numpy

```
data_list = [1,2,3]
```

```
data_list
```

```
[1, 2, 3]
```

```
import numpy as np
```

```
arr = np.array(data_list)
```

```
arr
```

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

```
data_list2 = [[10,20,30],[40,50,60],[70,80,90]]
```

```
arr2 = np.array(data_list2)
```

```
arr2
```

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

```
arr3 = np.array([1,2,3,4,5])
```

```
arr3
```

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

```
arr3[0]
```

```
1
```

```
arr3[4]
```

```
5
```

```
arr2
```

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

```
arr2[2,2] # 2. satır 2. sütun
```

```
90
```

```
len(arr2)
```

```
3
```

```
type(arr2)
```

```
numpy.ndarray
```

```
np.arange(10,20) # 10'dan 20'ye kadar 20 dahil değil
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

```
np.arange(0,100,5)
```

```
array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,
80,
       85, 90, 95])
```

```
np.zeros(10)
```

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

```
np.ones(10)
```

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

```
np.zeros((2,2))
```

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

```
np.zeros((5,8))
```

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

```
np.ones((3,3))
```

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

aşağıdaki örnekte, 0 ile 100 arasını 4 eşit parçaya bölerek 5 tane değer tutmayı sağlıyor.

```
np.linspace(0,100,5)
```

```
array([  0.,  25.,  50.,  75., 100.])
```

```
np.linspace(0,1,6)
```

```
array([0. , 0.2, 0.4, 0.6, 0.8, 1. ])
```

birim matris

```
np.eye(6)
```

```
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.]])
```

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

```
6
```

```
np.random.randint(15) # (0,15) olarak algılar
```

```
9
```

```
np.random.randint(1, 20, 5) #1 ile 20 arasında 5 değer üretip np.array yaptı.
```

```
array([16, 14,  9, 18,  5])
```

```
.randeom.rand metodu. tek parametre alır ve bu parametre değerine göre 0 ile 1 arası sayı üretir.
```

```
np.random.rand(5) #0 ile 1 arasında 5 tane sayı üretti.
```

```
array([0.16001316, 0.96133198, 0.84773853, 0.04469474, 0.14778514])
```

```
np.random.randn(5)
```

```
array([-0.5375281 ,  0.87092993,  1.09334292,  0.59251109,
1.73848473])
```

```
.reshape metodu
```

```
arr = np.arange(25)
```

```
arr # dizi eleman sayısı ile matris botyutu tam uyuşmalı, eksik ya da fazla olamaz.
```

```
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, 24])
```

```
arr.reshape(5,5)
```

```
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, 24]])
```

```
arr = np.random.randint(1,100,10)
```

```
arr
```

```
array([78, 22, 12, 73, 84, 73, 26, 18, 54, 51])
```

```
arr.max()
```

```
84
```

```
arr.min()
```

```
12
```

```
arr.sum() #arr elemanlarının toplamı
```

```
491
```

```
arr.mean() # ortalaması
```

```
49.1
```

```
arr.argmax()
```

```
4
```

```
arr.argmin()
```

```
2
```

Determinant hesaplama

```
arr = np.random.randint(1, 100, 25)
```

```
arr
```

```
array([ 6, 75, 26, 91, 42, 26, 37, 97, 81, 89, 60,  9, 80, 34, 39, 47,
26,
       11, 86, 21, 57, 32, 76, 77, 93])
```

```
arrshape = arr.reshape(5,5)
arrshape
```

```
array([[ 6, 75, 26, 91, 42],
       [26, 37, 97, 81, 89],
       [60,  9, 80, 34, 39],
       [47, 26, 11, 86, 21],
       [57, 32, 76, 77, 93]])
```

```
np.linalg.det(arrshape)
```

```
599560716.0000005
```

```
arr2 = np.array([[10,20],[1,2]])
arr2
```

```
array([[10, 20],
       [ 1,  2]])
```

```
np.linalg.det(arr2)
```

```
-1.1102230246251546e-15
```

```
round(np.linalg.det(arr2))
```

```
0
```

array lerin indekslenmesi ve parçalanması

```
arr = np.arange(1,10)
arr
```

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

```
arr[2]
```

```
3
```

```
arr[1:4]
```

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

```
arr[:4]
```

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

```
arr[::2]
```

```
array([1, 3, 5, 7, 9])
```

```
arr
```

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

```
arr[:3]
```

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

```
arr[:3] = 36
```

```
arr
```

```
array([36, 36, 36,  4,  5,  6,  7,  8,  9])

arr = np.arange(1,10)
arr

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

arr2 = arr
arr2

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

arr2[:3] = 100
arr2

array([100, 100, 100,   4,   5,   6,   7,   8,   9])

arr #arr2' dek ideğişiklik arr' yi etkiledi. Sebebi bellekten okuma !

array([100, 100, 100,   4,   5,   6,   7,   8,   9])
```

değişiklik olsun istemiyorsak copy metodu kullanılmalı

```
arr= np.arange(1,10)
arr

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

arr2 = arr.copy()
arr2

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

arr2[:3] = 1000
arr2

array([1000, 1000, 1000,    4,    5,    6,    7,    8,    9])

arr

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

Çok boyutlu matrisler, parçalama, indeksleme

```
arr = np.arange(1,21)
arr

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

```
arr.reshape(5,4)
```

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

```
arr
```

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

```
arr = arr.reshape(5,4)
arr
```

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

```
arr[0,0]
```

```
1
```

```
arr[:,:2]
```

```
array([[ 1,  2],
       [ 5,  6],
       [ 9, 10],
       [13, 14],
       [17, 18]])
```

Virgünden öncesi satırlar için, tümünü aldık, virgülden sonrası sütunlar için 2' ye kadar 2 dahil değil.

```
arr[:2,:]
```

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

```
arr[:2]
```

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

array'leri filtreleme

```
arr > 5
```

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

```
arr = np.arange(1,11)
arr
```

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

```
booleanArray = arr > 5
booleanArray
```

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

```
arr[booleanArray]
```

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

```
arr[arr>6]
```

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

numpy arraylerinin temel operasyonları

```
arr1 = np.array([10,20,30,40,50,60])
arr2 = np.array([2,3,4,5,6,7])
```

```
arr1 + arr2
```

```
array([12, 23, 34, 45, 56, 67])
```

```
arr1 - arr2
```

```
array([ 8, 17, 26, 35, 44, 53])
```

```
arr1 * arr2 #inner product - skaler çarpma
```

```
array([ 20,  60, 120, 200, 300, 420])
```

```
arr1 + 10
```

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

```
arr2 - 4
```

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

arr1 / 6

array([ 1.66666667,  3.33333333,  5.        ,  6.66666667,
8.33333333,
       10.        ])

.sqrt()

np.sqrt(arr1)

array([3.16227766, 4.47213595, 5.47722558, 6.32455532, 7.07106781,
       7.74596669])



https://numpy.org/doc/1.24/reference/routines.linalg.html#module-
numpy.linalg
```