How to install numpy in your system for first time

```
!conda install --yes numpy
```

```
Collecting package metadata (current_repodata.json): ...working...
done
Solving environment: ...working... done

# All requested packages already installed.
```

Importing numpy

```
import numpy as np
```

Creating 1D array from list

```
mylist = [1,2,3,4]
mylist, type(mylist)
```

```
([1, 2, 3, 4], list)
```

```
myarr = np.array(mylist)
myarr, type(myarr)
```

```
(array([1, 2, 3, 4]), numpy.ndarray)
```

Creating 2D array from list

```
mymatrix = [[1,2,3],[4,5,6],[7,8,9]] # Here we create matrix , matrix
is list inside list for 1D
mymatrix, type(mymatrix)              # Here we check the type of
mymatrix thats also a lsit
```

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

```
mymatrixarr = np.array(mymatrix)
mymatrixarr, type(mymatrixarr) # Here we can see that create the 2
dimension array and type is also matrix
```

```
(array([[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]),
 numpy.ndarray)
```

We can create array directly as well

```
np.arange(0,21,2)  # Here we use a range command, here is 3 perimeter
first is start , second is end and third is step size
```

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

Creating Array of zero, one or any number

```
np.zeros(5) # Here 5 zeros but that 0. thats float numpy return as
float
```

```
array([0., 0., 0., 0., 0.])
```

```
np.zeros((5,3)) # Here it create zero matrix of 5,3 5 rows and 3 columns
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
np.ones((2,3)) # here is matrix of 1 , There is only 0 and 1 array available , but 1*5 we can create any
```

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

```
np.ones(10)*5 # Here array of 5 same like this we can create any number array
```

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

Identity matirx :That is sqaure matrix

```
np.eye(5) # here is 5 by 5 Idenity Matrix and diagonal element is 1 rest are zeros
```

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

Linearly space number

```
np.linspace(0,10,11) # Here we can see 11 number are linealy at at equal space from nearby numbers
```

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

```
np.linspace(0,10,10) # Here there are 10 so they are in float
```

```
array([ 0.        ,  1.11111111,  2.22222222,  3.33333333,
4.44444444,
       5.55555556,  6.66666667,  7.77777778,  8.88888889,
10.        ])
```

Random Number between 0 to 1

```
np.random.rand(2) # Here is giving 2 random numbers between zero and 1
```

```
array([0.24817465, 0.83862759])
```

```
np.random.rand(2,5) # Here we are create random number between 0 to 1 in form of matrix
```

```
array([[0.39101794, 0.91363615, 0.84499994, 0.11684269, 0.58424501],
       [0.90666546, 0.0817708 , 0.11114417, 0.13968912, 0.12015077]])
```

Random Number between 0 to 1 , with standard normal distribution

```
np.random.randn(5) # "standard normal" distribution where mean(avg) 0 and variance 1 so there would be -ve number too
```

```
array([ 0.59096167,  1.3304227 ,  1.7836412 ,  0.90679955, -0.92197213])
```

```
np.random.randn(5,2) # Here we are create standard normal distribution between 0 to 1 in form of matrix
```

```
array([[-1.05148119,  0.60168389],
       [-1.7198416 , -0.14386829],
       [ 0.17185619, -0.51937709],
       [ 1.21211037,  0.00422937],
       [-1.78746815,  0.89133712]])
```

Random integer between certain numbers

```
np.random.randint(0,101,5) # Here we will get random 5 integer bwtween o to 100
```

```
array([48,  2, 85,  6,  4])
```

```
np.random.randint(0,101,(5,2)) # Here we will get random 5 integer bwtween o to 100 in form of matrix
```

```
array([[84, 12],
       [ 1, 82],
       [41, 95],
       [53, 61],
       [61, 63]])
```

Here we are setting seed number so we get same result on random by hitting seed

```
np.random.seed(42) # Here we give any random number as 42 we can set same result as we hit 42 with seed
np.random.randint(0,101,3)
```

```
array([51, 92, 14])
```

```
np.random.seed(42) # Still same result
np.random.randint(0,101,3)
```

```
array([51, 92, 14])
```

```
np.random.seed(42) # work on matrix as well
np.random.randint(0,101,(3,4))
```

```
array([[51, 92, 14, 71],
       [60, 20, 82, 86],
       [74, 74, 87, 99]])
```

```
np.random.seed(42) # Still same result
np.random.randint(0,101,(3,4))
```

```
array([[51, 92, 14, 71],
       [60, 20, 82, 86],
       [74, 74, 87, 99]])
```

