Numpy:

Numpy stands for numerical python it is used to perform different mathematical operations on array.
Array is denoted with '[]'.
It is unordered(index positition is fixed), mutable(item can be modified) and homogeneouse(items of similar datatypes) collections of items.

In [3]:
```
import numpy as np
```

In [4]:
```
np.__version__
```

Out[4]: '1.24.3'

In [5]:
```
#Creating 1-d array
a = [3,4,5,6,7,8]
print(a)
type(a)
```

```
[3, 4, 5, 6, 7, 8]
```

Out[5]: list

In [7]:
```
arr1 = np.array(a)
print(arr1)
type(arr1) #numpy.ndarray
```

```
[3 4 5 6 7 8]
```

Out[7]: numpy.ndarray

In [8]:
```
#numpy.ndarray :-Numerical python.ndiemensional array
```

In [9]:
```
#creating array from tuple
t = (5,7,8,3,2)
print(t)
type(t)
```

```
(5, 7, 8, 3, 2)
```

Out[9]: tuple

In [10]:
```
arr2 = np.array(t)
print(arr2)
type(arr2)
```

```
[5 7 8 3 2]
```

Out[10]:
```
numpy.ndarray
```

In [11]:
```
#creating array from dictionary
d = {1:'A',2:'B',3:"C"}
print(d)
type(d)
```

```
{1: 'A', 2: 'B', 3: 'C'}
```

Out[11]:
```
dict
```

In [12]:
```
arr3 = np.array(d)
print(arr3)
type(arr3)
```

```
{1: 'A', 2: 'B', 3: 'C'}
```

Out[12]:
```
numpy.ndarray
```

In [13]:
```
#creating 2d array
a = np.array([[2,3,4],[4,6,9]])
print(a)
```

```
[[2 3 4]
 [4 6 9]]
```

In [14]:
```
b = np.array([[4,5,6,7,8],[5,2,6,8,3],[4,5,7,8,3],[1,5,8,3,5],[5,3,7,9,2]])
print(b)
```

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

In [16]:
```
#we can't take different length
c = np.array([[3,4,5],[6,7]])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[16], line 2
      1 #we can't take different length
----> 2 c = np.array([[3,4,5],[6,7]])

ValueError: setting an array element with a sequence. The requested array has an inhomogeneous shape after 1 dimensions. The detected shape was (2,) + in
homogeneous part.
```

In [17]:
```
t = np.array(((2,3,4),(6,7,8),(2,9,5)))
print(t)
```

```
[[2 3 4]
 [6 7 8]
 [2 9 5]]
```

In [18]:
```
#Array attributes
#shape: Shape of an array
t.shape
```

Out[18]: (3, 3)

In [19]:
```
#size:elements present in an array
t.size
```

Out[19]: 9

In [20]:
```
#ndim:-diemsions of array
t.ndim
```

Out[20]: 2

In [23]:
```
#arange:
np.arange(10)
```

Out[23]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [25]:
```
np.arange(2,101,2)
```

Out[25]:
```
array([  2,   4,   6,   8,  10,  12,  14,  16,  18,  20,  22,  24,  26,
        28,  30,  32,  34,  36,  38,  40,  42,  44,  46,  48,  50,  52,
        54,  56,  58,  60,  62,  64,  66,  68,  70,  72,  74,  76,  78,
        80,  82,  84,  86,  88,  90,  92,  94,  96,  98, 100])
```

In [27]:
```
#zeros
np.zeros((2,2))
```

Out[27]:
```
array([[0., 0.],
       [0., 0.]])
```

In [28]:
```
#ones
np.ones((5,7))
```

Out[28]:
```
array([[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., 1.],
       [1., 1., 1., 1., 1., 1., 1.]])
```

In [29]:
```
#full
np.full((5,6),355)
```

Out[29]:
```
array([[355, 355, 355, 355, 355, 355],
       [355, 355, 355, 355, 355, 355],
       [355, 355, 355, 355, 355, 355],
       [355, 355, 355, 355, 355, 355],
       [355, 355, 355, 355, 355, 355]])
```

In [31]:
```
#eye
np.eye((6))
```

Out[31]:
```
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 [32]:
```
#linspace
np.linspace(1,10,5)
```

Out[32]:
```
array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

Random

In [37]:
```
#Random
np.random.rand()
```

Out[37]:
```
0.5495994458796208
```

In [38]:
```
np.random.rand(6)
```

Out[38]:
```
array([0.81608969, 0.60971333, 0.5229311 , 0.36624903, 0.29266193,
       0.13654031])
```

