

Numpy Modülü

Arrays

```
import numpy as np
```

Sayıları list olarak kaydederek üzerinde sayısal işlemler yapamayız.

```
a=[1,2,3]
b=[4,5,6]
a+b
[1,2,3,4,5,6] #+ komutu a ve b list olduğu için birleştirme olarak
                algılanır
2*a
[1,2,3,1,2,3]
```

Görüldüğü gibi sayısal değerler list tipinde tutulduğunda sayısal işlemler yapılamaz. Bunun için bu tipteki yapıları np.array tipinde yazmak gerekir.

```
a=np.array([1,2,3])
b=np.array([4,5,6])
a+b
array([5,7,9]) #karşılıklı elemanlar toplandı
2*a
array([2,4,6]) #a'nın her elemanı 2 ile çarpıldı.
```

İki boyutlu Array (Matris)

np.array'ler tek boyutlu olmak zorunda değildir.

```
A=np.array([[1,2,3], [4,5,6]])
print(A)
[[1 2 3]
 [4 5 6]]
```

Numpy ile matris oluştururken np.array() fonksiyonunun içine her matrisin her bir satırı liste şeklinde yazılır.

Başlangıç Array'i

np.zeros komutu ile bütün elemanları 0 olan bir başlangıç arrayi (bir yada iki boyutlu) oluşturabiliriz.

```
a=np.zeros((2,3)) #2x3 boyurunda elemanları 0 olan bir matris
                oluştur.
print(a)
[[0. 0. 0.]
 [0. 0. 0.]
```

np.zeros komutuna benzer olarak np.ones ile bütün elemanları 1 olan bir başlangıç arrayi (bir yada iki boyutlu) oluşturabiliriz.

```
b=np.ones((2,3)) #2x3 boyurunda elemanları 1 olan bir matris
cccccccccccccccccoluřtur.
print(b)
[[1. 1. 1.]
 [1. 1. 1.]
```

Array'in Elemanlarını Çağırarak

np.array tek boyutlu ise list'teki gibi 0 dan başlayarak arrayin elemanlarını çağırabiliriz.

```
b=np.array([-3,6 ,7])
b[1]
6
```

np.array iki boyutlu ise (matris ise) I satırın j. elemanını $[(i, j)]$ ile çağırabiliriz.

```
A=np.array([[2,4,6, 9], [-1,6,4,1], [7,0, 0,-1]])
print(A)
[[ 2 4 6 9]
 [-1 6 4 1]
 [ 7 0 0 -1]]
A[(1,2)]
4
```

Bu matrisin herhangi bir i. satırının tamamını çağırarak için: $A[i, :]$

```
A[1, :] #A'nın 1. satırını çağıralım
array([-1, 6, 4, 1])
```

Benzer şekilde matrisin herhangi bir j. sütunu çağırarak için: $A[:, j]$

```
A[:, 2] #A'nın 2. sütununu çağıralım
array([6,4,0])
```

A'nın ilk iki satırını çağırarak.

```
A[0:2, :]
array([[2, 4, 6, 9],
       [-1, 6, 4, 1]])
```

Not: Yukarıdaki örnekte ilk iki satırı çağırırken, aslında bu satırların indisleri 0 ve 1 idi. Fakat kodda $[0:2, 0]$ yazmamız gertektir; eğer $[0:1, 0]$ yazsaydık yalnızca birinci satır gelecekti.

Benzer şekilde A'nın son iki sütununu çağırarak.

```
A[:, 2:4]
array([[6, 9],
       [4, 1],
       [0, -1] ])
```

np.shape komutu

`np.shape()` komutu aldığı `np.array`'in boyutlarını getirir.

```
np.shape(A)
(3, 4)
```

Eğer özel olarak yalnızca satır sayısını almak istiyorsak: `np.shape(A)[0]`

```
np.shape(A)[0]
3
```

Eğer yalnızca sütun sayısını almak istiyorsak: `np.shape(A)[1]`

```
np.shape(A)[1]
4
```

Şimdi `np.shape()`'i kullanarak son iki sütunu getirelim:

```
A[:, np.shape(A)[1]-2: np.shape(A)[1]]
array([[ 6,  9],
       [ 4,  1],
       [ 0, -1]])
```

np.size komutu

`np.size()`, `np.shape()` komutu gibidir, arrayin boyutları getirilirken kullanılır.

A iki boyutlu array (matris) iken `np.size(A)`, A'nın satır sayısı ile sütun sayısının çarpımını verir.

```
np.size(A) #A burada 3 x4 matris
12
```

Öte yandan `np.size()` ile satır yada sütun sayısını getirmek biraz daha kolaydır.

`np.size(A, 0)` satır sayısını, `np.size(A, 1)` sütun sayısını getirir.

```
np.size(A, 0)
3
np.size(A, 1) #A burada 3 x4 matris
4
```

np.arange komutu

`np.arange(n)` : n uzunluğunda, elemanları 0'dan n'ye kadar olan (n dahil değil) bir array oluşturur.

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

Boolean İndeksleme

Bir `np.array`'in elemanlarından belirli bir şarta uyanlar getirilebilir.

```
b=np.array([-3, 5, 8, -1, 2])
b[b>0] #b'nin 0'dan büyük elemanlarını getir
```

```
b=array([5, 8, 2])
b[b%2==0] #b'nin 2'ye bölümünden kalan 0 olanlari getir.
```

Matris icinde Boolean indeksleme yapılabilir.

```
A=np.array([[2,4,6, 9], [-1,6,4,1], [7,0, 0,-1]])
A[A>0]
array([2, 4, 6, 9, 6, 4, 1, 7])
```

np.dot komutu

np.dot(u,v) ile u ve v gibi aynı uzunluktaki iki array'in iç çarpımını (nokta çarpımını) yapabiliriz

(iç çarpım: iki vektörün karşılıklı elemanlarının çarpımının toplamı)

```
u=np.array([3,-1, 4])
v=np.array([2,0, 12])
np.dot(u,v)
54 #6+0+48
```

****np.dot() aynı zamanda matris- matris çarpımında kullanılır. ****

```
A=np.array([[2,4,6, 9], [-1,6,4,1], [7,0, 0,-1]]) # 3x4 matris
C=np.array([[2,5], [0,4], [-1,3], [3,2]]) # 4x2 matris
np.dot(A,C) ^bu carpimla 3x2 matris elde edilir.
array ([ [25,62],
          [-3, 33],
          [11, 33] ])
```

np.sqrt komutu

np.sqrt() komutu, tek bir sayı alıyorsa bunun karekökünü bulur yada array alıyorsa arraydeki her elemanın karekökünü bulur.

```
u=np.array([3,16, 4])
np.sqrt(u)
array([1.73205081, 4. ,2. ])
```

np.sum komutu

np.sum() komutu, aldığı arraydeki bütün elemanların toplamını verir.

```
u=np.array([3,16, 4])
np.sum(u)
23
```

Bir matrisin satır- sütun toplamı için de np.sum kullanılır. A bir matris iken, A'nın sütunlarının toplamı: np.sum(A, 0) ile, satırlarının toplamı np.sum(A, 1) ile bulunur.

```
print(A)
[[ 2  4  6  9 ]
 [-1  6  4  1 ]]
```

```
[7 0 0 -1]]
np.sum(A,0)
array([8, 10, 10, 9])
np.sum(A,1)
array([21, 10, 6])
```

Broadcasting Özelliği

Numpy'in broadcasting özelliği ile farklı uzunluktaki arraylerle sayısal işlemler yapabiliriz.

```
A=np.array([[1,2],[3,4],[5,6]])
b=np.array([1,1])
A+b
array([[2, 3],
       [4, 5],
       [6, 7]])
```

Yukarıdaki örnekte A, 3x2 boyutunda bir matris ve b, 1x2 uzunluğunda bir matris olmasına rağmen python toplama yaparken hata vermedi. Burada olan, python'un iki matrisi toplayabilmek için b'yi üç satır halinde yazarak çoğalttığı böylece b'yi de A ile aynı boyuta getirdiği (3x2), böylece toplamaı gerçekleştirdiğidir.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix}$$

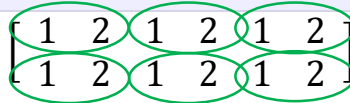
Soru: Eğer b=[1, 1, 1] olsaydı A ile toplanabilir miydi?

np.tile komutu

np.tile aldığı arrayi istenilen boyutlarda çoğaltır.

Örneğin v=np.array([1, 2]) olsun. 2x3 bir matris oluşturarak, bu matrisin her bir elemanına [1,2] yazmak için:

```
v=np.array([1,2])
np.tile(v,(2,3))
array([[1, 2, 1, 2, 1, 2],
       [1, 2, 1, 2, 1, 2]])
```



$\begin{bmatrix} 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 2 & 1 & 2 \end{bmatrix}$

np.where komutu

np.where(arr==3) arrr arrayinde 3 olan yerin positionini bulur.