Reshape

```
arr = np.arange(0,24)
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19, 20, 21, 22, 23])
```

```
arr.reshape(3,8) # Here we reshape 1D array into a matrix
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20, 21, 22, 23]])
```

```
arr.reshape(3,5) # We get error because you cant fit 24 items in 3*5 =
15 , so make sure about size and item number
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_2404\2322086622.py in <module>
----> 1 arr.reshape(3,5)

ValueError: cannot reshape array of size 24 into shape (3,5)
```

```
arr.shape # .shape is use to find shape of array, (24,) means thats 1D
and havinh only 1 row we will know in detail later
```

```
(24,)
```

```
arr.reshape(3,8)
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20, 21, 22, 23]])
```

```
arr1 = np.random.randint(2,10,(3,4))
arr1
```

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

```
ar2=arr1.reshape(1,12)
ar2
```

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

```
ar2.shape
```

```
(2, 6)
```

Some more functions like max , min, argmax , argmin

```
ranarr = np.random.randint(0,101,5)
ranarr
```

```
array([70, 14, 25, 26, 89])
```

```
ranarr.sum()
```

```
224
```

```
ranarr.mean()
```

```
44.8
```

```
ranarr.var() # variance
```

```
856.5600000000001
```

```
ranarr.std()  # standard devation
```

```
29.267046314925597
```

```
ranarr.max()
```

```
89
```

```
ranarr.min()
```

```
14
```

```
ranarr.argmax() # to find the place of maximum item 0,1,2
```

```
4
```

```
ranarr.argmin() # to find the place of minimum item 0
```

```
1
```

Indexing and Selecting

```
arr = np.arange(0,11)
arr
```

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

```
arr[8]
```

```
8
```

```
arr[2:5] # Here it start from 2 and end at 5 but 5 is not includes
```

```
array([2, 3, 4])
```

```
arr[:5] # arr[0:5] Both are same
```

```
array([0, 1, 2, 3, 4])
```

```
arr[5:] # it show till last
```

```
array([ 5,  6,  7,  8,  9, 10])
```

```
arr[5:-1] # This doesnt till last item
```

```
array([5, 6, 7, 8, 9])
```

Here we are going to change more than one item at once , Broadcasting 1 value

```
arr[0:5] = 100
arr
```

```
array([100, 100, 100, 100, 100,   5,   6,   7,   8,   9,  10])
```

Slicing from array

```
arr1=np.arange(0,11)
arr1
```

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

```
slice_of_arr=arr1[0:5] # here we slice 0 to 5 but 5 is exclude
slice_of_arr
```

```
array([0, 1, 2, 3, 4])
```

```
slice_of_arr[:] = 99 # That means all items in array
slice_of_arr
```

```
array([99, 99, 99, 99, 99])
```

```
arr1 # Note : That slicing effect normal array as well, as seen in
next
```

```
array([99, 99, 99, 99, 99,  5,  6,  7,  8,  9, 10])
```

```
arr1_copy = arr1.copy() # Here we have create a copy og your array
```

```
arr1_copy[:] = 80 # here we have set all values at 80
arr1_copy
```

```
array([80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80])
```

```
arr1 # and here our backup copy that still there
```

```
array([99, 99, 99, 99, 99,  5,  6,  7,  8,  9, 10])
```

Indexing on 2D matrix

```
arr_2d = np.array([[10,20,30],[40,50,60],[70,80,90]]) # Here we create 2D matrix by array
arr_2d
```

```
array([[10, 20, 30],
       [40, 50, 60],
       [70, 80, 90]])
```

```
arr_2d.shape # Here is 3 rows by 3 columns so for call raw we can use arr_2d[0] for first row
```

```
(3, 3)
```

```
arr_2d[2] # extract 3rd row from array
```

```
array([70, 80, 90])
```

```
arr_2d[2,1] # or arr_2d[2][1] for grabbing single value from whole matrix
```

```
80
```

```
arr_2d[:2,1:] # Here we grab subsection of matrix
```

```
array([[20, 30],
       [50, 60]])
```

Sum in matrix array

```
arr_2d
```

```
array([[10, 20, 30],
       [40, 50, 60],
       [70, 80, 90]])
```

```
arr_2d.sum()
```

```
450
```

```
arr_2d.sum(axis=0) # here sum is column wise top to bottom
```

```
array([120, 150, 180])
```

```
arr_2d.sum(axis=1) # here sum is row wise left to right
```

```
array([ 60, 150, 240])
```

condition in array

```
ar3 = np.random.randint(0,10,10)
ar3
```

```
array([3, 8, 2, 4, 2, 6, 4, 8, 6, 1])
```

```
bool_ar = ar3 > 4 # It check each item one by one
bool_ar
```

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

```
ar3[bool_ar] # Here it extract only those which are true,
```

```
array([8, 6, 8, 6])
```

```
ar3[ar3>4] # Here we see that item in ar3 which are greater than 4
```

```
array([8, 6, 8, 6])
```

Mathmatical operations in array

```
ar4 = np.arange(0,11,1)
ar4
```

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

```
ar4 + 5 # adding
```

```
array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

```
ar4 - 10
```

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

```
ar4 * 10
```

```
array([  0,  10,  20,  30,  40,  50,  60,  70,  80,  90, 100])
```

Array with array

```
ar4 + ar4 # array can be perfrom mathamatical operations with array as
well
```

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

```
ar4 * ar4
```

```
array([  0,   1,   4,   9,  16,  25,  36,  49,  64,  81, 100])
```

```
ar4/ar4 # here first item is 0 so 0/0 is nan(not a number) , its show
nan in numpy else it throw division by zero error
```

```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_1976\3399568524.py:1:
RuntimeWarning: invalid value encountered in true_divide
  ar4/ar4 # here first item is 0 so 0/0 is nan(not a number) , its
show nan in numpy else it throw division by zero error
```

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

```
ar4 / 0 # Here we still got result with warning because 0/0 is = nan (
not a number)
```

```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_1976\3844087298.py:1:
RuntimeWarning: divide by zero encountered in true_divide
  ar4 / 0 # Here we still got result with warning because 0/0 is = nan
( not a number)
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_1976\3844087298.py:1:
RuntimeWarning: invalid value encountered in true_divide
  ar4 / 0 # Here we still got result with warning because 0/0 is = nan
( not a number)
```

```
array([nan, inf, inf, inf, inf, inf, inf, inf, inf, inf, inf])
```

```
0/0
```

```
---------------------------------------------------------------------
-----
ZeroDivisionError                         Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_1976\182040962.py in <module>
----> 1 0/0

ZeroDivisionError: division by zero
```

```
1/0
```

```
---------------------------------------------------------------------
-----
ZeroDivisionError                         Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_1976\2354412189.py in <module>
----> 1 1/0

ZeroDivisionError: division by zero
```

```
1/ar4 # Here 1/0 that is inf(infinite) normally it give division by
zero error by in numpy it show inf with
    # warring divide by zero e
```

```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_1976\2823468765.py:1:
RuntimeWarning: divide by zero encountered in true_divide
  1/ar4 # Here 1/0 that is inf(infinite) normally it give division by
zero error by in numpy it show inf with
```

```
array([       inf, 1.        , 0.5       , 0.33333333, 0.25      ,
       0.2       , 0.16666667, 0.14285714, 0.125     , 0.11111111,
       0.1       ])
```

But make sure that both array have equal number of items

```
ar5=np.arange(0,5,1)
ar5
```

```
array([0, 1, 2, 3, 4])
```

```
ar4 + ar5 # When there are not equal number of item in both array
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
~\AppData\Local\Temp\ipykernel_1976\146937153.py in <module>
----> 1 ar4 + ar5 # When there are not equal number of item in both
array

ValueError: operands could not be broadcast together with shapes (11,)
(5,)
```

Numpy can perform complex operation as well

```
np.sqrt(ar4)
```

```
array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ,
       2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.        ,
       3.16227766])
```

```
np.sin(ar4)
```

```
array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -
0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,
0.41211849,
       -0.54402111])
```

```
np.log(ar4) # as log0 is undefine so it show inf
```

```
C:\Users\Chromsy\AppData\Local\Temp\ipykernel_1976\3094053908.py:1:
RuntimeWarning: divide by zero encountered in log
  np.log(ar4) # as log0 is undefine so it show inf
```

```
array([      -inf, 0.        , 0.69314718, 1.09861229, 1.38629436,
       1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458,
       2.30258509])
```