In [1]:
```
import numpy as np
```

In [2]:
```
arr=np.array([2,55,34,8,11,17,60,94,100])
```

ARRAY MEMORY

In [3]:
```
print(arr. flags["WRITEABLE"] )  # means array owns its memory
print(arr.flags["C_CONTIGUOUS"])
print(arr.flags["OWNDATA"])
```

```
True
True
True
```

In [4]:
```
arr.shape
```

Out[4]: (9,)

In [5]:
```
arr.strides             #Tuple of bytes to step in each dimension when traversing an array.
```

Out[5]: (8,)

In [6]:
```
print(arr.ndim)         #Number of array dimensions.
arr.size
arr.itemsize
arr.nbytes              #Total bytes consumed by the elements of the array.

arr.base                #Base object if memory is from some other object.
y=arr[1:4]
y.base is arr
```

```
1
```

Out[6]: True

In [7]:
```
arr.dtype         #Data-type of the array's elements
```

Out[7]: dtype('int64')

In [8]:
```
complex_arr=np.array([ 381.          +0.j        ,   82.65246204+135.09934947j,
       -103.91594351 +15.94751083j,  -85.5         -7.79422863j,
        -74.73651853 +15.94812435j,  -74.73651853 -15.94812435j,
        -85.5         +7.79422863j, -103.91594351 -15.94751083j,
         82.65246204-135.09934947j])
real_part=complex_arr.real
real_part
imaginary_part=complex_arr.imag
imaginary_part
```

Out[8]:
```
array([   0.        ,  135.09934947,   15.94751083,   -7.79422863,
         15.94812435,  -15.94812435,    7.79422863,  -15.94751083,
       -135.09934947])
```

In [9]:
```
new_arr=arr.reshape(3,3)
new_arr
print(new_arr.T)   #TRANSPOSE OF AN ARRAY.
```

```
[[  2   8  60]
 [ 55  11  94]
 [ 34  17 100]]
```

------ARRAY METHODS------

In [10]:
```
arr.item(2)   #0 dan başlayarak 2. elemanı seçer
arr.dtype
```

Out[10]:
```
dtype('int64')
```

In [11]:
```
my_arr=arr.astype(str,casting='safe',copy=False)   #veri tipini degistirdik normalde int di
my_arr.dtype
```

Out[11]:
```
dtype('<U21')
```

In [12]:
```
arr.getfield(float) #Veriyi ham bitleriyle tutar ama başka tipmiş gibi okur
```

Out[12]:
```
array([9.88e-324, 2.72e-322, 1.68e-322, 3.95e-323, 5.43e-323, 8.40e-323,
       2.96e-322, 4.64e-322, 4.94e-322])
```

In [13]:
```
arrayim=np.random.random((3,2))
arrayim
```

Out[13]:
```
array([[0.58024734, 0.65913581],
       [0.82403493, 0.81159347],
       [0.52678231, 0.7154612 ]])
```

In [14]:
```
zeros=np.zeros((4,2))
zeros
```

Out[14]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [15]:
```
empty_arr=np.empty(4)
empty_arr.fill(6)         #verilen sayı ile arrayi doldurur
empty_arr
```

Out[15]:
```
array([6., 6., 6., 6.])
```

-------SHAPE MANUPILATION ------

In [16]:
```
arr=arr.reshape(3,3)
```

In [17]:
```
empty_arr.resize(3,2,refcheck=False)      #yeniden boyutlandırır ve ekleme yapar.
empty_arr
```

Out[17]:
```
array([[6., 6.],
       [6., 6.],
       [0., 0.]])
```

In [18]:
```
arr.swapaxes(0,1)    #axis 0 (satırlar) ↔ axis 1 (sütunlar)
```

Out[18]:
```
array([[  2,   8,  60],
       [ 55,  11,  94],
       [ 34,  17, 100]])
```

In [19]:
```
two_dim=np.random.randint(10,80,20).reshape(4,5)   #two dimension to one dimension
two_dim
flat=two_dim.flatten()
flat
```

Out[19]:
```
array([73, 78, 48, 71, 79, 77, 11, 30, 74, 56, 23, 49, 10, 63, 45, 46, 66,
       11, 41, 67], dtype=int32)
```

In [20]:
```
dummy_array = np.array([[[42], [51], [63]]])  # shape: (1, 3, 1)
print("Orijinal şekil:", dummy_array.shape)

squeezed = np.squeeze(dummy_array)               #3 boyutlu arrayi tek boyutlu hale getirdi
print("Sıkıştırılmış şekil:", squeezed.shape)
print(squeezed)
```

```
Orijinal şekil: (1, 3, 1)
Sıkıştırılmış şekil: (3,)
[42 51 63]
```