In [39]:
```
np.random.rand(6,7)
```

Out[39]:
```
array([[0.19504573, 0.64384212, 0.22549298, 0.31382776, 0.94589953,
        0.23433148, 0.88749216],
       [0.45496813, 0.21580202, 0.91485938, 0.24901948, 0.66566021,
        0.44906192, 0.93520994],
       [0.93835495, 0.68499226, 0.51930411, 0.44313705, 0.4210442 ,
        0.36879323, 0.97622877],
       [0.76640358, 0.97777608, 0.43097626, 0.21582521, 0.35729831,
        0.69824693, 0.92604431],
       [0.16928925, 0.29370603, 0.72996637, 0.59329486, 0.5649826 ,
        0.25767902, 0.60685733],
       [0.72948823, 0.00286309, 0.75300421, 0.72680406, 0.4717863 ,
        0.86029758, 0.31624309]])
```

In [47]:
```
np.random.randint(5)
```

Out[47]:
```
0
```

In [50]:
```
np.random.randint(1,9,(5,5))
```

Out[50]:
```
array([[6, 4, 1, 3, 5],
       [4, 2, 1, 4, 6],
       [2, 2, 4, 3, 8],
       [2, 6, 3, 6, 4],
       [8, 7, 5, 1, 4]])
```

Mathematical operations

In [51]:
```
arr1 = np.array([3,4,5,6,7])
arr2 = np.array([4,5,3,9,8])
print(arr1)
print(arr2)
```

```
[3 4 5 6 7]
[4 5 3 9 8]
```

In [52]:
```
arr1+arr2
```

Out[52]:
```
array([ 7,  9,  8, 15, 15])
```

In [54]:
```
#addition
np.add(arr1,arr2)
```

Out[54]:
```
array([ 7,  9,  8, 15, 15])
```

In [58]:
```
np.add(arr1,5)
```

Out[58]:
```
array([ 8,  9, 10, 11, 12])
```

In [55]:
```
#subtraction
np.subtract(arr1,arr2)
```

Out[55]:
```
array([-1, -1,  2, -3, -1])
```

In [56]:
```
#multiplication
np.multiply(arr1,arr2)
```

Out[56]:
```
array([12, 20, 15, 54, 56])
```

In [57]:
```
#division
np.divide(arr1,arr2)
```

Out[57]:
```
array([0.75      , 0.8       , 1.66666667, 0.66666667, 0.875     ])
```

In [59]:
```
#modulus
np.mod(arr1,arr2)
```

Out[59]:
```
array([3, 4, 2, 6, 7])
```

In [60]:
```
#log
np.log(arr1)
```

Out[60]:
```
array([1.09861229, 1.38629436, 1.60943791, 1.79175947, 1.94591015])
```

In [62]:
```
#sqrt()
np.sqrt(255)
```

Out[62]:
```
15.968719422671311
```

In [64]:
```
#sin
np.sin(arr1)
```

Out[64]:
```
array([ 0.14112001, -0.7568025 , -0.95892427, -0.2794155 ,  0.6569866 ])
```

In [65]:
```
#cos()
np.cos(arr2)
```

Out[65]:
```
array([-0.65364362,  0.28366219, -0.9899925 , -0.91113026, -0.14550003])
```

In [66]:
```
#power()
np.power(arr2,5)
```

Out[66]:
```
array([ 1024,  3125,   243, 59049, 32768], dtype=int32)
```

Statistical function

In [67]:
```
#min()
np.min([3,4,5,6,7,8])
```

Out[67]:
```
3
```

In [68]:
```
print(arr1)
print(arr2)
```

```
[3 4 5 6 7]
[4 5 3 9 8]
```

In [69]:
```
#max()
np.max(arr2)
```

Out[69]:
```
9
```

In [70]:
```
#average()
np.average(arr1)
```

Out[70]:
```
5.0
```

In [71]:
```
#mean()
np.mean(arr2)
```

Out[71]:
```
5.8
```

In [72]:
```
#std():standard deviation
np.std(arr2)
```

Out[72]:
```
2.3151673805580453
```

In [73]:
```
#meadian
np.median(arr1)
```

Out[73]:
```
5.0
```

In [74]:
```
#var:
np.var(arr1)
```

Out[74]:
```
2.0
```

In [75]:
```
#cumsum()
print(arr1)
np.cumsum(arr1)
```

```
[3 4 5 6 7]
```

Out[75]:
```
array([ 3,  7, 12, 18, 25])
```

In [80]:
```
#repeate
n = np.array([4,5,6,7,8,9])
np.repeat(n,3)
```

Out[80]:
```
array([4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7, 8, 8, 8, 9, 9, 9])
```

