In [ ]:

```
#importing numpy if not present pip3 install numpy
import numpy as np
```

In [ ]:

```
#1-DIMENSIONAL ARRAY
a=np.array([1,2,3,4])
print(a)
```

```
[1 2 3 4]
```

In [ ]:

```
#2-DIMENSIONAL ARRAY
a=np.array([[1,2,3,4],[4,5,6,7]])
print(a)
```

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

In [ ]:

```
#GET DIMENSION
print(a.ndim)
```

```
2
```

In [ ]:

```
#GET shape
print(a.shape)
```

```
(2, 4)
```

In [ ]:

```
#GET type
print(a.dtype)
```

```
int64
```

In [ ]:

```
a=np.array([[1,2,3,4],[4,5,6,7]],dtype="int8")
print(a)
print(a.dtype)
```

```
[[1 2 3 4]
 [4 5 6 7]]
int8
```

In [ ]:

```
#print item size and total size
a=np.array([[1,2,3,4],[4,5,6,7]],dtype="int64")
print(a.itemsize)
print(a.nbytes)
```

```
8
64
```

In [ ]:

```
#get a specific element a[r][c]
a=np.array([[1,2,3,4],[0,5,6,7]])
print(a[1][2])
```

```
6
```

In [ ]:

```
#to print an entire row
a=np.array([[1,2,3,4],[0,5,6,7]])
print(a[1,:])
```

```
[0 5 6 7]
```

In [ ]:

```
a=np.array([[1,2,3,4],[0,5,6,7]])
print(a[1,3])
```

```
7
```

In [ ]:

```
#print selected number a[startindex:stopindex,stepsize]
a=np.array([[1,2,3,4,9,10,11],[0,5,6,7,12,14,15]])
print(a[0,2:5:2])
```

```
[3 9]
```

In [ ]:

```
#changing values in an array
a=np.array([[1,2,3,4,9,10,11],[0,5,6,7,12,14,15]])
a[1,5]=30
print(a)
```

```
[[ 1  2  3  4  9 10 11]
 [ 0  5  6  7 12 30 15]]
```

In [ ]:

```
a[1,:]=2
print(a)
```

```
[[ 1  2  3  4  9 10 11]
 [ 2  2  2  2  2  2  2]]
```

In [ ]:

```
#3 dimensional array
a=np.array([[[1,2],[3,4]],[[5,6],[7,8]],[[9,10],[11,12]]])
print (a)
print(a.ndim)
print(a[1,0,1])
```

```
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
3
6
```

In [ ]:

```
#different ways to create an array
```

In [ ]:

```
#all zeros
import numpy as np
a=np.zeros ((2, 3),dtype="int8")
print(a)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

In [ ]:

```
import numpy as np
a=np.zeros ((2, 3),dtype="int8")
print(a)
```

```
[[0 0 0]
 [0 0 0]]
```

In [ ]:

```
#all ones
import numpy as np
a=np.ones((2, 3))
print(a)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

In [ ]:

```
#any other number
import numpy as np
a=np.full((2, 3),5)
print(a)
```

```
[[5 5 5]
 [5 5 5]]
```

In [ ]:

```
#shape of first array value is inserted
import numpy as np
a=np.array([[[1,2],[3,4]],[[5,6],[7,8]],[[9,10],[11,12]]])
a=np.full_like(a,5)
print(a)
```

```
[[[5 5]
  [5 5]]

 [[5 5]
  [5 5]]

 [[5 5]
  [5 5]]]
```

In [ ]:

```
#random numbers
import numpy as np
np.random.randint(low=3,high=10,size=5)
```

Out[ ]:

```
array([7, 5, 7, 5, 6])
```

In [ ]:

```
#np.identity matrix
import numpy as np
ar=np.identity(3)
print(ar)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [ ]:

```
b = np.eye(2, dtype = float)
print("Matrix b : \n", b)

# matrix with R=4 C=5 and 1 on diagonal
# below main diagonal
a = np.eye(4, 5, k = 2)

print("\nMatrix a : \n", a)
```

```
Matrix b :
 [[1. 0.]
 [0. 1.]]

Matrix a :
 [[0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0.]]
```

In [ ]:

```
#arrange
import numpy as np
a=np.arange(0,12,1)
print(a)
print(np.reshape(a,(3,4)))
a=np.arange(0,12,2)
print(a)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[ 0  2  4  6  8 10]
```

In [ ]:

```
#linespace
import numpy as np
a=np.linspace(2.0, 3.0, num=8)
print(a)
```

```
[2.         2.14285714 2.28571429 2.42857143 2.57142857 2.71428571
 2.85714286 3.        ]
```

In [ ]:

```
#llogspace
import numpy as np
a=np.logspace(2, 3, num=8)
print(a)
```

```
[ 100.          138.94954944  193.06977289  268.26957953  372.75937203
  517.94746792  719.685673   1000.        ]
```

In [ ]:

```
#identity
import numpy as np
a=np.identity(3)
print(a)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [ ]:

```python
# Python program to demonstrate
# basic operations on single array
import numpy as np

a = np.array([1, 2, 5, 3])

# add 1 to every element
print ("Adding 1 to every element:", a+1)

# subtract 3 from each element
print ("Subtracting 3 from each element:", a-3)

# multiply each element by 10
print ("Multiplying each element by 10:", a*10)

# square each element
print ("Squaring each element:", a**2)

# modify existing array
a *= 2
print ("Doubled each element of original array:", a)

# transpose of array
a = np.array([[1, 2, 3], [3, 4, 5], [9, 6, 0]])

print ("\nOriginal array:\n", a)
print ("Transpose of array:\n", a.T)
```

```
Adding 1 to every element: [2 3 6 4]
Subtracting 3 from each element: [-2 -1  2  0]
Multiplying each element by 10: [10 20 50 30]
Squaring each element: [ 1  4 25  9]
Doubled each element of original array: [ 2  4 10  6]

Original array:
 [[1 2 3]
 [3 4 5]
 [9 6 0]]
Transpose of array:
 [[1 3 9]
 [2 4 6]
 [3 5 0]]
```

In [ ]:

```python
# Python program to demonstrate
# binary operators in Numpy
import numpy as np

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

# add arrays
print ("Array sum:\n", a + b)

# multiply arrays (elementwise multiplication)
print ("Array multiplication:\n", a*b)
 # matrix dot product
# matrix multiplication
print ("Matrix multiplication:\n", a.dot(b))
# matrix dot product
# multiply matrices with @ operator
D = a @ b
print(D)
```

```
Array sum:
 [[5 5]
 [5 5]]
Array multiplication:
 [[4 6]
 [6 4]]
Matrix multiplication:
 [[ 8  5]
 [20 13]]
[[ 8  5]
 [20 13]]
```

In [ ]:

```python
#array division
import numpy as np
A = np.array([[1, 2, 3],[4, 5, 6]])
print(A)
# define second matrix
B = np.array([[1, 2, 3],[4, 5, 6]])
print(B)
# divide matrices
C = A / B
print(C)
```

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

In [ ]:

```
# create an array of sine values
a = np.array([0,11,1])
print ("Sine values of array elements:", np.sin(a))

# exponential values
a = np.array([0, 1, 2, 3])
print ("Exponent of array elements:", np.exp(a))

# square root of array values
print ("Square root of array elements:", np.sqrt(a))
```

```
Sine values of array elements: [ 0.         -0.99999021  0.84147098]
Exponent of array elements: [ 1.          2.71828183  7.3890561  20.085
53692]
Square root of array elements: [0.         1.         1.41421356 1.7320
5081]
```

In [ ]:

```
# Python program to demonstrate sorting in numpy
import numpy as np
a = np.array([[1, 4, 2],[3, 4, 6], [0, -1, 5]])
 # sorted array
print ("Array elements in sorted order:\n",
                    np.sort(a,axis=None))

# sort array row-wise
print ("Row-wise sorted array:\n",
                np.sort(a, axis = 1))

# specify sort algorithm
print ("Column wise sort by applying merge-sort:\n",
            np.sort(a, axis = 0, kind = 'mergesort'))
```

```
Array elements in sorted order:
 [-1  0  1  2  3  4  4  5  6]
Row-wise sorted array:
 [[ 1  2  4]
 [ 3  4  6]
 [-1  0  5]]
Column wise sort by applying merge-sort:
 [[ 0 -1  2]
 [ 1  4  5]
 [ 3  4  6]]
```

In [ ]:

```
#append list
import numpy as np
A=np.array([10,20,30])
print(A)
A=np.append(A,[40,50])
print(A)
```

```
[10 20 30]
[10 20 30 40 50]
```

In [ ]:

```
#Add two matrix and find the transpose of the result ( university question)
def readmatrix(x,r,c):
    for i in range(r):
        for j in range(c):
            x[i][j]=int(input('enter elements row by row'))
import numpy as np
r1=int(input('rows of a'))
c1=int(input('columns of a'))
r2=int(input('rows of b'))
c2=int(input('columns of b'))
if r1!=r2 or c1!=c2:
    print("cant add matrices")
else:
    A=np.zeros((r1,c1))
    print("Enter the elements of A")
    readmatrix(A,r1,c1)
    B=np.zeros((r2,c2))
    print("Enter the elements of B")
    readmatrix(B,r2,c2)
    print("Matrix A")
    print(A)
    print("Matrix B")
    print(B)
    C=A+B
    print("sum")
    print(C)
    print("transpose of sum")
    print(C.T)
```

In [ ]:

```
import numpy
# define orthogonal matrix
Q = np.array([[1, 0],[0, -1]])
print(Q)
# inverse equivalence
V = np.linalg.inv(Q)
print(V)
tran=Q.T
if((V==tran).all()):
 print("orthogonal")
```

```
[[ 1  0]
 [ 0 -1]]
[[ 1.  0.]
 [-0. -1.]]
orthogonal
```

In [ ]:

```
#adding a new row to a array
A=np.array([[1,2],[3,4]])
A=np.append(A,[[5,6]],axis=0)
print(A)
```

```
[[1 2]
 [3 4]
 [5 6]]
```

In [ ]:

```python
#adding a new coulumn to a array
A=np.append(A,[[5],[6]],axis=1)
print(A)
```

```
[[1 2 5]
 [3 4 6]]
```

In [ ]:

```python
#delete an elemnet from array
A=np.array([10,20,30,40,50,60,70,80])
print(A)
A=np.delete(A,1)
print(A)
```

```
[10 20 30 40 50 60 70 80]
[10 30 40 50 60 70 80]
```

In [ ]:

```python
A=np.array([[10,20,30],[40,50,60],[70,80,90]])
print(A)
```

```
[[10 20 30]
 [40 50 60]
 [70 80 90]]
```