--------SELECTION AND MANUPILATION----------:

In [21]:
```
two_dim
```

Out[21]:
```
array([[73, 78, 48, 71, 79],
       [77, 11, 30, 74, 56],
       [23, 49, 10, 63, 45],
       [46, 66, 11, 41, 67]], dtype=int32)
```

In [22]:
```
two_dim[1:3]                          #1 satırdan 3 . satıra kadar verir 3 dahil değil
```

Out[22]:
```
array([[77, 11, 30, 74, 56],
       [23, 49, 10, 63, 45]], dtype=int32)
```

In [23]:
```
two_dim[1:3,0:2]                    #1 ve 2. satırın 0 ve 1. elemanları    [satır:satır,sütun:sütun]
```

Out[23]:
```
array([[77, 11],
       [23, 49]], dtype=int32)
```

In [24]:
```
two_dim.take([1,2,3])                   #verilen indislerdekini seçer
```

Out[24]:
```
array([78, 48, 71], dtype=int32)
```

In [25]:
```
two_dim.put(1,11)                      #verilen indise verilen numarayı yazar
two_dim.take([1,2,3])
```

Out[25]:
```
array([11, 48, 71], dtype=int32)
```

In [26]:
```
two_dim.repeat(3)                      #her elemanı verilen sayı kadar tekrar  yazar
```

Out[26]:
```
array([73, 73, 73, 11, 11, 11, 48, 48, 48, 71, 71, 71, 79, 79, 79, 77, 77,
       77, 11, 11, 11, 30, 30, 30, 74, 74, 74, 56, 56, 56, 23, 23, 23, 49,
       49, 49, 10, 10, 10, 63, 63, 63, 45, 45, 45, 46, 46, 46, 66, 66, 66,
       11, 11, 11, 41, 41, 41, 67, 67, 67], dtype=int32)
```

In [27]:
```
two_dim.sort()
two_dim
```

Out[27]:
```
array([[11, 48, 71, 73, 79],
       [11, 30, 56, 74, 77],
       [10, 23, 45, 49, 63],
       [11, 41, 46, 66, 67]], dtype=int32)
```

In [28]:
```
two_dim.argsort()              #elemanların kaçıncı büyük olduğunu syöler biz zaten sıralamıştık
```

Out[28]:
```
array([[0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4]])
```

In [29]:
```
partitioned=two_dim.ravel()

partitioned.partition(8)    #the value of the element in k-th position is in the position it would be in a sorted array. In the output array,
partitioned                  #verilen indeksteki elemanı ortaya küçüklerini sola büyüklerini sağa yazar bu örenkte 8. indekste 33 vardı
```

Out[29]:
```
array([10, 11, 11, 11, 23, 30, 41, 45, 46, 48, 49, 56, 63, 66, 67, 71, 73,
       74, 77, 79], dtype=int32)
```

In [30]:
```
two_dim.nonzero()            #0 olmayan her satırdaki eleman için  o satırın indeks numarasını yazar. ve 2. arrayda her saatır için 0 dan n e kadar yazar.
```

Out[30]:
```
(array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3]),
 array([0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4]))
```

In [31]:
```
arr = np.array([10, 20, 30, 40, 50])                #condition: Boolean (True/False) değerler içeren bir 1D dizi.
                                                      #axis: Filtrenin uygulanacağı eksen (0: satır, 1: sütun).
mask = [True, False, True, False, True]
filtered = np.compress(mask, arr)                     #array: Filtrelenecek NumPy dizisi.
print(filtered)  # [10 30 50]
```

```
[10 30 50]
```

```
----------CALCULATION-----------
```

In [32]:
```
arr
```

Out[32]:
```
array([10, 20, 30, 40, 50])
```

In [33]:
```
arr.max()
```

Out[33]:
```
np.int64(50)
```

In [34]:
```
arr.min()
```

Out[34]:
```
np.int64(10)
```

In [35]:
```
arr.argmin()
```

Out[35]:
```
np.int64(0)
```

In [36]:
```
arr.clip(20,50)          #alt ve üst değer değişikliği yapar 20 den küçükleri 20'ye büyükleri de 50 ye eşitler
```

Out[36]:
```
array([20, 20, 30, 40, 50])
```

In [37]:
```
comp= np.array([1+2j, 3-4j, -2+0.5j, 0-1j])
comp.conj()                                           #complex conjugate ini verir
```

Out[37]:
```
array([ 1.-2.j ,  3.+4.j , -2.-0.5j,  0.+1.j ])
```

In [38]:
```
fractions=np.array([1.234, 5.67891, 0.0001234, -2.3456])
fractions.round(2)                                    #küsürat kısmını verilen basmağa indirir
```