In [81]:
```
#tile
n = np.array([4,5,6,7,8,9])
np.tile(n,3)
```

Out[81]:
```
array([4, 5, 6, 7, 8, 9, 4, 5, 6, 7, 8, 9, 4, 5, 6, 7, 8, 9])
```

In [82]:
```
#where()
n = np.array([4,5,6,7,8,9,5])
np.where(n==5)
```

Out[82]:
```
(array([1, 6], dtype=int64),)
```

Vector Math

In [83]:
```
a = np.array([[2,3],[6,7]])
b = np.array([[4,5],[8,9]])
print(a)
print(b)
```

```
[[2 3]
 [6 7]]
[[4 5]
 [8 9]]
```

In [84]:
```
#dot product
np.dot(a,b)
```

Out[84]:
```
array([[32, 37],
       [80, 93]])
```

In [85]:
```
#cross()
np.cross(a,b)
```

Out[85]:
```
array([-2, -2])
```

In [86]:
```
#transpose()
print(a)
```

```
[[2 3]
 [6 7]]
```

In [87]:
```
np.transpose(a)
```

Out[87]:
```
array([[2, 6],
       [3, 7]])
```

In [88]:
```
print(n)
```

```
[4 5 6 7 8 9 5]
```

In [89]:
```
#indexing
n[2]
```

Out[89]:
```
6
```

In [91]:
```
#slicing
n[1:6:2]
```

Out[91]:
```
array([5, 7, 9])
```

In [92]:
```
a = np.array([[4,5,6],[2,8,9],[8,9,3]])
print(a)
```

```
[[4 5 6]
 [2 8 9]
 [8 9 3]]
```

In [94]:
```
b = np.array([[2,3],[5,6],[5,9]])
print(b)
```

```
[[2 3]
 [5 6]
 [5 9]]
```

In [95]:
```
c = np.array([[2,3,4],[9,8,7]])
print(c)
```

```
[[2 3 4]
 [9 8 7]]
```

In [97]:
```
print(a)
```

```
[[4 5 6]
 [2 8 9]
 [8 9 3]]
```

In [98]:
```
#slicing
#var[row,column]
# row[start:stop+1:step]
# column[start:stop+1:step]
a[0:2:1,1:3:1]
```

Out[98]:
```
array([[5, 6],
       [8, 9]])
```

In [100]:
```
a[0:3:1,0:3:2]
```

Out[100]:
```
array([[4, 6],
       [2, 9],
       [8, 3]])
```

In [99]:
```
a[0:3:2,0:3:1]
```

Out[99]:
```
array([[4, 5, 6],
       [8, 9, 3]])
```

In [101]:
```
print(a)
print(b)
print(c)
```

```
[[4 5 6]
 [2 8 9]
 [8 9 3]]
[[2 3]
 [5 6]
 [5 9]]
[[2 3 4]
 [9 8 7]]
```

In [102]:
```
#concatenate
np.concatenate((a,b),axis=1)
```

Out[102]:
```
array([[4, 5, 6, 2, 3],
       [2, 8, 9, 5, 6],
       [8, 9, 3, 5, 9]])
```

In [103]:
```
np.concatenate((a,c),axis=0)
```

Out[103]:
```
array([[4, 5, 6],
       [2, 8, 9],
       [8, 9, 3],
       [2, 3, 4],
       [9, 8, 7]])
```

In [104]:
```
np.concatenate((a,c),axis=1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[104], line 1
----> 1 np.concatenate((a,c),axis=1)

File <__array_function__ internals>:200, in concatenate(*args, **kwargs)

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

In [106]:
```
#vstack
# np.concatenate((a,c),axis=0)
np.vstack((a,c))
```

Out[106]:
```
array([[4, 5, 6],
       [2, 8, 9],
       [8, 9, 3],
       [2, 3, 4],
       [9, 8, 7]])
```

In [107]:
```
#hstack
# np.concatenate((a,b),axis=1)
np.hstack((a,b))
```

Out[107]:
```
array([[4, 5, 6, 2, 3],
       [2, 8, 9, 5, 6],
       [8, 9, 3, 5, 9]])
```

In [ ]: