np.sort

Return a sorted copy of an array. But normal sorted function ek list return karta hai lekin np.sort ek numpy array return karta hai

https://numpy.org/doc/stable/reference/generated/numpy.sort.html

```
# code
import numpy as np
a = np.random.randint(1,100,15)
a
```

```
array([38, 11, 89, 98, 46, 84, 33, 26, 54,  2, 12, 78, 34, 47, 70])
```

```
b =np.random.randint(1,100,24).reshape(6,4)
b
```

```
array([[10, 25, 95, 51],
       [14, 24, 41, 66],
       [47, 10, 71, 10],
       [19, 90, 43, 40],
       [77, 56,  1, 22],
       [16, 10, 14, 54]])
```

```
np.sort(a)
```

```
array([12, 17, 28, 33, 33, 35, 37, 48, 53, 54, 63, 64, 82, 85, 88])
```

```
np.sort(a)[::-1]  # abb reverse order me sort kar dega
```

```
array([88, 85, 82, 64, 63, 54, 53, 48, 37, 35, 33, 33, 28, 17, 12])
```

```
np.sort(b)  #row wise sorting ho jata hai and ye by default hota hai
agar aap explicitely karna chahate ho to axis=1 kar do and if aap
column wise karna chahate ho to axis=0 de do
```

```
array([[10, 47, 59, 78],
       [52, 60, 78, 78],
       [14, 34, 51, 71],
       [17, 75, 80, 89],
       [36, 61, 88, 96],
       [35, 40, 46, 93]])
```

```
np.sort(b,axis=1)  # row wise sorting
```

```
array([[10, 47, 59, 78],
       [52, 60, 78, 78],
       [14, 34, 51, 71],
       [17, 75, 80, 89],
       [36, 61, 88, 96],
       [35, 40, 46, 93]])
```

```
np.sort(b,axis=0)  #column  wise sorting
```

```
array([[17, 46, 10, 34],
       [35, 51, 14, 36],
       [47, 52, 40, 59],
       [71, 78, 61, 60],
       [78, 80, 78, 75],
       [88, 96, 89, 93]])
```

np.append

The numpy.append() appends values along the mentioned axis at the end of the array

https://numpy.org/doc/stable/reference/generated/numpy.append.html

```
a
```

```
array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])
```

```
np.append(a,100)  # a ke end me 100 append kar dega
```

```
array([ 37,  88,  64,  35,  12,  54,  33,  28,  85,  33,  53,  17,
48,
        63,  82, 100])
```

```
b
```

```
array([[10, 25, 95, 51],
       [14, 24, 41, 66],
       [47, 10, 71, 10],
       [19, 90, 43, 40],
       [77, 56,  1, 22],
       [16, 10, 14, 54]])
```

```
np.append(b,np.random.random((b.shape[0],1)),axis=1) # last me ek aur
column add ho jayegan
# ((b.shape[0],1)),axis=1)   b ka jo shape hai uska zeroth item
```

```
array([[47.        , 78.        , 10.        , 59.        ,
0.80997312],
       [78.        , 52.        , 78.        , 60.        ,  0.8083781
],
       [71.        , 51.        , 14.        , 34.        ,
0.71278017],
       [17.        , 80.        , 89.        , 75.        ,
0.61652769],
       [88.        , 96.        , 61.        , 36.        ,
0.79994109],
       [35.        , 46.        , 40.        , 93.        ,
0.81752219]])
```

np.concatenate

numpy.concatenate() function concatenate a sequence of arrays along an existing axis. generally 2D (tabular data) array ke sath use karte hain

https://numpy.org/doc/stable/reference/generated/numpy.concatenate.html

```
c = np.arange(6).reshape(2,3)
d = np.arange(6,12).reshape(2,3)

print(c)
print()
print(d)
```

```
[[0 1 2]
 [3 4 5]]

[[ 6  7  8]
 [ 9 10 11]]
```

```
np.concatenate((c,d),axis=0)   # vertically concatinate ho gya similar hstack se bhi kar sakte the
```

```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

```
np.concatenate((c,d),axis=1)  # horizontally concatinate
```

```
array([[ 0,  1,  2,  6,  7,  8],
       [ 3,  4,  5,  9, 10, 11]])
```

np.unique

With the help of np.unique() method, we can get the unique values from an array given as parameter in np.unique() method.

https://numpy.org/doc/stable/reference/generated/numpy.unique.html/

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

np.unique(e)
```

