In [1]:
```
import numpy as np
import pandas as pd
import os
import sys
import math
import random
import warnings
```

In [2]:
```
dir('warnings')
```

Out[2]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
```

```
 'partition',
 'removeprefix',
 'removesuffix',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

What is Numpy?

multi dim arrays. 1D, 2D, ...etc.

Lists are slow. --->

Numpy arrays are faster. ----->

5 ---> 00000101 ===> (1byte)

Numpy --> Int32 (4bytes)

5 ----> Int 16, 32, 64 (2, 4, 8 bytes)

---> It is faster to read fewer bytes of memory. ---> No type checking when iterating through objects.

--> stores in contiguous memory locations. --> Benefits: SIMD (single instruction multiple data) vector processing. --> Effective cache utilization.

Lists

size--> 4 bytes reference count--> 8 bytes object type--> 8 bytes object value--> 8 bytes

--> list elements stored in random memory locations. -- with different pointers.

Differences:

Lists: --> Insertion, Deletion, appending, concatenation, etc.

Numpy: --> Insertion, Deletion, appending, concatenation. etc. manv more we can do.

#Examples:

In [12]:
```
a= [1,2,3,4]
b = [3,4,5,6]
print(a+b)
print(a*b)
print(a-b)
print(a/b)
```

```
[1, 2, 3, 4, 3, 4, 5, 6]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[12], line 4
      2 b = [3,4,5,6]
      3 print(a+b)
----> 4 print(a*b)
      5 print(a-b)
      6 print(a/b)

TypeError: can't multiply sequence by non-int of type 'list'
```

In [14]:
```
x = np.array([1,2,3,4])
y = np.array([3,4,5,6])
print(x+y)
print(x*y)
print(x/y)
print(x-y)
```

```
[ 4  6  8 10]
[ 3  8 15 24]
[0.33333333 0.5        0.6        0.66666667]
[-2 -2 -2 -2]
```

In [22]:
```
aa=x.tolist()
```

In [24]:
```
a
```

Out[24]:
```
[1, 2, 3, 4]
```

In [34]:
```
type(x)
```

Out[34]:
```
numpy.ndarray
```

In [36]:
```
len(x)
```

Out[36]:
```
4
```

In [42]:
```
# Applications of Numpy: --> MATLAB replacement
# Plotting (matplotlib)
#Backend (Pandas, connet4, digital photography)
# Machine Learning (Tensorflow, scikitlearn)
```

In [44]:
```
a
```

Out[44]:
```
[1, 2, 3, 4]
```

In [46]:
```
x
```

Out[46]:
```
array([1, 2, 3, 4])
```

In [48]:
```
b
```

Out[48]:
```
[3, 4, 5, 6]
```

In [50]:
```
y
```

Out[50]:
```
array([3, 4, 5, 6])
```

In [52]:
```
print(sys.getsizeof(a))
print(sys.getsizeof(b))
print(x.nbytes)
print(y.nbytes)
```

```
88
88
16
16
```

In [54]:
```
x.dtype
```

Out[54]:
```
dtype('int32')
```

In [56]:
```
y.dtype
```

Out[56]:
```
dtype('int32')
```

In [62]:
```
x = np.array([1,2,3,4], dtype = 'int64')
y = np.array([3,4,5,6], dtype = 'int32')
```

