```
import numpy as np
```

Rename notebook

```
#reshape()r,c
a=np.arange(1,13).reshape(4,3)
print(a)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

```
a=np.linspace(2,4,10).reshape(5,2)
print(a)
```

```
[[2.         2.22222222]
 [2.44444444 2.66666667]
 [2.88888889 3.11111111]
 [3.33333333 3.55555556]
 [3.77777778 4.        ]]
```

```
a=np.array([1,2,3,4,5,6,7,8]).reshape(2,2,2)#Normal array can be reshaped.
print(a)
```

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

 [[5 6]
  [7 8]]]
```

```
a=np.array([1,2,3,4,5,6,7,8])
a.resize(2,2,2)
print(a)
```

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

 [[5 6]
  [7 8]]]
```

```
#Array operations
a=np.array([3,5,7,9])
b=np.array([1,2,4,6])#list will throw an error while array iterative themselves if you perfrom same operation in list
print(a-b)
```

```
[2 3 3 3]
```

```
a=np.array([3,5,7,9])
b=np.array([1,2,4,6])
print(a+b)
```

```
[ 4  7 11 15]
```

```
#Matrix addition
a=np.array([3,5,7,9]).reshape(2,2)
b=np.array([1,2,4,6]).reshape(2,2)
print(a)
print("__________________")
print(b)
print("__________________")
print(a+b)
```

```
[[3 5]
 [7 9]]
__________________
[[1 2]
 [4 6]]
__________________
[[ 4  7]
 [11 15]]
```

```
#Matrix multiplication
a=np.array([3,5,7,9]).reshape(2,2)
b=np.array([1,2,4 Rename notebook )
print(a)
print("__________________")
print(b)
print("__________________")
print(a@b)
```

```
[[3 5]
 [7 9]]
__________________
[[1 2]
 [4 6]]
__________________
[[23 36]
 [43 68]]
```

```
#Matrix multiplication
a=np.array([3,5,7,9]).reshape(2,2)
b=np.array([1,2,4,6]).reshape(2,2)
print(a)
print("__________________")
print(b)
print("__________________")
print(a.dot(b))
```

```
[[3 5]
 [7 9]]
__________________
[[1 2]
 [4 6]]
__________________
[[23 36]
 [43 68]]
```

```
#max and min func -applicable to only Numeric datatypes(int,float)
a=np.array([11,22,33,44,55])
b=np.array([2.43,5.33,6.23,3.23])
print(a.max())
print(a.min())
print(b.max())
print(b.min())
```

```
55
11
6.23
2.43
```

```
a=np.array([3,7,5,9]).reshape(2,2)
print(a)
print("_________________")
print(a.max(axis=0))#axis=0 means column
print("_________________")
print(a.max(axis=1))#axis=1 means row
```

```
[[3 7]
 [5 9]]
_________________
[5 9]
_________________
[7 9]
```

```
a=np.array([3,7,5,9]).reshape(2,2)
print(a)
print("_________________")
print(a.min(axis=0))#axis=0 means column
print("_________________")
print(a.min(axis=1))#axis=1 means row
```

```
[[3 7]
 [5 9]]
_________________
[3 7]
```

```
    _______________
    [3 5]
```

Rename notebook

```
#Joining arrays
a=np.array([3,5,7,9]).reshape(2,2)
b=np.array([1,2,4,6]).reshape(2,2)
print(a)
print("_______________")
print(b)
print("After vertically joining the arrays")
print(np.vstack((a,b)))
```

```
    [[3 5]
     [7 9]]
    _______________
    [[1 2]
     [4 6]]
    After vertically joining the arrays
    [[3 5]
     [7 9]
     [1 2]
     [4 6]]
```

```
#Joining arrays
a=np.array([3,5,7,9]).reshape(2,2)
b=np.array([1,2,4,6]).reshape(2,2)
print(a)
print("_______________")
print(b)
print("After horizontally joining the arrays")
print(np.hstack((a,b)))
```

```
    [[3 5]
     [7 9]]
    _______________
    [[1 2]
     [4 6]]
    After horizontally joining the arrays
    [[3 5 1 2]
     [7 9 4 6]]
```

```
#Joining arrays
a=np.array([3,5,7,9]).reshape(2,2)
b=np.array([1,2,4,6]).reshape(2,2)
print(a)
print("_______________")
print(b)
print("After vertically joining the arrays")
print(np.stack((a,b),axis=1))
```

```
    [[3 5]
     [7 9]]
    _______________
    [[1 2]
     [4 6]]
    After vertically joining the arrays
    [[[3 5]
      [1 2]]

