])

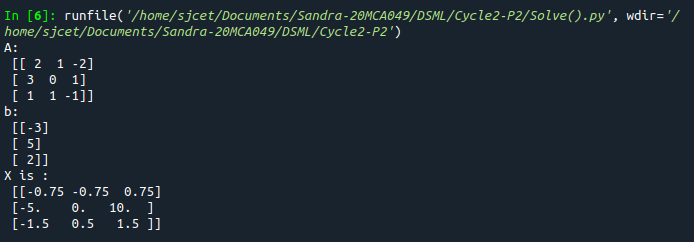
print("A:\n",A)

print("b:\n",b)

X=np.multiply((np.linalg.inv(A)),b)

print("X is : \n",X)

**Output:**



1. Write a program to perform the SVD of a given matrix. Also reconstruct the given matrix from the 3 matrices obtained after performing SVD.

**Code:**

import numpy as np;

A = np.array([[1, 2], [3, 4], [5, 6]])

print("Matrix:\n",A)

# SVD

U, s, VT = np.linalg.svd(A)

print("U:\n", U)

print("Sigma:\n",s)

print("V^[T:\n](../../../../../../T:/n)",VT)

**Output:**