In [64]:
```
x.nbytes
```

Out[64]:
```
32
```

In [66]:
```
y.nbytes
```

Out[66]:
```
16
```

In [68]:
```
x.itemsize
```

Out[68]:
```
8
```

In [70]:
```
x.size
```

Out[70]:
```
4
```

In [74]:
```
y = np.array([3,4,5,6], dtype = 'int16')
```

In [76]:
```
y.shape
```

Out[76]:
```
(4,)
```

In [78]:
```
y
```

Out[78]:
```
array([3, 4, 5, 6], dtype=int16)
```

In [80]:
```
y.ndim
```

Out[80]:
```
1
```

In [82]:
```
z = np.array([[1,2,3], [3,4,5]])
```

In [84]:
```
z
```

Out[84]:
```
array([[1, 2, 3],
       [3, 4, 5]])
```

In [86]:
```
type(z)
```

Out[86]:
```
numpy.ndarray
```

In [88]:
```
print(type(z))
```

```
<class 'numpy.ndarray'>
```

In [90]:
```
print(z)
```

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

In [92]:
```
z.itemsize
```

Out[92]:
```
4
```

In [94]:
```
y.itemsize
```

Out[94]:
```
2
```

In [96]:
```
z.shape
```

Out[96]:
```
(2, 3)
```

In [98]:
```
z.size
```

Out[98]:
```
6
```

In [100]:
```
z.nbytes
```

Out[100]:
```
24
```

Accessing/ changing specific elements, rows, and columns etc

In [117]:
```
an = np.array([[2,3,4,4,5,6],[5,6,9, 7, 34, 98]])
```

In [119]:
```
an
```

Out[119]:
```
array([[ 2,  3,  4,  4,  5,  6],
       [ 5,  6,  9,  7, 34, 98]])
```

In [121]:
```
print(an)
```

```
[[ 2  3  4  4  5  6]
 [ 5  6  9  7 34 98]]
```

In [127]:
```
an[1:]
```

Out[127]:
```
array([[ 5,  6,  9,  7, 34, 98]])
```

In [129]:
```
an[1:2]
```

Out[129]:
```
array([[ 5,  6,  9,  7, 34, 98]])
```

In [131]:
```
an[1, 1]
```

Out[131]:
```
6
```

In [133]:
```
an[1, 3]
```

Out[133]:
```
7
```

In [135]:
```
an[0, 1]
```

Out[135]:
```
3
```

In [137]:
```
an[0:, 1]
```

Out[137]:
```
array([3, 6])
```

In [139]:
```
an
```

Out[139]:
```
array([[ 2,  3,  4,  4,  5,  6],
       [ 5,  6,  9,  7, 34, 98]])
```

In [161]:
```
an[1][2]
```

Out[161]:
```
9
```

In [173]:
```
an[1,]
```

Out[173]:
```
array([ 5,  6,  9,  7, 34, 98])
```

In [175]:
```
an[0,]
```

Out[175]:
```
array([2, 3, 4, 4, 5, 6])
```

In [177]:
```
an[0,0]
```

Out[177]:
```
2
```

In [179]:
```
an[0, 5]
```

Out[179]:
```
6
```

In [189]:
```
n=an[0:3, 1:3]
```

In [187]:
```
an
```

Out[187]:
```
array([[ 2,  3,  4,  4,  5,  6],
       [ 5,  6,  9,  7, 34, 98]])
```

In [191]:
```
n
```

Out[191]:
```
array([[3, 4],
       [6, 9]])
```

In [195]:
```
m = np.array([2,3])
```

In [197]:
```
n*m
```

Out[197]:
```
array([[ 6, 12],
       [12, 27]])
```

In [199]:
```
an[0, 1:]
```

Out[199]:
```
array([3, 4, 4, 5, 6])
```

In [201]:
```
an[1, 4:]
```

Out[201]:
```
array([34, 98])
```

In [203]:
```
an[1,2]=333
```

In [205]:
```
an
```

Out[205]:
```
array([[  2,   3,   4,   4,   5,   6],
       [  5,   6, 333,   7,  34,  98]])
```

In [241]:
```
t = np.array([[[1,2,3], [26,6,5]], [[2,5,6], [2,4,5]]])
```

In [243]:
```
type(t)
```

Out[243]:
```
numpy.ndarray
```

In [245]:
```
t.dtype
```

Out[245]:
```
dtype('int32')
```

In [247]:
```
t
```

Out[247]:
```
array([[[ 1,  2,  3],
        [26,  6,  5]],

       [[ 2,  5,  6],
        [ 2,  4,  5]]])
```

In [249]:
```
print(t)
```

```
[[[ 1  2  3]
  [26  6  5]]

 [[ 2  5  6]
  [ 2  4  5]]]
```

In [251]:
```
t[0, 1,1]=90
```

In [253]:
```
t
```

Out[253]:
```
array([[[ 1,  2,  3],
        [26, 90,  5]],

       [[ 2,  5,  6],
        [ 2,  4,  5]]])
```

In [258]:
```
t[1,1,1]
```

Out[258]:
```
4
```

In [260]:
```
t[:, 1, :]=[[3,3,3],[4,4,4]]
```

In [262]:
```
t
```

Out[262]:
```
array([[[1, 2, 3],
        [3, 3, 3]],

       [[2, 5, 6],
        [4, 4, 4]]])
```

Initializing different types of Arrays

In [267]:
```
np.zeros(3)
```

Out[267]:
```
array([0., 0., 0.])
```

In [271]:
```
np.zeros(2,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[271], line 1
----> 1 np.zeros(2,2)

TypeError: Cannot interpret '2' as a data type
```

In [289]:
```
np.zeros((2,2))
```

Out[289]:
```
array([[0., 0.],
       [0., 0.]])
```

In [291]:
```
x=np.zeros([2,2])
```

In [293]:
```
x
```

Out[293]:
```
array([[0., 0.],
       [0., 0.]])
```

In [295]:
```
x.itemsize
```

Out[295]:
```
8
```

In [297]:
```
x.dtype
```

Out[297]:
```
dtype('float64')
```

In [299]:
```
print(x)
```

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

In [303]:
```
np.zeros((2,3))
```

Out[303]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [307]:
```
np.zeros((2,3), dtype = 'int32')
```

Out[307]:
```
array([[0, 0, 0],
       [0, 0, 0]])
```

In [311]:
```
np.ones(3)
```

Out[311]:
```
array([1., 1., 1.])
```

In [313]:
```
np.ones((2,8))
```

Out[313]:
```
array([[1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.]])
```

In [315]:
```
np.eye(2,4)
```

Out[315]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.]])
```

In [317]:
```
np.eye(6,6)
```

Out[317]:
```
array([[1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1.]])
```

In [319]:
```
np.full(2,4)
```

Out[319]:
```
array([4, 4])
```

In [323]:
```
np.full((2,4), 3)
```

Out[323]:
```
array([[3, 3, 3, 3],
       [3, 3, 3, 3]])
```

In [331]:
```
ab=np.full((3,3), 8, dtype='int32')
```

In [329]:
```
np.full((3,3), 'mahi')
```

Out[329]:
```
array([['mahi', 'mahi', 'mahi'],
       ['mahi', 'mahi', 'mahi'],
       ['mahi', 'mahi', 'mahi']], dtype='<U4')
```

In [333]:
```
ab
```

Out[333]:
```
array([[8, 8, 8],
       [8, 8, 8],
       [8, 8, 8]])
```

In [335]:
```
ha = ab.tolist()
```

In [337]:
```
ha
```

Out[337]:
```
[[8, 8, 8], [8, 8, 8], [8, 8, 8]]
```

In [339]:
```
ab.nbytes
```

Out[339]:
```
36
```

In [341]:
```
sys.getsizeof(ha)
```

Out[341]:
```
80
```

In [351]:
```
np.full_like(x,5)
```

Out[351]:
```
array([[5., 5.],
       [5., 5.]])
```

In [355]:
```
np.full(x.size, 4)
```

Out[355]:
```
array([4, 4, 4, 4])
```

Random nums

In [366]:
```
np.random.rand(3,4,2)
```

Out[366]:
```
array([[[0.58870476, 0.1806366 ],
        [0.06935401, 0.66321611],
        [0.5394793 , 0.92929262],
        [0.12807834, 0.68918482]],

       [[0.06814056, 0.40599471],
        [0.92661179, 0.72360297],
        [0.93759588, 0.17274513],
        [0.13890322, 0.78715703]],

       [[0.65721033, 0.76393359],
        [0.18842272, 0.45799107],
        [0.68568686, 0.90640691],
        [0.581805  , 0.90994045]]])
```

In [368]:
```
np.random.rand(3,3)
```

Out[368]:
```
array([[0.2573538 , 0.68236617, 0.39337228],
       [0.19115194, 0.25719452, 0.04527217],
       [0.08295908, 0.71919781, 0.38535149]])
```

In [378]:
```
nn=np.random.rand(1,2,3)
```

In [380]:
```
nn
```

Out[380]:
```
array([[[0.48117469, 0.86280699, 0.08100837],
        [0.41922225, 0.48811367, 0.32552666]]])
```

In [382]:
```
nn.shape
```

Out[382]:
```
(1, 2, 3)
```

In [384]:
```
nx = np.random.rand(2,2,3)
```

In [386]:
```
nx
```

Out[386]:
```
array([[[0.98086055, 0.10585815, 0.29230378],
        [0.93379458, 0.19027932, 0.19003993]],