     [[7 9]
      [4 6]]]
```

```
a=np.arange(24).reshape(2,3,4)
print(a)
print("After dstack")
print(np.dstack(a))
#no.of rows--> no.of groups
#no.of columns in every group--> no.of rows in every group [0,0],[0,1],[0,2]......
```

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

     [[12 13 14 15]
      [16 17 18 19]
```

```
  [20 21 22 23]]]
After dstack
[[[ 0 12]
  [ 1 13]
  [ 2 14]
  [ 3 15]]

 [[ 4 16]
  [ 5 17]
  [ 6 18]
  [ 7 19]]

 [[ 8 20]
  [ 9 21]
  [10 22]
  [11 23]]]
```

```
#Splitting arrays
a=np.arange(18).reshape(6,3)
print(a)
print("____________")
#Splits vertically(rows)
np.vsplit(a,3)#(arrayname,no.of pieces)--> if no.of pieces/no.of rows is possible
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [15 16 17]]
____________
[array([[0, 1, 2],
        [3, 4, 5]]),
 array([[ 6,  7,  8],
        [ 9, 10, 11]]),
 array([[12, 13, 14],
        [15, 16, 17]])]
```

```
#Splitting arrays
a=np.arange(18).reshape(6,3)
print(a)
print("____________")
np.vsplit(a,(2,3))
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [15 16 17]]
____________
[array([[0, 1, 2],
        [3, 4, 5]]),
 array([[6, 7, 8]]),
 array([[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]])]
```

```
#Splitting arrays
a=np.arange(18).reshape(3,6)
print(a)
print("____________")
np.hsplit(a,3)#Splits horizontally(columns)
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]]
____________
[array([[ 0,  1],
        [ 6,  7],
        [12, 13]]),
 array([[ 2,  3],
        [ 8,  9],
        [14, 15]]),
 array([[ 4,  5],
        [10, 11],
        [16, 17]])]
```

```
#Splitting arrays
a=np.arange(18).r
print(a)
print("____________")
np.hsplit(a,(2,5))#portablly Splits (columns)
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]]
____________
[array([[ 0,  1],
        [ 6,  7],
        [12, 13]]),
 array([[ 2,  3,  4],
        [ 8,  9, 10],
        [14, 15, 16]]),
 array([[ 5],
        [11],
        [17]])]
```

```
#argmax function display the position of maximum element
a=np.array([11,63,43,56]).reshape(2,2)
print(a)
print(a.argmax())
print(a.argmin())
```

```
[[11 63]
 [43 56]]
1
0
```

```
#argmax function display the position of maximum element
a=np.array([11,63,43,56]).reshape(2,2)
print(a)
print("______________")
print(a.argmax(axis=0))#Column
print(a.argmin(axis=0))#position[0,0]
                                #[1,2]
```

```
[[11 63]
 [43 56]]
______________
[1 0]
[0 1]
```

```
#argmax function display the position of maximum element
a=np.array([11,63,43,56]).reshape(2,2)
print(a)
print("______________")   #position[0,1]
                                  #[0,1]
print(a.argmax(axis=1))#Row
```

```
[[11 63]
 [43 56]]
______________
[1 1]
```

STATISTICS

```
#STATISTICS
#mean,median,standard deviation,variance
a=np.array([1,2,3,4,5])
print("Mean: ",np.mean(a))
print("Median: ",np.median(a))
print("Variance: ",np.var(a))
print("Standard Deviation: ",np.std(a))
```

```
Mean:  3.0
Median:  3.0
Variance:  2.0
Standard Deviation:  1.4142135623730951
```

```
a=np.pi
print(a)#radians
print(np.rad2deg(
```

Rename notebook

```
    3.141592653589793
    180.0
```

```
a=np.array([np.pi/4,np.pi/3,np.pi/2,np.pi])
print(a)
```

```
    [0.78539816 1.04719755 1.57079633 3.14159265]
```

```
#Radian to degree conversion
a=np.array([np.pi/4,np.pi/3,np.pi/2,np.pi])
b=(np.rad2deg(a))
print(b)
```

```
    [ 45.  60.  90. 180.]
```

```
#Degree to radian conversion
c=np.array([30,45,60,90,120,180])
print(np.deg2rad(c))
```

```
    [0.52359878 0.78539816 1.04719755 1.57079633 2.0943951  3.14159265]
```

```
#Trignometry value(sin,cos,tan)
c=np.array([0,30,45,60,90,120,180])
print(np.sin(c))
print(np.cos(c))
print(np.tan(c))
```

```
    [ 0.         -0.98803162  0.85090352 -0.30481062  0.89399666  0.58061118
     -0.80115264]
    [ 1.          0.15425145  0.52532199 -0.95241298 -0.44807362  0.81418097
     -0.59846007]
    [ 0.         -6.4053312   1.61977519  0.32004039 -1.99520041  0.71312301
      1.33869021]
```

```
np.arcsin(1)
```

```
    1.5707963267948966
```

```
#Pythogorous theorem
a=8
b=6
print(np.hypot(a,b))
```

```
    10.0
```

