1. Write a NumPy program to get the numpy version and show numpy build configuration.

import numpy as np

print(np.\_\_version\_\_)

print(np.show\_config())

2.Write a NumPy program to get help on the add function.

help(np.add)

Help on ufunc:

add = <ufunc 'add'>

add(x1, x2, /, out=None, \*, where=True, casting='same\_kind', order='K', dtype=None, subok=True[, signature, extobj])

Add arguments element-wise.

Parameters

----------

x1, x2 : array\_like

The arrays to be added.

If ``x1.shape != x2.shape``, they must be broadcastable to a common

shape (which becomes the shape of the output).

out : ndarray, None, or tuple of ndarray and None, optional

A location into which the result is stored. If provided, it must have

a shape that the inputs broadcast to. If not provided or None,

a freshly-allocated array is returned. A tuple (possible only as a

keyword argument) must have length equal to the number of outputs.

where : array\_like, optional

This condition is broadcast over the input. At locations where the

condition is True, the `out` array will be set to the ufunc result.

Elsewhere, the `out` array will retain its original value.

Note that if an uninitialized `out` array is created via the default

``out=None``, locations within it where the condition is False will

remain uninitialized.

\*\*kwargs

For other keyword-only arguments, see the

:ref:`ufunc docs <ufuncs.kwargs>`.

Returns

-------

add : ndarray or scalar

The sum of `x1` and `x2`, element-wise.

This is a scalar if both `x1` and `x2` are scalars.

Notes

-----

Equivalent to `x1` + `x2` in terms of array broadcasting.

Examples

--------

>>> np.add(1.0, 4.0)

5.0

>>> x1 = np.arange(9.0).reshape((3, 3))

>>> x2 = np.arange(3.0)

>>> np.add(x1, x2)

array([[ 0., 2., 4.],

[ 3., 5., 7.],

[ 6., 8., 10.]])

The ``+`` operator can be used as a shorthand for ``np.add`` on ndarrays.

>>> x1 = np.arange(9.0).reshape((3, 3))

>>> x2 = np.arange(3.0)

>>> x1 + x2

array([[ 0., 2., 4.],

[ 3., 5., 7.],

[ 6., 8., 10.]])

3.Write a NumPy program to test whether none of the elements of a given array is zero.

l=[1,2,3]

a=np.array(l)

if 0 in a:

print('zero present in given numpy array')

else:

print('no zero is there in the given numpy array')

no zero is there in the given numpy array

4.Write a NumPy program to create an array of 10 zeros, 10 ones, 10 fives

l=[0]\*10

l.extend([1]\*10)

l.extend([5]\*10)

b=np.array(l)

print(b)

[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 5 5 5 5 5 5 5 5 5 5]

5.Write a NumPy program to create an array of all the even integers from 30 to 70

li=[]

for i in range(30,71,2):

li.append(i)

even=np.array(li)

print(even)

[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]

6.Write a NumPy program to create a 3x3 identity matrix

print(np.array([1]\*9).reshape(3,3))

[[1 1 1]

[1 1 1]

[1 1 1]]

7.Write a NumPy program to create a vector with values ​​from 0 to 20 and change the sign of the numbers in the range from 9 to 15.

import numpy as np

x = np.arange(21)

print("Original vector:")

print(x)

print("After changing the sign of the numbers in the range from 9 to 15:")

x[(x >= 9) & (x <= 15)] \*= -1

print(x)

Output:

Original vector:

[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

After changing the sign of the numbers in the range from 9 to 15:

[ 0 1 2 3 4 5 6 7 8 -9 -10 -11 -12 -13 -14 -15 16 17

18 19 20]

8.Write a NumPy program to find the number of rows and columns of a given matrix.

test=[[1 ,1],

[1 ,1 ],

[1, 1]]

print('no of rows:',len(np.array(test)))

print('no of columns:',len(np.array(test)[0]))

no of rows: 3

no of columns: 2

1. 9. Write a NumPy program to create a 10x10 matrix, in which the elements on the borders will be equal to 1, and inside 0.

x = np.ones((10, 10))

x[1:-1, 1:-1] = 0

print(x)

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

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

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

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

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

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

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

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

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

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

1. 10. Write a NumPy program to compute sum of all elements, sum of each column and sum of each row of a given array.

sumarr=np.array([[1,2,3],[4,5,6],[7,8,9]]).reshape(3,3)

print('sum of 1st row:',sum(sumarr[0]),'\nsum of 2nd row:',sum(sumarr[1]),'\nsum of 3rd row:',sum(sumarr[2]))

print('sum of 1st column:',sum(sumarr)[0],'\nsum of 2nd row:',sum(sumarr)[1],'\nsum of 3rd row:',sum(sumarr)[2])

print('sum of given arrayis :',sum(sumarr[0])+sum(sumarr[1])+sum(sumarr[2]))

sum of 1st row: 6

sum of 2nd row: 15

sum of 3rd row: 24

sum of 1st column: 12

sum of 2nd row: 15

sum of 3rd row: 18

sum of given arrayis : 45

11. Write a NumPy program to convert a given list into an array, then again convert it into a list. Check initial list and final list are equal or not.

l11=[[1,2,3],[4,5,6],[7,8,9]]

arr11=np.array(l11)

print(arr11)

print(list(arr11))

[[1 2 3]

[4 5 6]

[7 8 9]]

[array([1, 2, 3]), array([4, 5, 6]), array([7, 8, 9])]

12. Write a NumPy program to create a 3x3x3 array filled with arbitrary values.

print(np.ones((3,3,3)))

[[[1. 1. 1.]

[1. 1. 1.]

[1. 1. 1.]]

[[1. 1. 1.]

[1. 1. 1.]

[1. 1. 1.]]

[[1. 1. 1.]

[1. 1. 1.]

[1. 1. 1.]]]

13. Write a NumPy program to create a 5x5 zero matrix with elements on the main diagonal equal to 1, 2, 3, 4, 5

l13=np.zeros((5,5))

print('before:\n',l13)

for i in range(len(l13)):

l13[i,i]=i+1

print('after :\n',l13)

before:

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

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

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

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

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

after :

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

[0. 2. 0. 0. 0.]

[0. 0. 3. 0. 0.]

[0. 0. 0. 4. 0.]

[0. 0. 0. 0. 5.]]

14. Write a NumPy program to extract all numbers from a given array which are less and greater than a specified number.

l14=np.array([[1,2,3],[4,5,6],[7,8,9]])

less=[]

greater=[]

num=int(input('enter a number:'))

for j in l14:

for i in j:

if i>num:

greater.append(i)

else:

less.append(i)

print('elements in the given array lessthan or equal to',num,'are:\n')

for i in range(len(less)):

print(less[i],'\t')

print('elements in the given array greaterthan',num,'are:\n')

for i in range(len(greater)):

print(greater[i],'\t')

enter a number:4

elements in the given array lessthan or equal to 4 are:

1

2

3

4

elements in the given array greaterthan 4 are:

5

6

7

8

9

15. Write a NumPy program to compute the sum of the diagonal element of a given array.

l15=np.array([[1,2,3],[4,5,6],[7,8,9]])

print(l15)

s=0

print('sum of diagonal elements:')

for i in range(len(l15)):

s+=l15[i,i]

print(s)

[[1 2 3]

[4 5 6]

[7 8 9]]

sum of diagonal elements:

15

16. Get the common items between two arrays.

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

b = np.array([7,2,10,2,7,4,9,4,9,8])

Desired Output:

array([2, 4])

l16a = np.array([1,2,3,2,3,4,3,4,5,6])

l16b = np.array([7,2,10,2,7,4,9,4,9,8])

common=[]

for i in l16a:

if i in l16b:

common.append(i)

print('common elements among given two arrays are:')

print(np.array(list(set(common))))

common elements among given two arrays are:

[2 4]