       [[0.38229419, 0.69087852, 0.71436751],
        [0.77912328, 0.08589799, 0.4390983 ]]])
```

In [390]:
```
np.random.randint(10, size=(3,4))
```

Out[390]:
```
array([[5, 9, 0, 7],
       [1, 6, 9, 0],
       [7, 9, 5, 2]])
```

In [411]:
```
np.random.randint(2,6, size=(4,4))
```

Out[411]:
```
array([[3, 4, 4, 5],
       [3, 5, 5, 3],
       [4, 2, 3, 2],
       [3, 5, 3, 3]])
```

In [413]:
```
np.identity(5)
```

Out[413]:
```
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.]])
```

In [415]:
```
np.identity(5, dtype = 'int32')
```

Out[415]:
```
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]])
```

In [421]:
```
xa = np.array([[2,8]])
np.repeat(xa, 4, axis =0)
```

Out[421]:
```
array([[2, 8],
       [2, 8],
       [2, 8],
       [2, 8]])
```

In [425]:
```
u=np.repeat(xa, 4, axis = 1)
```

In [427]:
```
u
```

Out[427]:
```
array([[2, 2, 2, 2, 8, 8, 8, 8]])
```

In [429]:
```
u.reshape(4,2)
```

Out[429]:
```
array([[2, 2],
       [2, 2],
       [8, 8],
       [8, 8]])
```

In [433]:
```
u.reshape(2,4)
```

Out[433]:
```
array([[2, 2, 2, 2],
       [8, 8, 8, 8]])
```

In [435]:
```
u.reshape(8,1)
```

Out[435]:
```
array([[2],
       [2],
       [2],
       [2],
       [8],
       [8],
       [8],
       [8]])
```

In [437]:
```
u
```

Out[437]:
```
array([[2, 2, 2, 2, 8, 8, 8, 8]])
```

In [439]:
```
u.shape
```

Out[439]:
```
(1, 8)
```

In [441]:
```
x
```

Out[441]:
```
array([[0., 0.],
       [0., 0.]])
```

In [443]:
```
y
```

Out[443]:
```
array([3, 4, 5, 6], dtype=int16)
```

In [449]:
```
x=y.copy()
```

In [451]:
```
x
```

Out[451]:
```
array([[0., 0.],
       [0., 0.]])
```

In [455]:
```
ba = b.copy()
```

In [457]:
```
ba
```

Out[457]:
```
[3, 4, 5, 6]
```

In [459]:
```
abc = x
```

In [461]:
```
abc
```

Out[461]:
```
array([[0., 0.],
       [0., 0.]])
```

Mathematics

In [464]:
```
a = np.array([1,2,3])
print(a)
```