Searching

```
a=np.array([3,5,8,9,3])
print(np.where(a==8))#Equal
```

```
    (array([2]),)
```

```
a=np.array([3,5,8,9,3,6])
print(np.where(a%2==0))#even
```

```
    (array([2, 5]),)
```

```
a=np.array([3,5,8,9,3])
print(np.where(a>5))#Greater than 5
```

```
    (array([2, 3]),)
```

```
a=np.array([3,5,8,9,3])
print(np.searchsorted(a,8))
```

```
     2
```

Linear algebra anc Rename notebook

```
a=np.array([12,34,45,67])
b=np.array([11,13,24,35])
print(np.add(a,b))#addition

     [ 23  47  69 102]
```

```
a=np.array([12,34,45,67])
b=np.array([11,13,24,35])
print(np.subtract(a,b))#subtraction

     [ 1 21 21 32]
```

```
a=np.array([12,34,45,67])
b=np.array([11,13,24,35])
print(np.divide(a,b))#division -Quatient

     [1.09090909 2.61538462 1.875      1.91428571]
```

```
a=np.array([12,34,45,67])
b=np.array([11,13,24,35])
print(np.mod(a,b))#modulus -Remainder

     [ 1  8 21 32]
```

```
a=np.array([12,34,45,67])
b=np.array([11,13,24,35])
print(np.divmod(a,b))#first division and then modulus-quatient and reaminder

     (array([1, 2, 1, 1]), array([ 1,  8, 21, 32]))
```

```
a=np.array([5,7,4,9,2])
print(np.sort(a))#sorting

     [2 4 5 7 9]
```

```
a=np.array([5,7,4,9,2])
print(np.diff(a))#difference -difference between second and fisrt element

     [ 2 -3  5 -7]
```

```
#Union
a=np.array([12,34,45,67])
b=np.array([11,13,24,35])
print(np.union1d(a,b))

     [11 12 13 24 34 35 45 67]
```

```
#intersection
a=np.array([12,34,45,67])
b=np.array([11,12,24,35])
print(np.intersect1d(a,b))

     [12]
```

```
#A-B
a=np.array([12,34,45,67])
b=np.array([11,12,24,35])
print(np.setdiff1d(a,b))


     [34 45 67]
```

```
a=np.array([12,34,45,67])
b=np.array([11,13,24,35])
print(np.setdiff1
```

Rename notebook

```
[11 13 24 35]
```

Rounding

```
#truncate -remove all the decimal valies
a=np.trunc([-2.345673,4.34554])
print(a)
```

```
[-2.  4.]
```

```
#fix similar to truncate
a=np.fix([-2.345673,4.34554])
print(a)
```

```
[-2.  4.]
```

```
#around -(value,precision)upto precision only be display
a=np.around(2.345673,4)
print(a)
```

```
2.3457
```

```
a=np.array([-2.345673,4.34554])
print(np.floor(a))
```

```
[-3.  4.]
```

```
a=np.array([-2.345673,4.34554])
print(np.ceil(a))
```

```
[-2.  5.]
```

More fuctions!

```
#Cummulative sum
a=np.array([2,5,11,7,9])
print(np.cumsum(a))
```

```
[ 2  7 18 25 34]
```

```
#Cummulative product
a=np.array([2,5,11,7,9])
print(np.cumprod(a))
```

```
[   2   10  110  770 6930]
```

```
#LCM and GCD for two elements
a=5
b=10
print(np.lcm(a,b))
print(np.gcd(a,b))
```

```
10
5
```

```
#LCM more than two elements using reduce function
a=np.array([2,3,4,5])
print(np.lcm.reduce(a))
```

```
60
```

```
#Inverse of a matrix
a=np.array([[1,2],[3,4]])
print(np.linalg.i
```

Rename notebook

```
[[-2.   1. ]
 [ 1.5 -0.5]]
```

Random

```
from numpy import random as rd
```

```
a=rd.rand()#default limit lies b/w 0-0.9999
print(a)
```

```
0.30506490333521463
```

```
a=rd.rand(5)#rand(n)-n numbers
print(a)
```

```
[0.16148123 0.19471522 0.66320513 0.2545873  0.72708335]
```

```
a=rd.randint(5)#randint(n)-limit is n and only integers are displayed
print(a)
```

```
2
```

```
a=rd.randint(10,size=(2))#randint(limit,size)
print(a)
```

```
[9 8]
```

```
a=rd.randint(10,size=(2,3))#randint(limit,size=(shape))
print(a)
```

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

```
#Inner joint
a=np.array([1,2,3])
b=np.array([4,5,6])
#1x4 + 2x5 + 3x6
print(np.inner(a,b))
```

```
32
```

```
#Outer joint
a=np.array([1,2,3])
b=np.array([4,5,6])
#one element in a multiplies with all other elements in b
#1x(4,5,6) 2x(4,5,6) 3x(4,5,6)
print(np.outer(a,b))
```

```
[[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
```

```
#Cross joint
a=np.array([1,2,3])
b=np.array([3,7,8])
#2x8-3x7 1x8-3x3
print(np.cross(a,b))
```

```
[-5  1  1]
```

Rename notebook