In [1]:

```
import numpy as np
import random
```

In [ ]:

```

```

In [16]:

```
#Write a NumPy program to find the dot product of two arrays of different dimensions.
arr1=np.arange(1,9).reshape(4,2)
print(arr1)

arr2=np.arange(1,3)
print(arr2)

#dot product
print('dot product is:',np.dot(arr1,arr2))
```

```
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
[1 2]
dot product is: [ 5 11 17 23]
```

In [28]:

```
#Write a NumPy program to create a 3x3 identity matrix and stack it vertically and horizontally.

arr=np.identity(3)
print(arr)

#vertical stacking
vert=np.vstack((arr,arr,arr))
print('vertical stacking:\n',vert)

#horizontial stacking
hori=np.hstack((arr,arr,arr))
print('horizontial stacking:\n',hori)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
vertical stacking:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
horizontial stacking:
 [[1. 0. 0. 1. 0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 1. 0.]
 [0. 0. 1. 0. 0. 1. 0. 0. 1.]]
```

In [42]:

```
#Write a NumPy program to create a 4x4 array with random values and find the sum of each row.

arr=np.random.rand(4,4)
print('4X4 Array:\n',arr)

#Row Sum

row_sum=np.sum(arr,axis=1)
print('Sum of each row:\n',row_sum)
```

```
4X4 Array:
 [[0.19166284 0.65084096 0.22302248 0.628329  ]
 [0.29590978 0.29832877 0.65992853 0.03094882]
 [0.20744298 0.33675211 0.47403564 0.38904156]
 [0.52942704 0.27100724 0.98757    0.35378776]]
Sum of each row:
 [1.69385527 1.28511591 1.40727229 2.14179206]
```

In [58]:

```
#Write a NumPy program to create a 3x3 array with random values and subtract the mean of each row from each element.

arr=np.random.rand(3,3)
print('Original Array:\n',arr)

array_mean=np.mean(arr,axis=1,keepdims=True)
print('Each Row Mean:\n',array_mean)

#subtract mean of each row from row

sub_array=arr-array_mean
print('Subtracted Array:\n',sub_array)
```

```
Original Array:
 [[0.54841487 0.8515766  0.63312329]
 [0.50612695 0.92682301 0.27529464]
 [0.46377625 0.7218959  0.53639224]]
Each Row Mean:
 [[0.67770492]
 [0.56941487]
 [0.57402146]]
Subtracted Array:
 [[-0.12929005  0.17387168 -0.04458163]
 [-0.06328792  0.35740814 -0.29412023]
 [-0.11024521  0.14787443 -0.03762922]]
```

In [61]:

```
#Write a NumPy program to create a 5x5 array with random values and normalize it row-wise.

arr=np.random.rand(5,5)
print('Original Array:\n',arr)

norm_array=arr/np.linalg.norm(arr,axis=1,keepdims=True)
print('Normalized Array:\n',norm_array)
```

```
Original Array:
 [[0.15467937 0.82237085 0.76817429 0.46469158 0.70657009]
 [0.30831668 0.39016893 0.89130238 0.55325044 0.7930352 ]
 [0.32479928 0.23542285 0.15098221 0.82554652 0.59164318]
 [0.21613526 0.44798536 0.68408876 0.6521372  0.39284488]
 [0.47919933 0.72576927 0.8749128  0.07549099 0.51631681]]
Normalized Array:
 [[0.109225   0.58070741 0.54243715 0.32813644 0.49893608]
 [0.21929384 0.27751222 0.63394925 0.39350585 0.56405557]
 [0.29462961 0.2135551  0.1369579  0.74886386 0.53668713]
 [0.18992929 0.39366803 0.60114436 0.57306687 0.34521322]
 [0.35775817 0.5418411  0.6531879  0.05635968 0.38546915]]
```

In [62]:

```
# Write a NumPy program to create a 5x5 array with random values and normalize it column-wise.
arr=np.random.rand(5,5)
print('Original Array:\n',arr)

norm_array=arr/np.linalg.norm(arr,axis=0,keepdims=True)
print('Normalized Array:\n',norm_array)
```

```
Original Array:
 [[0.15286664 0.84377275 0.90783813 0.59055212 0.94627811]
 [0.33313968 0.51099086 0.23517539 0.32323594 0.56381614]
 [0.76049787 0.24150406 0.0802202  0.73343056 0.41429525]
 [0.97535781 0.07437061 0.73844108 0.2505782  0.77899485]
 [0.83117678 0.21856723 0.50280049 0.71075109 0.8370005 ]]
Normalized Array:
 [[0.09961572 0.81016316 0.69957658 0.472955   0.57670149]
 [0.21709085 0.4906368  0.18122526 0.25886971 0.34361316]
 [0.49557929 0.23188435 0.06181738 0.58738195 0.25248887]
 [0.63559302 0.07140824 0.56903986 0.20068036 0.47475208]
 [0.54163729 0.20986115 0.38745613 0.56921866 0.51010315]]
```

In [66]:

```python
# Write a NumPy program to create a 3x3x3 array with random values and find the sum along the last axis.
arr=np.random.rand(3,3,3)
print("Array:\n",arr)

#sum along the last axis
last_axis_sum=np.sum(arr,axis=2)
print('\nsum along the last axis:\n',last_axis_sum)
```

```
Array:
 [[[0.64303664 0.91395506 0.5341238 ]
  [0.25463109 0.46563856 0.4160902 ]
  [0.48666172 0.44313653 0.53252837]]

 [[0.18070431 0.80988229 0.0377392 ]
  [0.51745014 0.39755181 0.51394841]
  [0.47651251 0.13136509 0.84708872]]

 [[0.62572373 0.59895342 0.45397323]
  [0.1183245  0.79058313 0.99799839]
  [0.77493554 0.24075916 0.05903701]]]

sum along the last axis:
 [[2.0911155  1.13635985 1.46232663]
 [1.02832579 1.42895035 1.45496632]
 [1.67865039 1.90690602 1.07473171]]
```