```
[1 2 3]
```

In [466]:
```
a+38
```

Out[466]:
```
array([39, 40, 41])
```

In [468]:
```
a - 23
```

Out[468]:
```
array([-22, -21, -20])
```

In [470]:
```
a /3
```

Out[470]:
```
array([0.33333333, 0.66666667, 1.        ])
```

In [472]:
```
a*3
```

Out[472]:
```
array([3, 6, 9])
```

In [480]:
```
b = np.array([3,4,2])
```

In [482]:
```
b
```

Out[482]:
```
array([3, 4, 2])
```

In [484]:
```
b+a
```

Out[484]:
```
array([4, 6, 5])
```

In [486]:
```
np.sin(a)
```

Out[486]:
```
array([0.84147098, 0.90929743, 0.14112001])
```

In [488]:
```
np.cos(a)
```

Out[488]:
```
array([ 0.54030231, -0.41614684, -0.9899925 ])
```

In [506]:
```
av = np.ones((2,3))
print(av)
aj = np.full((3,2), 6)
print(aj)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
[[6 6]
 [6 6]
 [6 6]]
```

In [510]:
```
np.matmul(av, aj)
```

Out[510]:
```
array([[18., 18.],
       [18., 18.]])
```

In [517]:
```
np.linspace(1,3,10)
```

Out[517]:
```
array([1.        , 1.22222222, 1.44444444, 1.66666667, 1.88888889,
       2.11111111, 2.33333333, 2.55555556, 2.77777778, 3.        ])
```

In [519]:
```
## math calculations
# Determinant, trace, singular vector dimentions
# eigen values
# matrix norm
#Inverse
#Transpose, etc.
```

Statistics

In [522]:
```
#min, max, mean, mod, etc.
```

In [524]:
```
s = np.array([[2,3,4], [3,4,5]])
s
```

Out[524]:
```
array([[2, 3, 4],
       [3, 4, 5]])
```

In [526]:
```
s.mean()
```

Out[526]:
```
3.5
```

In [528]:
```
s.min()
```

Out[528]:
```
2
```

In [530]:
```
s.max()
```

Out[530]:
```
5
```

In [532]:
```
s.std()
```

Out[532]:
```
0.9574271077563381
```

In [536]:
```
s
```

Out[536]:
```
array([[2, 3, 4],
       [3, 4, 5]])
```

In [538]:
```
s.sum()
```

Out[538]: 21

In [540]:
```
np.min(s)
```

Out[540]: 2

In [542]:
```
np.max(s)
```

Out[542]: 5

In [544]:
```
np.mean(s)
```

Out[544]: 3.5

In [546]:
```
np.std(s)
```

Out[546]: 0.9574271077563381

In [548]:
```
np.min(s, axis = 0)
```

Out[548]: array([2, 3, 4])

In [550]:
```
np.min(s, axis = 1)
```

Out[550]: array([2, 3])

In [552]:
```
np.sum(s)
```

Out[552]: 21

In [554]:
```
np.sum(s, axis = 0)
```

Out[554]: array([5, 7, 9])

In [556]:
```
np.sum(s, axis = 1)
```

Out[556]: array([9, 12])

In [558]:
```
g = np.array([2,3,4])
h = np.array([38, 34, 87])
np.vstack([g,h])
```

Out[558]:
```
array([[ 2,  3,  4],
       [38, 34, 87]])
```

In [560]:
```
np.hstack([g, h])
```

Out[560]: array([2, 3, 4, 38, 34, 87])

In [562]:
```
np.vstack([g,h, g, h])
```

Out[562]:
```
array([[ 2,  3,  4],
       [38, 34, 87],
       [ 2,  3,  4],
       [38, 34, 87]])
```

In [566]:
```
np.hstack([g,h,g,h])
```

Out[566]:
```
array([ 2,  3,  4, 38, 34, 87,  2,  3,  4, 38, 34, 87])
```

In [570]:
```
o =np.array([2,3])
np.vstack([g,o])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[570], line 2
      1 o =np.array([2,3])
----> 2 np.vstack([g,o])

File ~\anaconda3\Lib\site-packages\numpy\core\shape_base.py:289, in vstack(t
up, dtype, casting)
    287 if not isinstance(arrs, list):
    288     arrs = [arrs]
--> 289 return _nx.concatenate(arrs, 0, dtype=dtype, casting=casting)

ValueError: all the input array dimensions except for the concatenation axis
must match exactly, but along dimension 1, the array at index 0 has size 3 a
nd the array at index 1 has size 2
```

In [574]:
```
hu = np.zeros((2,2))
hi = np.ones((2,2))
```

In [576]:
```
aaa = np.vstack([hu, hi])
```

In [578]:
```
aaa
```

Out[578]:
```
array([[0., 0.],
       [0., 0.],
       [1., 1.],
       [1., 1.]])
```

In [594]:
```
data = np.genfromtxt('sample.txt', delimiter=',')
```

In [600]:
```
data
```

Out[600]:
```
array([[1.000e+02, 2.000e+00, 3.000e+00, 4.980e+02, 8.200e+01, 2.000e+02,
        2.800e+01, 9.800e+01],
       [2.830e+02, 3.848e+03, 2.930e+02, 3.000e+00, 3.830e+02, 4.030e+02,
        3.800e+01, 8.300e+01],
       [3.000e+01, 8.300e+01, 4.000e+00, 3.830e+02, 3.000e+00, 3.000e+00,
        3.000e+00, 8.700e+01]])
```

In [602]:
```
data.astype('int32')
```

Out[602]:
```
array([[ 100,    2,    3,  498,   82,  200,   28,   98],
       [ 283, 3848,  293,    3,  383,  403,   38,   83],
       [  30,   83,    4,  383,    3,    3,    3,   87]])
```

In [608]:
```
data = data.astype('int32') #inplace = True
```

In [610]:
```
data
```

Out[610]:
```
array([[ 100,    2,    3,  498,   82,  200,   28,   98],
       [ 283, 3848,  293,    3,  383,  403,   38,   83],
       [  30,   83,    4,  383,    3,    3,    3,   87]])
```

In [612]:
```
data > 200
```

Out[612]:
```
array([[False, False, False,  True, False, False, False, False],
       [ True,  True,  True, False,  True,  True, False, False],
       [False, False, False,  True, False, False, False, False]])
```

In [614]:
```
data [ data > 200]
```

Out[614]:
```
array([ 498,  283, 3848,  293,  383,  403,  383])
```

In [616]:
```
data < 100
```

Out[616]:
```
array([[False,  True,  True, False,  True, False,  True,  True],
       [False, False, False,  True, False, False,  True,  True],
       [ True,  True,  True, False,  True,  True,  True,  True]])
```

In [620]:
```
ii=data [ data < 100]
```

In [622]:
```
ii
```

Out[622]:
```
array([ 2,  3, 82, 28, 98,  3, 38, 83, 30, 83,  4,  3,  3,  3, 87])
```

In [626]:
```
ii[[2,4,9]]
```

Out[626]:
```
array([82, 98, 83])
```

In [628]:
```
data
```

Out[628]:
```
array([[ 100,    2,    3,  498,   82,  200,   28,   98],
       [ 283, 3848,  293,    3,  383,  403,   38,   83],
       [  30,   83,    4,  383,    3,    3,    3,   87]])
```

In [646]:
```
yy = np.any(data > 100)
```

In [648]:
```
yy
```

Out[648]:
```
True
```

In [650]:
```
data[yy]
```

Out[650]:
```
array([[[ 100,    2,    3,  498,   82,  200,   28,   98],
        [ 283, 3848,  293,    3,  383,  403,   38,   83],
        [  30,   83,    4,  383,    3,    3,    3,   87]]])
```

In [652]:
```
np.any(data > 100, axis = 0)
```

Out[652]:
```
array([ True,  True,  True,  True,  True,  True, False, False])
```

In [662]:
```
iio = np.all(data > 100, axis = 0)
```

In [666]:
```
iio
```

Out[666]:
```
array([False, False, False, False, False, False, False, False])
```

In [670]:
```
data
```

Out[670]:
```
array([[ 100,    2,    3,  498,   82,  200,   28,   98],
       [ 283, 3848,  293,    3,  383,  403,   38,   83],
       [  30,   83,    4,  383,    3,    3,    3,   87]])
```

In [678]:
```
(data>100) & (data < 100)
```

Out[678]:
```
array([[False, False, False, False, False, False, False, False],
       [False, False, False, False, False, False, False, False],
       [False, False, False, False, False, False, False, False]])
```

In [684]:
```
data [(data>100) & (data < 100)]
```

Out[684]:
```
array([], dtype=int32)
```

In [686]:
```
data [(data>100) | (data < 100)]
```

Out[686]:
```
array([   2,    3,  498,   82,  200,   28,   98,  283, 3848,  293,    3,
        383,  403,   38,   83,   30,   83,    4,  383,    3,    3,    3,
         87])
```

In [688]:
```
data [~(data>100) & (data < 100)]
```

Out[688]:
```
array([ 2,  3, 82, 28, 98,  3, 38, 83, 30, 83,  4,  3,  3,  3, 87])
```

In [690]:
```
data [~(data>100) & ~(data < 100)]
```

Out[690]:
```
array([100])
```

In [692]:
```
data
```

Out[692]:
```
array([[ 100,    2,    3,  498,   82,  200,   28,   98],
       [ 283, 3848,  293,    3,  383,  403,   38,   83],
       [  30,   83,    4,  383,    3,    3,    3,   87]])
```

In [ ]:
```

```