np.sort

Return a sorted copy of an array.

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

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

```
array([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

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

```
array([[12, 52, 42,  6],
       [29, 18, 47, 55],
       [61, 93, 83,  9],
       [38, 63, 44, 85],
       [ 8, 87, 31, 72],
       [40, 71,  2,  7]])
```

```
np.sort(a)[::-1]
```

```
array([94, 92, 78, 68, 53, 50, 38, 37, 30, 28, 21, 11,  9,  5,  2])
```

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

```
array([[ 8, 18,  2,  6],
       [12, 52, 31,  7],
       [29, 63, 42,  9],
       [38, 71, 44, 55],
       [40, 87, 47, 72],
       [61, 93, 83, 85]])
```

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

```
# code
np.append(a,200)
```

```
array([ 11,  53,  28,  50,  38,  37,  94,  92,   5,  30,  68,   9,  78,
         2,  21, 200])
```

```
b
```

```
array([[12, 52, 42,  6],
       [29, 18, 47, 55],
       [61, 93, 83,  9],
       [38, 63, 44, 85],
       [ 8, 87, 31, 72],
       [40, 71,  2,  7]])
```

```
np.append(b,np.random.random((b.shape[0],1)),axis=1)
```

```
array([[12.        , 52.        , 42.        ,  6.        ,  0.22006275],
       [29.        , 18.        , 47.        , 55.        ,  0.81740634],
       [61.        , 93.        , 83.        ,  9.        ,  0.89146072],
       [38.        , 63.        , 44.        , 85.        ,  0.84519124],
       [ 8.        , 87.        , 31.        , 72.        ,  0.24007274],
       [40.        , 71.        ,  2.        ,  7.        ,  0.48056374]])
```

np.concatenate

numpy.concatenate() function concatenate a sequence of arrays along an existing axis.

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

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

print(c)
print(d)
```

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

```
np.concatenate((c,d),axis=0)
```

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

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

```
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/

```
# code
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

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

```
# code
a.shape
```

```
(15,)
```

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

```
(1, 15)
```

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

```
array([[11],
       [53],
       [28],
       [50],
       [38],
       [37],
       [94],
       [92],
       [ 5],
       [30],
       [68],
       [ 9],
       [78],
       [ 2],
       [21]])
```

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([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

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

```
(array([ 1,  6,  7, 10, 12]),)
```

```
# replace all values > 50 with 0
np.where(a>50,0,a)
```

```
array([11,  0, 28, 50, 38, 37,  0,  0,  5, 30,  0,  9,  0,  2, 21])
```

```
np.where(a%2 == 0,0,a)
```

```
array([11, 53,  0,  0,  0, 37,  0,  0,  5,  0,  0,  9,  0,  0, 21])
```

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

```
# code
a
```

```
array([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

```
np.argmax(a)
```

```
6
```

```
b
```

```
array([[12, 52, 42,  6],
       [29, 18, 47, 55],
       [61, 93, 83,  9],
       [38, 63, 44, 85],
       [ 8, 87, 31, 72],
       [40, 71,  2,  7]])
```

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

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

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

```
array([1, 3, 1, 3, 1, 1])
```

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

```
13
```

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

```
a
```

```
array([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

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

```
array([ 11,  64,  92, 142, 180, 217, 311, 403, 408, 438, 506, 515, 593,
       595, 616])
```

```
b
```

```
array([[12, 52, 42,  6],
       [29, 18, 47, 55],
       [61, 93, 83,  9],
       [38, 63, 44, 85],
       [ 8, 87, 31, 72],
       [40, 71,  2,  7]])
```

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

```
array([[ 12,  64, 106, 112],
       [ 29,  47,  94, 149],
       [ 61, 154, 237, 246],
       [ 38, 101, 145, 230],
       [  8,  95, 126, 198],
       [ 40, 111, 113, 120]])
```

```
np.cumsum(b)
```

```
array([  12,   64,  106,  112,  141,  159,  206,  261,  322,  415,  498,
        507,  545,  608,  652,  737,  745,  832,  863,  935,  975, 1046,
       1048, 1055])
```

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

```
array([                  11,                  583,                16324,
                     816200,             31015600,           1147577200,
                107872256800,        9924247625600,       49621238128000,
            1488637143840000,   101227325781120000,   911045932030080000,
           -2725393596491966464, -5450787192983932928, -3786066610405281792])
```

```
a
```

```
array([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

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([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

```
np.percentile(a,50)
```

```
37.0
```

```
np.median(a)
```

```
37.0
```

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

```
# code
a
```

```
array([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

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

```
(array([9, 6]), 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

```
# code
a
```

```
array([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

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

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

```
array([50, 30])
```

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

```
# code
a
```

```
array([11, 53, 28, 50, 38, 37, 94, 92,  5, 30, 68,  9, 78,  2, 21])
```

```
np.flip(a)
```

```
array([21,  2, 78,  9, 68, 30,  5, 92, 94, 37, 38, 50, 28, 53, 11])
```

```
b
```

```
array([[12, 52, 42,  6],
       [29, 18, 47, 55],
       [61, 93, 83,  9],
       [38, 63, 44, 85],
       [ 8, 87, 31, 72],
       [40, 71,  2,  7]])
```

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

```
array([[ 6, 42, 52, 12],
       [55, 47, 18, 29],
       [ 9, 83, 93, 61],
       [85, 44, 63, 38],
       [72, 31, 87,  8],
       [ 7,  2, 71, 40]])
```

np.put

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

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

```
# code
a
```

```
array([110, 530,  28,  50,  38,  37,  94,  92,   5,  30,  68,   9,  78,
         2,  21])
```

```
np.put(a,[0,1],[110,530])
```

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

```
# code
a
```

```
array([110, 530,  28,  50,  38,  37,  94,  92,   5,  30,  68,   9,  78,
         2,  21])
```

```
np.delete(a,[0,2,4])
```

```
array([530,  50,  37,  94,  92,   5,  30,  68,   9,  78,   2,  21])
```

Set functions

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

```
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(n,m)
```

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

```
np.setxor1d(m,n)
```

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

```
m[np.in1d(m,1)]
```

```
array([1])
```

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

```
# code
a
```

```
array([110, 530,  28,  50,  38,  37,  94,  92,   5,  30,  68,   9,  78,
         2,  21])
```

```
np.clip(a,a_min=25,a_max=75)
```

```
array([75, 75, 28, 50, 38, 37, 75, 75, 25, 30, 68, 25, 75, 25, 25])
```

```
# 17. np.swapaxes
```

```
# 18. np.uniform
```

```
# 19. np.count_nonzero
```

```
# 21. np.tile
# https://www.kaggle.com/code/abhayparashar31/best-numpy-functions-for-data-science-50?scriptVersionId=98816580


# 22. np.repeat
# https://towardsdatascience.com/10-numpy-functions-you-should-know-1dc4863764c5



# 25. np.allclose and equals
```