In [71]:

```
# Write a NumPy program to create a 5x5 array with random values and sort each row and each column..

arr=np.random.rand(5,5)
print('Original Array:\n',arr)

#sort each row
sort_row=np.sort(arr,axis=1)
print('\n Sorted Array Row Wise:\n',sort_row)

#sort each column
sort_column=np.sort(arr,axis=0)
print('\n Sorted Array Column Wise:\n',sort_column)
```

```
Original Array:
 [[0.1697527  0.42058094 0.18414608 0.50326277 0.02342832]
 [0.71011527 0.34625079 0.64936326 0.99648686 0.6244156 ]
 [0.68503994 0.3040134  0.04623042 0.47195204 0.10670091]
 [0.66403171 0.30772919 0.31635111 0.0391489  0.91463406]
 [0.15779197 0.70356895 0.20840064 0.7335755  0.63611169]]

 Sorted Array Row Wise:
 [[0.02342832 0.1697527  0.18414608 0.42058094 0.50326277]
 [0.34625079 0.6244156  0.64936326 0.71011527 0.99648686]
 [0.04623042 0.10670091 0.3040134  0.47195204 0.68503994]
 [0.0391489  0.30772919 0.31635111 0.66403171 0.91463406]
 [0.15779197 0.20840064 0.63611169 0.70356895 0.7335755 ]]

 Sorted Array Column Wise:
 [[0.15779197 0.3040134  0.04623042 0.0391489  0.02342832]
 [0.1697527  0.30772919 0.18414608 0.47195204 0.10670091]
 [0.66403171 0.34625079 0.20840064 0.50326277 0.6244156 ]
 [0.68503994 0.42058094 0.31635111 0.7335755  0.63611169]
 [0.71011527 0.70356895 0.64936326 0.99648686 0.91463406]]
```

In [81]:

```
# Write a NumPy program to create a 5x5 array with random values and find the second-largest value in each row and ea

arr=np.random.rand(5,5)
print('Original Array:\n',arr)

#find the second-largest value in each row.
second_large=np.sort(arr,axis=1)[:,-2]
print('\nsecond-largest value in each row:\n',second_large)

#find the second-largest value in each column.
second_large_colmn=np.sort(arr,axis=0)[-2,:]
print('\nsecond-largest value in each row:\n',second_large_colmn)
```

```
Original Array:
 [[0.85689383 0.38553209 0.83125308 0.85592583 0.45073735]
 [0.4673078  0.46795145 0.72016841 0.47432157 0.51249225]
 [0.10035391 0.54103113 0.26514935 0.72536941 0.5107826 ]
 [0.43153625 0.68555536 0.78720863 0.4925543  0.78894111]
 [0.02996217 0.97135238 0.82079645 0.93905645 0.60784265]]

second-largest value in each row:
 [0.85592583 0.51249225 0.54103113 0.78720863 0.93905645]

second-largest value in each row:
 [0.4673078  0.68555536 0.82079645 0.85592583 0.60784265]
```

In [91]:

```
#Write a NumPy program to create a 5x5 array with random values and replace the minimum value with 0

arr=np.random.rand(5,5)
print('Original Array:\n',arr)

min_val=arr.min()

#print(min_val)

arr[arr==min_val]=0

print('\nReplaced array with zero:\n',arr)
```

```
Original Array:
 [[0.65286387 0.42255948 0.38474032 0.44970383 0.88727202]
 [0.84212707 0.22735667 0.30557384 0.61900359 0.79595243]
 [0.97560352 0.73040505 0.45494814 0.00581729 0.75162299]
 [0.20994904 0.55126347 0.33892228 0.46216491 0.28923874]
 [0.09783679 0.18141488 0.68750253 0.75100441 0.62907971]]

Replaced array with zero:
 [[0.65286387 0.42255948 0.38474032 0.44970383 0.88727202]
 [0.84212707 0.22735667 0.30557384 0.61900359 0.79595243]
 [0.97560352 0.73040505 0.45494814 0.         0.75162299]
 [0.20994904 0.55126347 0.33892228 0.46216491 0.28923874]
 [0.09783679 0.18141488 0.68750253 0.75100441 0.62907971]]
```

In [98]:

```
#Write a NumPy program to create a 5x5 array with random values and calculate the exponential of each element.

arr=np.random.rand(5,5)
print('Original Array:\n',arr)

exp_array=np.exp(arr)
print('\nexponential of each element:\n',exp_array)
```

```
Original Array:
 [[0.82882171 0.25584133 0.14609154 0.14542163 0.64217745]
 [0.54696562 0.09872609 0.05494649 0.28975616 0.25270098]
 [0.07317743 0.68278335 0.93202751 0.27135038 0.87595979]
 [0.61450714 0.09594082 0.57084674 0.01629019 0.00167642]
 [0.36473732 0.8475564  0.97087363 0.31941078 0.01547156]]

exponential of each element:
 [[2.29061814 1.29154778 1.15730213 1.15652709 1.90061486]
 [1.72800165 1.10376393 1.05648408 1.33610165 1.28749823]
 [1.07592143 1.97937937 2.53965313 1.3117346  2.40117883]
 [1.8487452  1.10069392 1.76976494 1.0164236  1.00167783]
 [1.44013566 2.33393667 2.64025005 1.37631657 1.01559187]]
```