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

np.expand_dims (expand dimensions)

With the help of Numpy.expand_dims() method, we can get the expanded dimensions of an array https://numpy.org/doc/stable/reference/generated/numpy.expand_dims.html

```
# koi array 1D me hai to usse 2D me kar sakte ha , 2D me hai to 3D me
kar sakte ho and so on

a

array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])

a.shape

(15,)

np.expand_dims(a,axis=0)  # ye 2D array hai with 1 row

array([[37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82]])

np.expand_dims(a,axis=0).shape

(1, 15)

np.expand_dims(a,axis=1)  # 2D array with 1 column

array([[37],
       [88],
       [64],
       [35],
       [12],
       [54],
       [33],
       [28],
       [85],
       [33],
       [53],
       [17],
       [48],
       [63],
       [82]])

np.expand_dims(a,axis=1).shape

(15, 1)
```

np.where

The numpy.where() function returns the indices of elements in an input array where the given condition is satisfied.

https://numpy.org/doc/stable/reference/generated/numpy.where.html

```
a
```

```
array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])

# find all indices with value grater than 50
np.where(a>50)

(array([ 1,  2,  5,  8, 10, 13, 14], dtype=int64),)

# replace all values > 50 with 0
# np.where(condition,true,false)
np.where(a>50,0,a)  # jaha jaha a>50 hai usko replace karo zero se and
baki sab jagah jo hai wahi rahne do

array([37,  0,  0, 35, 12,  0, 33, 28,  0, 33,  0, 17, 48,  0,  0])

#replace  all the even number with zero
np.where(a%2==0,0,a)

array([37,  0,  0, 35,  0,  0, 33,  0, 85, 33, 53, 17,  0, 63,  0])
```

np.argmax

The numpy.argmax() function returns indices of the max element of the array in a particular axis.

https://numpy.org/doc/stable/reference/generated/numpy.argmax.html

```
a

array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])

np.argmax(a)  #kon se index pe marimum number hai

1

b

array([[47, 78, 10, 59],
       [78, 52, 78, 60],
       [71, 51, 14, 34],
       [17, 80, 89, 75],
       [88, 96, 61, 36],
       [35, 46, 40, 93]])

np.argmax(b)

17

np.argmax(b,axis=0)  # har column me se maximum elemet index bata dega

array([4, 4, 3, 5], dtype=int64)

np.argmax(b,axis=1)   # har row  me se maximum elemet index bata dega

array([1, 0, 0, 2, 1, 3], dtype=int64)
```

np.argmin

The numpy.argmin() function returns indices of the min element of the array in a particular axis.

https://numpy.org/doc/stable/reference/generated/numpy.argmix.html

```
a
```

```
array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])
```

```
np.argmin(a)
```

```
4
```

```
b
```

```
array([[47, 78, 10, 59],
       [78, 52, 78, 60],
       [71, 51, 14, 34],
       [17, 80, 89, 75],
       [88, 96, 61, 36],
       [35, 46, 40, 93]])
```

```
np.argmin(b)
```

```
2
```

```
np.argmin(b,axis=0)
```

```
array([3, 5, 0, 2], dtype=int64)
```

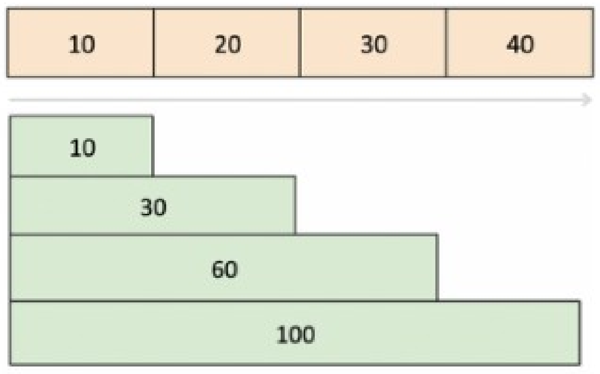
```
np.argmin(b,axis=1)
```

```
array([2, 1, 2, 0, 3, 0], dtype=int64)
```

np.cumsum

numpy.cumsum() function is used when we want to compute the cumulative sum of array elements over a given axis.

https://numpy.org/doc/stable/reference/generated/numpy.cumsum.html



Vector Cumulative Sum

```
a
```

```
array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])
```

```
np.cumsum(a)
```

```
array([ 37, 125, 189, 224, 236, 290, 323, 351, 436, 469, 522, 539, 
587,
       650, 732])
```

```
b
```

```
array([[47, 78, 10, 59],
       [78, 52, 78, 60],
       [71, 51, 14, 34],
       [17, 80, 89, 75],
       [88, 96, 61, 36],
       [35, 46, 40, 93]])
```

```
np.cumsum(b)  # yadi koi axis provide nahi klarenge to ye !D me 
convert kar dega
```

```
array([  47,  125,  135,  194,  272,  324,  402,  462,  533,  584, 
598,
         632,  649,  729,  818,  893,  981, 1077, 1138, 1174, 1209, 
1255,
       1295, 1388])
```

```
np.cumsum(b,axis=0)
```

```
array([[ 47,  78,  10,  59],
       [125, 130,  88, 119],
       [196, 181, 102, 153],
       [213, 261, 191, 228],
       [301, 357, 252, 264],
       [336, 403, 292, 357]])
```

```
np.cumsum(b,axis=1)
```

```
array([[ 47, 125, 135, 194],
       [ 78, 130, 208, 268],
       [ 71, 122, 136, 170],
       [ 17,  97, 186, 261],
       [ 88, 184, 245, 281],
       [ 35,  81, 121, 214]])
```

```
# cumprod (cumulative product ) :similar as cumsum but ye product kar
deta hai
```

```
a
```

```
array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])
```

```
np.cumprod(a)
```

```
array([         37,        3256,      208384,     7293440,
87521280,
         431181824,  1344098304, -1019953152,  -796672000,  -
520372224,
       -1809924096,  -703938560,   570687488,  1593573376,
1823997952])
```

```
b
```

```
array([[47, 78, 10, 59],
       [78, 52, 78, 60],
       [71, 51, 14, 34],
       [17, 80, 89, 75],
       [88, 96, 61, 36],
       [35, 46, 40, 93]])
```

```
np.cumprod(b)
```

```
array([         47,        3666,       36660,     2162940,
168709320,
         182950048,  1385201856,  1507732736,  -325158144,
596803840,
        -234680832,   610786304,  1793432576,  1740685312,
302170112,
        1187921920,  1457913856, -1774190592,  -851443712,  -
587202560,
         922746880,  -503316480,  1342177280,   268435456])
```

```
np.cumprod(b,axis=0)
```

```
array([[         47,          78,          10,          59],
       [       3666,        4056,         780,        3540],
       [     260286,      206856,       10920,      120360],
       [    4424862,    16548480,      971880,     9027000],
       [  389387856,  1588654080,    59284680,   324972000],
       [  743673072,    63643648, -1923580096,   157624928]])
```

```
np.cumprod(b,axis=1)
```

```
array([[       47,      3666,    36660,  2162940],
       [       78,      4056,   316368, 18982080],
       [       71,      3621,    50694,  1723596],
       [       17,      1360,   121040,  9078000],
       [       88,      8448,   515328, 18551808],
       [       35,      1610,    64400,  5989200]])
```

np.percentile

numpy.percentile()function used to compute the nth percentile of the given data (array elements) along the specified axis.

https://numpy.org/doc/stable/reference/generated/numpy.percentile.html

```
a
```

```
array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])
```

```
np.percentile(a,100)   # 100 percentile dhund rahe hain to ofcourse
maximum number hoga
```

```
88.0
```

```
np.percentile(a,50)  # aadhe log 48 se aage hain and aadhe log piche
```

```
48.0
```

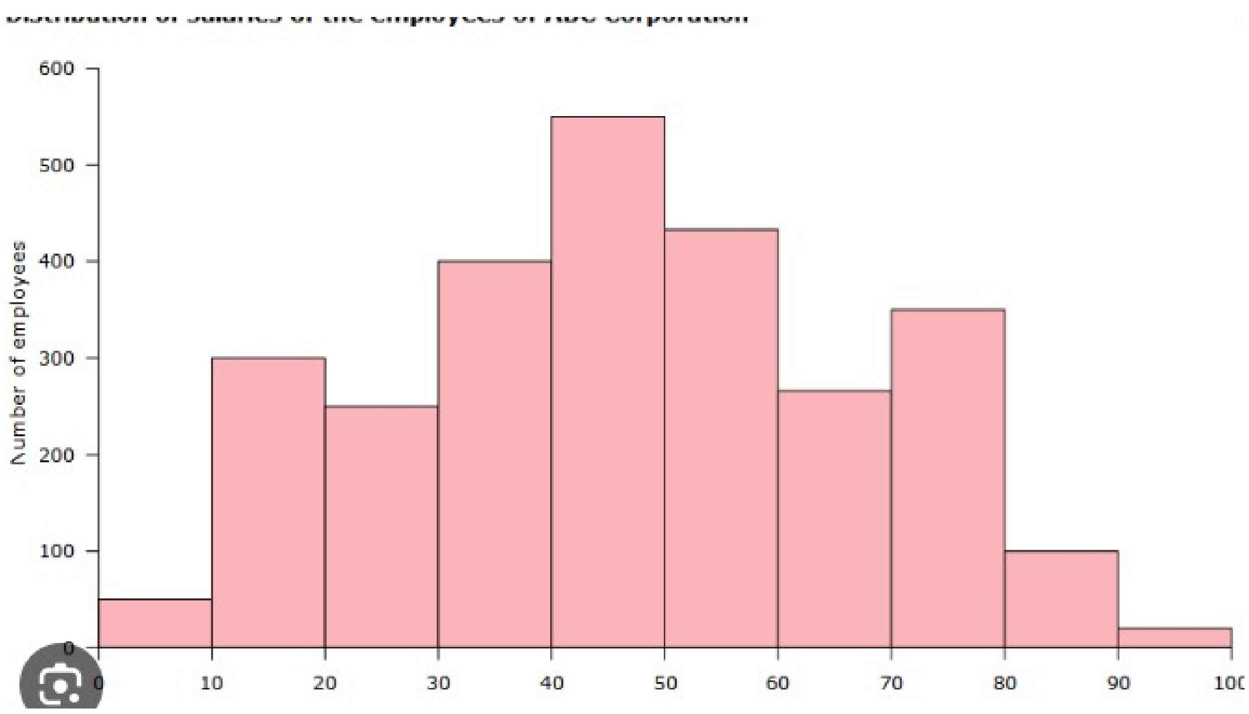
```
np.percentile(a,0)
```

```
12.0
```

np.histogram

Numpy has a built-in numpy.histogram() function which represents the frequency of data distribution in the graphical form.

https://numpy.org/doc/stable/reference/generated/numpy.histogram.html



```
a

array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])

np.histogram(a,bins=[0,10,20,30,40,50,60,70,80,90])  # bins basically
range batata hai ye return karega ki 0-10 ke bich me kitna data hai
phir 10-20 ke
                                                      #bich me kitna
and so on

(array([0, 2, 1, 4, 1, 2, 2, 0, 3], dtype=int64),
 array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90]))

np.histogram(a,bins=[0,50,100])

(array([8, 7], dtype=int64), array([  0,  50, 100]))
```

np.corrcoef

Return Pearson product-moment correlation coefficients.

https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html

```
salary = np.array([20000,40000,25000,35000,60000])
experience = np.array([1,3,2,4,2])

np.corrcoef(salary,experience)
```

```
array([[1.        , 0.25344572],
       [0.25344572, 1.        ]])
```

np.isin

With the help of numpy.isin() method, we can see that one array having values are checked in a different numpy array having different elements with different sizes.

https://numpy.org/doc/stable/reference/generated/numpy.isin.html

```
#isin multiple item ko ek sath search karne ke kaam aata hai
a
```

```
array([37, 88, 64, 35, 12, 54, 33, 28, 85, 33, 53, 17, 48, 63, 82])
```

```
items=[10,20,30,35,40,45,50,60,70,80,90,100]
```

```
a[np.isin(a,items)]
```

```
array([35])
```

np.flip

The numpy.flip() function reverses the order of array elements along the specified axis, preserving the shape of the array.

https://numpy.org/doc/stable/reference/generated/numpy.flip.html

```
a
```

```
array([38, 11, 89, 98, 46, 84, 33, 26, 54,  2, 12, 78, 34, 47, 70])
```

```
np.flip(a)  #just reverse kar dega
```

```
array([70, 47, 34, 78, 12,  2, 54, 26, 33, 84, 46, 98, 89, 11, 38])
```

```
b
```

```
array([[10, 25, 95, 51],
       [14, 24, 41, 66],
       [47, 10, 71, 10],
       [19, 90, 43, 40],
       [77, 56,  1, 22],
       [16, 10, 14, 54]])
```

```
np.flip(b)  # complete flip ho gya aap axis laga ke row aur column ko sirf flip kar sakte ho
```

```
array([[54, 14, 10, 16],
       [22,  1, 56, 77],
       [40, 43, 90, 19],
       [10, 71, 10, 47],
       [66, 41, 24, 14],
       [51, 95, 25, 10]])
```

```
np.flip(b,axis=0)
```

```
array([[16, 10, 14, 54],
       [77, 56,  1, 22],
       [19, 90, 43, 40],
       [47, 10, 71, 10],
       [14, 24, 41, 66],
       [10, 25, 95, 51]])
```

```
np.flip(b,axis=1)
```

```
array([[51, 95, 25, 10],
       [66, 41, 24, 14],
       [10, 71, 10, 47],
       [40, 43, 90, 19],
       [22,  1, 56, 77],
       [54, 14, 10, 16]])
```

np.put

The numpy.put() function replaces specific elements of an array with given values of p_array. Array indexed works on flattened array. (ye permanent change karta hai)

https://numpy.org/doc/stable/reference/generated/numpy.put.html

```
a
```

```
array([38, 11, 89, 98, 46, 84, 33, 26, 54,  2, 12, 78, 34, 47, 70])
```

```
# kisi element ko change kar sakte hain suppose 38 ko 110 me and 11 ko 
530 me change karna hai
np.put(a,[0,1],[110,530])  # 1st parameter batata hai ki kon se array 
me change karna hai 2nd batata hai kon kon se index pe change karna 
hai and 3rd batata hai ki change karke kya karna hai
```

```
a
```

```
array([110, 530,  89,  98,  46,  84,  33,  26,  54,   2,  12,  78,  
34,
        47,  70])
```

np.delete

The numpy.delete() function returns a new array with the deletion of sub-arrays along with the mentioned axis.

https://numpy.org/doc/stable/reference/generated/numpy.delete.html

```
#kisi particular index ko delete kar dega
a

array([110, 530,  89,  98,  46,  84,  33,  26,  54,   2,  12,  78,
34,
        47,  70])

np.delete(a,0)

array([530,  89,  98,  46,  84,  33,  26,  54,   2,  12,  78,  34,
47,
        70])

#deleting multiple index
np.delete(a,[2,4,7])

array([110, 530,  98,  84,  33,  54,   2,  12,  78,  34,  47,  70])
```

Set functions

- np.union1d
- np.intersect1d
- np.setdiff1d
- np.setxor1d
- np.in1d

```
# ye sare function array hi return karke dega
m = np.array([1,2,3,4,5])
n = np.array([3,4,5,6,7])

np.union1d(m,n)

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

np.intersect1d(m,n)

array([3, 4, 5])

np.setdiff1d(m,n) # wo sare item jo m me ho n me nahi

array([1, 2])

np.setxor1d(m,n)  # common ko hata dega baki sab rahne dega

array([1, 2, 6, 7])

np.in1d(m,5)   #ye check karta hai ki kisi particular array me koi
element exist karta hai ki nahi

array([False, False, False, False,  True])

m[np.in1d(m,5)]
```

```
array([5])
```

np.clip

numpy.clip() function is used to Clip (limit) the values in an array.

https://numpy.org/doc/stable/reference/generated/numpy.clip.html

```
a
```

```
array([110, 530,  89,  98,  46,  84,  33,  26,  54,   2,  12,  78,
34,
        47,  70])
```

```
# element ko ek range (clip) me rakh sakte hain jo bhi particular
range se bahar jayega usko range ke upper and lower element me convert
kar dega

np.clip(a,a_min=25,a_max=75)  #75 se jitna bhi jyada hai usko 75 me
and 25 se jitna bhi kam hai usko 25 me convert kar dega
```

```
array([75, 75, 75, 75, 46, 75, 33, 26, 54, 25, 25, 75, 34, 47, 70])
```