Out[38]:
```
array([ 1.23,  5.68,  0.  , -2.35])
```

In [39]:
```
arr=np.random.randint(10,90,16).reshape(4,4)
arr
```

Out[39]:
```
array([[34, 45, 75, 36],
       [82, 32, 38, 61],
       [59, 37, 48, 57],
       [48, 48, 74, 76]], dtype=int32)
```

In [40]:
```
arr=np.random.randint(10,90,16).reshape(4,4)
arr.sum(axis=0)    #satırları toplar verir
arr.sum(axis=1)    #sütunları toplar verir
```

Out[40]:
```
array([212, 207, 130, 128])
```

In [41]:
```
prices = np.array([100, 102, 105, 107, 110])
```

In [42]:
```
np.diff(prices)
```

Out[42]:
```
array([2, 3, 2, 3])
```

.sin .cos. arctan .arcos .hpot .degrees. radians .round. .florr .ceil .sum .cumsum .log .exp .log10 .power .pow .mod .sqrt .absolute GİBİ PEK ÇOK MATH FONKSİYONU MEVCUT

In [43]:
```
arr.cumsum(axis=0)    #kümülatif toplamı verir
```

Out[43]:
```
array([[ 54,  45,  35,  78],
       [131,  67, 102, 119],
       [172, 107, 112, 158],
       [197, 155, 145, 180]])
```

In [44]:
```
darray=np.array([10,22,5,32,84,55])
darray.var()      #arrayin varyansını verir
```

Out[44]:
```
np.float64(750.5555555555555)
```

In [45]:
```
darray.std()     #standart sapmasını verir
```

Out[45]:
```
np.float64(27.39626900794259)
```

In [46]:
```
darray.prod(axis=0)   #verilen eksende elemanların çarpımı
```

Out[46]:
```
np.int64(162624000)
```

In [47]:
```
darray.all()    #tüm değerler tru ise true döndürür
darray.any()     #1 tane bile true vars true verir
```

Out[47]:
```
np.True_
```

FOURIER TRANSFORM AND INVERSE FOURIER

In [48]:
```
fft_arr=np.fft.fft(arr)   #Compute the one-dimensional discrete Fourier Transform.
fft_arr
```

Out[48]:
```
array([[212. +0.j,  19.+33.j, -34. +0.j,  19.-33.j],
       [207. +0.j,  10.+19.j,  81. +0.j,  10.-19.j],
       [130. +0.j,  31. -1.j, -28. +0.j,  31. +1.j],
       [128. +0.j,  -8.-26.j, -12. +0.j,  -8.+26.j]])
```

In [49]:
```
normal_arr=np.fft.ifft(fft_arr).real
normal_arr
```

Out[49]:
```
array([[54., 45., 35., 78.],
       [77., 22., 67., 41.],
       [41., 40., 10., 39.],
       [25., 48., 33., 22.]])
```

LINEAR ALGEBRA

In [50]:
```
first_arr=np.array([58,3,7,99])
second_arr=np.array([22,1,43,11])
result=np.dot(first_arr,second_arr)
print(result)
```

```
2669
```

In [51]:
```
result=np.vdot(first_arr,second_arr)
result
```

Out[51]:
```
np.int64(2669)
```

In [52]:
```
first_arr=first_arr.reshape(2,2)
second_arr=second_arr.reshape(2,2)
result1=np.linalg.eig(first_arr)         #eigen value of matrix
result1
```

Out[52]:
```
EigResult(eigenvalues=array([57.49404846, 99.50595154]), eigenvectors=array([[-0.98607486, -0.07209073],
       [ 0.16630203, -0.99739808]]))
```

In [53]:
```
det=np.linalg.det(second_arr)   #determinant of array
det
```

Out[53]:
```
np.float64(198.99999999999997)
```

In [54]:
```
inverse=np.linalg.inv(first_arr)
inverse
```

Out[54]:
```
array([[ 0.01730467, -0.00052438],
       [-0.00122356,  0.01013809]])
```

In [55]:
```
t=np.linalg.matrix_transpose(first_arr)
t
```

Out[55]:
```
array([[58,  7],
       [ 3, 99]])
```

In [56]:
```
a = np.array([[1, 2], [3, 5]])   # 1*x0 + 2 * x1 = 1    and   3 * x0 + 5 * x1 = 2
b = np.array([1, 2])             # a arrayi katsayılar arrayi b arrayi sonuç arrayi solve methodu ile çözüyor
x = np.linalg.solve(a, b)
```