PROGRAM NO:2

AIM:Matrix operations (using vectorization) and transformation using python and SVD using Python

In [3]:

```
import numpy as np

list=[1,2,3,4]
np.array(list)

list2=[[5,6,7],[6,9,2],[4,1,0]]
np.array(list2)

np.arange(0,10)
np.arange(0,11,2)
np.zeros(3)
np.zeros((3,3))
np.ones(3)
np.ones((3,3))
np.eye(3)
np.linspace(0,3,6)
np.linspace(0,10,100)
np.random.rand(3)
np.random.rand(3,4)
np.random.randn(3)
np.random.randint(4,8,2)

l=np.arange(0,25)
l.reshape(5,5)
l.max()
l.min()
l.argmax()
l.argmin()
l.shape
l.reshape(25,1)
l.dtype

k=[[5,10,15],[20,25,30],[35,40,45]]
arr1=np.array(k)
arr1[1]
arr1[1:3]
arr1[1:2]=40
arr1[:]=40

arr1[1:3,0:2]

kk=arr1>20
arr1[kk]
```

Out[3]:

```
array([40, 40, 40, 40, 40, 40, 40, 40, 40])
```

In [4]:

```
import numpy as np
arr=np.arange(0,10)
arr+arr
```

Out[4]:

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

In [5]:

```
import numpy as np
np.zeros(10)
```

Out[5]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [6]:

```
v=np.arange(0,9)
v.reshape(3,3)
```

Out[6]:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [7]:

```
np.random.randn(1,50,25)
```

Out[7]:

```
array([[[ 0.18381789, -0.59662453,  1.69035425, ...,  0.79868756,
          1.92107277,  1.72989075],
        [ 1.09948134,  0.48546129, -0.85094803, ..., -0.36575627,
         -0.52300372, -1.50679128],
        [ 0.1306421 , -0.56734135,  0.49072883, ..., -0.78336048,
          1.51246162, -0.6503835 ],
        ...,
        [-1.18368046,  0.35059924, -1.12042168, ...,  0.41841656,
          0.8464394 ,  1.47906685],
        [ 0.2624534 ,  0.65425221,  0.98670929, ...,  0.13344227,
          0.33872611, -0.29160051],
        [ 2.02364386, -1.18765951,  1.72628144, ..., -1.46705916,
         -2.32152996,  1.23583662]]])
```

In [8]:

```
ar=np.arange(1,101)
d=ar.reshape(10,10)
d/100
```

Out[8]:

```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

In [ ]:

```
import numpy as np
from numpy.linalg import svd
```

In [ ]:

```
# this matrix has rank=2, since col3 = col1+co2,
## but col1 and col2 are independent from each other
```

```
A = np.array([[1,2,3], [4,5,6], [5,7,9]])

U, S, VT = svd(A)
```

In [ ]:

```
print("Left Singular Vectors:")
print(U)
print("Singular Values:")
print(np.diag(S))
print("Right Singular Vectors:")
print(VT)
```

```
Left Singular Vectors:
[[-0.2354116   0.78182354 -0.57735027]
 [-0.55937325 -0.5947842  -0.57735027]
 [-0.79478485  0.18703934  0.57735027]]
Singular Values:
[[1.56633231e+01 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 8.12593979e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.17807579e-15]]
Right Singular Vectors:
[[-0.41158755 -0.56381288 -0.71603821]
 [-0.8148184  -0.12429146  0.56623547]
 [-0.40824829  0.81649658 -0.40824829]]
```

In [ ]:

```
# Return the original matrix A
# @ is used for matrix multiplication in Py3, use np.matmul with Py2
print(U @ np.diag(S) @ VT)
```

```
[[1. 2. 3.]
 [4. 5. 6.]
 [5. 7. 9.]]
```

SVD

- **Sklearn Truncated SVD - This is used for dimensional reduction directly**

In [ ]:

```
import numpy as np
from sklearn.decomposition import TruncatedSVD
```

In [ ]:

```
A = np.array([[1,2,3], [4,5,6], [5,7,9]])
print("Original Matrix:")
A
```

```
Original Matrix:
```

Out[ ]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [5, 7, 9]])
```

In [ ]:

```
svd =  TruncatedSVD(n_components = 2)  # reduce to 2 features
A_transf = svd.fit_transform(A)

print("Singular values:")
print(svd.singular_values_)
print()

print("Transformed Matrix after reducing to 2 features:")
print(A_transf)
```

```
Singular values:
[15.66332312  0.81259398]

Transformed Matrix after reducing to 2 features:
[[ 3.68732795  0.6353051 ]
 [ 8.76164389 -0.48331806]
 [12.44897184  0.15198704]]
```

RESULT:Program executed sucessfully and output is obtained