Veri Manipülasyonu - Numpy

- **Numpy ile Array programlama, vektörler, matrisler ve çok boyutlu dizilerdeki verilere erişmek, bunları işlemek ve bunlar üzerinde çalışmak için güçlü, kompakt ve anlamlı bir syntax imkanı sağlar.**
- **Fizik, kimya, astronomi, yerbilimi, biyoloji, psikoloji, malzeme bilimi, mühendislik, finans ve ekonomi gibi çeşitli alanlarda önemli bir role sahiptir. - Örnek olarak, astronomide NumPy, yerçekimi dalgalarının keşfinde ve bir kara deliğin ilk görüntülenmesinde kullanılan yazılım yığınının önemli bir parçası olmuştur.**
- **NumPy, bilimsel Python ekosisteminin üzerine inşa edildiği temeldir.**
- **Özel ihtiyaçları olan kitleleri hedefleyen birçok şirket projelerini NumPy benzeri arayüzlerini geliştirmişlerdir.**
- **Ekosistemdeki merkezi konumu sayesinde NumPy, bu tür bilimsel hesaplamalar konusunda hızlı ve başarılı olması sayesinde gelecek on yıllık bilimsel ve endüstriyel analizleri desteklemek için esnek bir çerçeve sağlar.**
- **Matematiksel ve İstatistiksel methodları kullanarak veriyi düzenlemek**
- **Elimizdeki veriyi makine öğrenimi veya derin öğrenme modellerimize uygun hale getirmek**
- **Çalışmalarımızı tablolaştırmak veya pandas ve seaborn yardımıyla basit anlamda görselleştirmek**
- **İstatistiksel incelemeler(betimsel istatistikler vb.), Matris veya çoklu değişkenler barındıran denklemler üzerinde işlemler yapmak**
- **Veri okuma ve veri üzerinde işlemler yaparken hız kazanmak(Numpy, C++ dili kullanılarak yazılmıştır. Bu nedenle güçlü ve hız gerektiren işlemler yapmak için en ideal kütüphanelerden biridir.)**
- **Numpy bilgisayarlarımızdaki bellek tüketiminde tasarruf etmemizde de bize yardımcı olabilecek kadar başarılı bir kütüphanedir.**

In [ ]:

```
# Listeler yardımıyla array oluşturma
A = [[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]]


print("Type:", type(A))
print("A =", A)
print("A[0] =", A[0])       # 1. satır
print("A[1] =", A[1])       # 2. satır
print("A[1][2] =", A[1][2])    # 2.satırın 3. elementi
print("A[0][-1] =", A[0][-1])    # 1. satırın son elementi

column = [];         # boş liste
for row in A:
  column.append(row[2])
```

```
Type: <class 'list'>
A = [[1, 4, 5, 12], [-5, 8, 9, 0], [-6, 7, 11, 19]]
A[0] = [1, 4, 5, 12]
A[1] = [-5, 8, 9, 0]
A[1][2] = 9
A[0][-1] = 12
```

Numpy Array'i oluşturmak

In [ ]:

```
import numpy as np
```

In [ ]:

```
# Numpy kullanarak array oluşturma
array1 = np.array([1, 2, 3, 4])
array2 = np.array([10, 20, 30, 40])
```

In [ ]:

```
array1 * array2
```

Out[ ]:

```
array([ 10,  40,  90, 160])
```

In [ ]:

```
type(array1)
```

Out[ ]:

```
numpy.ndarray
```

In [ ]:

```
np.array([7.12, 4.34, 2, 13])
```

Out[ ]:

```
array([ 7.12,  4.34,  2.  , 13.  ])
```

In [ ]:

```
# array tipini değiştirmek için dtype argümanını kullanırız.
np.array([7.12, 4.34, 2, 13], dtype="float32")
```

Out[ ]:

```
array([ 7.12,  4.34,  2.  , 13.  ], dtype=float32)
```

In [ ]:

```
np.array([3.14, 4, 2, 13], dtype="int")
```

Out[ ]:

```
array([ 3,  4,  2, 13])
```

In [ ]:

```
np.zeros((5,5), dtype=int)
```

Out[ ]:

```
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]])
```

In [ ]:

```
np.zeros(8, dtype=int)
```

Out[ ]:

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

In [ ]:

```
np.ones((3,5), dtype=int)
```

Out[ ]:

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

In [ ]:

```
np.ones((3,5), dtype=int)  * 3
```

Out[ ]:

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

In [ ]:

```
np.zeros((2,5), dtype=int)
```

Out[ ]:

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

In [ ]:

```
np.full((3,5), 3)
```

Out[ ]:

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

In [ ]:

```
s5 = np.eye(5)
s5
```

Out[ ]:

```
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 [ ]:

```
# Python içerisinde bulunan range fonksiyonu gibi düşünebiliriz. belirli aralıkta sayı ür
etmek için kullanılır
np.arange(0, 31, 3)
```

Out[ ]:

```
array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27, 30])
```

In [ ]:

```
# Belirttiğimiz aralıkta eşit aralıklı sayılar döndürmek için kullanırız.
np.linspace(0, 1, 10)
```

Out[ ]:

```
array([0.        , 0.11111111, 0.22222222, 0.33333333, 0.44444444,
       0.55555556, 0.66666667, 0.77777778, 0.88888889, 1.        ])
```

Examples

Draw samples from the distribution:

```
>>> mu, sigma = 0, 0.1 # mean and standard deviation
>>> s = np.random.normal(mu, sigma, 1000)
```

In [ ]:

```
# Normal (Gauss) bir dağılımdan rastgele örnekler çizmek için kullanılır.
```

```
np.random.normal(10, 4, (3,4))
```

Out[ ]:

```
array([[10.04044979, 17.83443059,  8.00940101, 15.47998922],
       [12.62146461,  9.72559348, 17.18583621,  8.86033077],
       [ 4.71471194,  7.31620795, 12.0010429 , 17.82896252]])
```

In [ ]:

```
# random integer değerlerden oluşan bir numpy array'i oluşturmak için kullanılır.
np.random.randint(0, 10, (3,3))
```

Out[ ]:

```
array([[4, 0, 7],
       [3, 3, 7],
       [3, 0, 5]])
```

Numpy Array Özellikleri

- **ndim: boyut sayısı**
- **shape: boyut bilgisi**
- **size: toplam eleman sayısı**
- **dtype: array veri tipi**

In [ ]:

```
np.random.randint(10, size=10)
```

Out[ ]:

```
array([3, 2, 2, 2, 0, 3, 9, 8, 8, 8])
```

In [ ]:

```
a = np.random.randint(10, size=10)
a
```

Out[ ]:

```
array([0, 8, 3, 6, 6, 2, 0, 7, 9, 9])
```

In [ ]:

```
a.ndim
```

Out[ ]:

```
1
```

In [ ]:

```
a.shape
```

Out[ ]:

```
(10,)
```

In [ ]:

```
a.size
```

Out[ ]:

```
10
```

In [ ]:

```
a.dtype
```

Out[ ]:

```
dtype('int64')
```

In [ ]:

```
b = np.random.randint(10, size=(3,5))
b
```

Out[ ]:

```
array([[2, 8, 1, 1, 3],
       [2, 2, 8, 1, 7],
       [0, 4, 1, 0, 9]])
```

In [ ]:

```
b.ndim
```

Out[ ]:

```
2
```

In [ ]:

```
b.shape
```

Out[ ]:

```
(3, 5)
```

In [ ]:

```
b.size
```

Out[ ]:

```
15
```

In [ ]:

```
b.dtype
```

Out[ ]:

```
dtype('int64')
```

Reshaping(Array şekillendirme)

In [ ]:

```
np.arange(0, 20).reshape(4,5)
```

Out[ ]:

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

In [ ]:

```
reshape_array = np.arange(0, 20)
reshape_array
```

Out[ ]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

In [ ]:

```
reshape_array.ndim
```

Out[ ]:

```
1
```

In [ ]:

```
reshape_array_dim = reshape_array.reshape((4,5)).ndim
reshape_array_dim
```

Out[ ]:

```
2
```

In [ ]:

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

Out[ ]:

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

In [ ]:

```
# çok boyutlu arrayin vektörel hale gelmesini sağladık
reshape_array.ravel()
```

Out[ ]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

Concatenation Method(Array Birleştirme)

In [ ]:

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

In [ ]:

```
np.concatenate([x,y])
```

Out[ ]:

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

In [ ]:

```
z = np.array([7, 8, 9])
```

In [ ]:

```
np.concatenate([x, y, z])
```

Out[ ]:

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

In [ ]:

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

In [ ]:

```
# satır bazında bir birleştirme yapmış olduk. Burada axis argümanı default özelliğini kul
lanıyor.
```

```
# default olarak argümanımız axis=0'dur
np.concatenate([a, a])
```

Out[ ]:

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

In [ ]:

```
# axis argümanı sütun bazında bir birleştirme yapmak isterse, axis=1 olmalıdır
np.concatenate([a, a], axis=1)
```

Out[ ]:

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

Splitting(Array Ayırma)

In [ ]:

```
# Arrayinizi birden çok alt diziye bölmek için kullanabilirsiniz
x = np.array([12,21, 23, 19, 100, 3, 2, 1])
x
```

Out[ ]:

```
array([ 12,  21,  23,  19, 100,   3,   2,   1])
```

In [ ]:

```
np.split(x, [3,5])
```

Out[ ]:

```
[array([12, 21, 23]), array([ 19, 100]), array([3, 2, 1])]
```

In [ ]:

```
np.split(x, [1,3,5])
```

Out[ ]:

```
[array([12]), array([21, 23]), array([ 19, 100]), array([3, 2, 1])]
```

In [ ]:

```
array1, array2, array3 = np.split(x, [3,5])
```

In [ ]:

```
print(array1)
print(array2)
print(array3)
```

```
[12 21 23]
[ 19 100]
[3 2 1]
```

In [ ]:

```
array_split = np.arange(16).reshape(4,4)
array_split
```

Out[ ]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

[12, 13, 14, 15]])

In [ ]:

```
# Diziyi dikey olarak (satır bazında) birden çok alt diziye bölmek için kullanılır
np.vsplit(array_split, [2])
```

Out[ ]:

```
[array([[0, 1, 2, 3],
        [4, 5, 6, 7]]), array([[ 8,  9, 10, 11],
        [12, 13, 14, 15]])]
```

In [ ]:

```
array_split1, array_split2 = np.vsplit(array_split, [2])
```

In [ ]:

```
array_split1
```

Out[ ]:

```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [ ]:

```
array_split2
```

Out[ ]:

```
array([[ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [ ]:

```
array_split
```

Out[ ]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [ ]:

```
# Bir diziyi yatay olarak (sütun bazında) birden çok alt diziye ayırmak için kullanılır
array_split_right, array_split_left = np.hsplit(array_split, [2])
```

In [ ]:

```
array_split_right
```

Out[ ]:

```
array([[ 0,  1],
       [ 4,  5],
       [ 8,  9],
       [12, 13]])
```

In [ ]:

```
array_split_left
```

Out[ ]:

```
array([[ 2,  3],
       [ 6,  7],
       [10, 11],
       [14, 15]])
```

Sorting(Sıralama)

In [ ]:

```
sorting = np.array([23, 34, 123, 44, 87, 34, 86])
sorting
```

Out[ ]:

```
array([ 23,  34, 123,  44,  87,  34,  86])
```

In [ ]:

```
np.sort(sorting)
```

Out[ ]:

```
array([ 23,  34,  34,  44,  86,  87, 123])
```

In [ ]:

```
sorting
```

Out[ ]:

```
array([ 23,  34, 123,  44,  87,  34,  86])
```

In [ ]:

```
sorting.sort()
```

In [ ]:

```
sorting
```

Out[ ]:

```
array([ 23,  34,  34,  44,  86,  87, 123])
```

In [ ]:

```
array1 = np.random.normal(12, 4, (4,4))
array1
```

Out[ ]:

```
array([[20.16406276, 16.33595264, 16.08531066,  8.5238828 ],
       [14.56537613, 17.13399125, 12.87368681,  3.2549271 ],
       [ 2.63415952,  9.97411793, 16.87537586, 10.05397595],
       [12.34671858, 10.57437578, 16.52861619,  7.0935615 ]])
```

In [ ]:

```
# Bir diziyi sütun bazında sıralamak için kullanılır
np.sort(array1, axis=0)
```

Out[ ]:

```
array([[ 2.63415952,  9.97411793, 12.87368681,  3.2549271 ],
       [12.34671858, 10.57437578, 16.08531066,  7.0935615 ],
       [14.56537613, 16.33595264, 16.52861619,  8.5238828 ],
       [20.16406276, 17.13399125, 16.87537586, 10.05397595]])
```

In [ ]:

```
# Bir diziyi satır bazında sıralamak için kullanılır
np.sort(array1, axis=1)
```

Out[ ]:

```
array([[ 8.5238828 , 16.08531066, 16.33595264, 20.16406276],
       [ 3.2549271 , 12.87368681, 14.56537613, 17.13399125],
       [ 2.63415952,  9.97411793, 10.05397595, 16.87537586],
       [ 7.0935615 , 10.57437578, 12.34671858, 16.52861619]])
```

```
       [ 7.0953615 , 10.57437578, 12.34671838, 10.52861619]])
```

Indexing(indeksleme işlemleri)

In [ ]:

```
array3 = np.random.randint(20, size=20)
array3
```

Out[ ]:

```
array([18, 19, 17, 14, 10,  3,  4,  0, 15, 14, 16, 13, 11, 16,  0, 13, 15,
        1, 12,  0])
```

In [ ]:

```
array3[0]
```

Out[ ]:

```
18
```

In [ ]:

```
array3[-1]
```

Out[ ]:

```
0
```

In [ ]:

```
array3[0] = 100
```

In [ ]:

```
array3
```

Out[ ]:

```
array([100,  19,  17,  14,  10,   3,   4,   0,  15,  14,  16,  13,  11,
        16,   0,  13,  15,   1,  12,   0])
```

In [ ]:

```
array4 = np.random.randint(20, size=(4,5))
array4
```

Out[ ]:

```
array([[10, 17, 12,  3,  9],
       [ 9, 17, 17,  0, 11],
       [ 7,  5,  0, 16,  1],
       [11, 18,  8,  7,  4]])
```

In [ ]:

```
array4[0,0]
```

Out[ ]:

```
10
```

In [ ]:

```
array4[2,4]
```

Out[ ]:

```
1
```

In [ ]:

```
array4[1,4] = 1200
```

In [ ]:

```
array4
```

Out[ ]:

```
array([[  10,   17,   12,    3,    9],
       [   9,   17,   17,    0, 1200],
       [   7,    5,    0,   16,    1],
       [  11,   18,    8,    7,    4]])
```

In [ ]:

```
array4[2,4] = 4.21
```

In [ ]:

```
array4
```

Out[ ]:

```
array([[  10,   17,   12,    3,    9],
       [   9,   17,   17,    0, 1200],
       [   7,    5,    0,   16,    4],
       [  11,   18,    8,    7,    4]])
```

Slicing(Arrayler'in alt kümelerine ayırmak)

In [ ]:

```
array_slicing = np.arange(0, 15)
array_slicing
```

Out[ ]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [ ]:

```
array_slicing[0:5]
```

Out[ ]:

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

In [ ]:

```
array_slicing[:7]
```

Out[ ]:

```
array([0, 1, 2, 3, 4, 5, 6])
```

In [ ]:

```
array_slicing[3:]
```

Out[ ]:

```
array([ 3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [ ]:

```
array_slicing[1::2]
```

Out[ ]:

```
array([ 1,  3,  5,  7,  9, 11, 13])
```

In [ ]:

```
array_slicing[0::3]
```

Out[ ]:

```
array([ 0,  3,  6,  9, 12])
```

In [ ]:

```
m = np.random.randint(10, size=(5,5))
m
```

Out[ ]:

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

In [ ]:

```
m[:,0]
```

Out[ ]:

```
array([5, 0, 6, 4, 6])
```

In [ ]:

```
m[:,1]
```

Out[ ]:

```
array([0, 0, 0, 9, 6])
```

In [ ]:

```
m[1,:]
```

Out[ ]:

```
array([0, 0, 1, 7, 2])
```

In [ ]:

```
m[1]
```

Out[ ]:

```
array([0, 0, 1, 7, 2])
```

In [ ]:

```
m[1,2:3]
```

Out[ ]:

```
array([1])
```

In [ ]:

```
m[0:2,0:3]
```

Out[ ]:

```
array([[5, 0, 8],
       [0, 0, 1]])
```

Subset Operations(Subset işlemleri)

In [ ]:

```
a = np.random.randint(10, size=(5,5))
```

```
a = np.random.randint(10, size=(5,5))
a
```

Out[ ]:

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

In [ ]:

```
subset_a = a[0:3, 0:2]
subset_a
```

Out[ ]:

```
array([[2, 0],
       [3, 8],
       [2, 9]])
```

In [ ]:

```
subset_a[0, 0] = 1234
subset_a[1,1] = 556
```

In [ ]:

```
subset_a
```

Out[ ]:

```
array([[1234,    0],
       [   3,  556],
       [   2,    9]])
```

In [ ]:

```
a
```

Out[ ]:

```
array([[1234,    0,    7,    7,    0],
       [   3,  556,    7,    2,    5],
       [   2,    9,    1,    1,    9],
       [   6,    6,    1,    4,    7],
       [   3,    7,    6,    7,    7]])
```

In [ ]:

```
m = np.random.randint(10, size=(5,5))
m
```

Out[ ]:

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

In [ ]:

```
subset_m = m[0:3, 0:2].copy()
subset_m
```

Out[ ]:

```
array([[4, 2],
       [9, 5],
       [0, 0]])
```

In [ ]:

```
subset_m[0,0] = 5435
```

In [ ]:

```
subset_m
```

Out[ ]:

```
array([[5435,    2],
       [   9,    5],
       [   0,    0]])
```

In [ ]:

```
m
```

Out[ ]:

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

Fancy Indexing

In [ ]:

```
v = np.arange(0, 30, 3)
v
```

Out[ ]:

```
array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27])
```

In [ ]:

```
v[1]
```

Out[ ]:

```
3
```

In [ ]:

```
v[3]
```

Out[ ]:

```
9
```

In [ ]:

```
v[5]
```

Out[ ]:

```
15
```

In [ ]:

```
[v[1], v[3], v[5]]
```

Out[ ]:

```
[3, 9, 15]
```

In [ ]:

```
indis_value = [1, 3, 5]
```

In [ ]:

```
v
```

Out[ ]:

```
array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27])
```

In [ ]:

```
v[indis_value]
```

Out[ ]:

```
array([ 3,  9, 15])
```

In [ ]:

```
m = np.arange(9).reshape((3,3))
m
```

Out[ ]:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [ ]:

```
row = np.array([0, 1])
column = np.array([1, 2])
```

In [ ]:

```
m[row, column]
```

Out[ ]:

```
array([1, 5])
```

In [ ]:

```
m
```

Out[ ]:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [ ]:

```
m[1, [1,2]]
```

Out[ ]:

```
array([4, 5])
```

In [ ]:

```
m[1, [0,2]]
```

Out[ ]:

```
array([3, 5])
```

In [ ]:

```
m[0, [1,2]]
```

Out[ ]:

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

In [ ]:

```
m[0:, [1,2]]
```

Out[ ]:

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

In [ ]:

```
m[[1,2], 0:]
```

Out[ ]:

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

Logical Operations

In [ ]:

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

Out[ ]:

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

In [ ]:

```
v > 5
```

Out[ ]:

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

In [ ]:

```
v < 3
```

Out[ ]:

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

In [ ]:

```
v[v < 3]
```

Out[ ]:

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

In [ ]:

```
v[v > 3]
```

Out[ ]:

```
array([4, 5])
```

In [ ]:

```
v[v==3]
```

Out[ ]:

```
array([3])
```

In [ ]:

```
v[v >= 3]
```

Out[ ]:

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

In [ ]:

```
v[v <= 3]
```

Out[ ]:

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

In [ ]:

```
v[v !=3 ]
```

Out[ ]:

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

In [ ]:

```
v
```

Out[ ]:

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

In [ ]:

```
v * 2
```

Out[ ]:

```
array([ 2,  4,  6,  8, 10])
```

In [ ]:

```
v / 5
```

Out[ ]:

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

In [ ]:

```
v * 5 / 10
```

Out[ ]:

```
array([0.5, 1. , 1.5, 2. , 2.5])
```

In [ ]:

```
v ** 2
```

Out[ ]:

```
array([ 1,  4,  9, 16, 25])
```

Mathematical Operations

In [ ]:

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

Out[ ]:

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

In [ ]:

```
v = 1
```

```
v - 1
```

Out[ ]:

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

In [ ]:

```
v * 5
```

Out[ ]:

```
array([ 5, 10, 15, 20, 25])
```

In [ ]:

```
v / 5
```

Out[ ]:

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

In [ ]:

```
v * 5 / 10 - 1
```

Out[ ]:

```
array([-0.5,  0. ,  0.5,  1. ,  1.5])
```

In [ ]:

```
np.subtract(v, 1)
```

Out[ ]:

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

In [ ]:

```
np.add(v, 1)
```

Out[ ]:

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

In [ ]:

```
np.multiply(v, 4)
```

Out[ ]:

```
array([ 4,  8, 12, 16, 20])
```

In [ ]:

```
np.divide(v, 3)
```

Out[ ]:

```
array([0.33333333, 0.66666667, 1.        , 1.33333333, 1.66666667])
```

In [ ]:

```
np.power(v, 3)
```

Out[ ]:

```
array([  1,   8,  27,  64, 125])
```

In [ ]:

```
np.mod(v, 2)
```

Out[ ]:

```
array([1, 0, 1, 0, 1])
```

In [ ]:

```
np.absolute(np.array([-3,-1]))
```

Out[ ]:

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

In [ ]:

```
np.sin(360)
```

Out[ ]:

```
0.9589157234143065
```

In [ ]:

```
np.cos(180)
```

Out[ ]:

```
-0.5984600690578581
```

In [ ]:

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

In [ ]:

```
np.log(v)
```

Out[ ]:

```
array([0.        , 0.69314718, 1.09861229])
```

In [ ]:

```
np.log10(v)
```

Out[ ]:

```
array([0.        , 0.30103   , 0.47712125])
```

In [ ]:

```
np.mean(v)
```

Out[ ]:

```
2.0
```

In [ ]:

```
v = np.random.randint(1,20, size=(4,4))
v
```

Out[ ]:

```
array([[ 7,  2,  3, 18],
       [14, 18, 15, 13],
       [ 2,  1,  8, 11],
       [ 9,  4, 11, 18]])
```

In [ ]:

```
print(np.mean(v, axis=0)) # returns mean along specific axis
print(v.sum()) # returns the sum of array
print(v.min()) # returns the min value of array
print(v.max()) # returns the max value of array
print(v.max(axis=0)) # returns the max value of specific axis
```

```
print(v.var()) # returns the variance of array
print(np.std(v, axis=1)) # returns the standard deviation of specific axis
print(np.corrcoef(v)) # returns correlation coeffcient of arrray
```

```
[ 8.    6.25  9.25 15.  ]
154
1
18
[14 18 15 18]
34.359375
[6.34428877 1.87082869 4.15331193 5.02493781]
[[ 1.         -0.77933605  0.69260461  0.88614506]
 [-0.77933605  1.         -0.67566392 -0.87758509]
 [ 0.69260461 -0.67566392  1.          0.92237069]
 [ 0.88614506 -0.87758509  0.92237069  1.        ]]
```

In [ ]:

```
# matris çarpımı
a = np.array([9, 10])
b = np.array([11,12])
np.dot(a, b)
```

Out[ ]:

```
219
```

In [ ]:

```
a = np.array([[1, 2,3],
              [4,5,6],
              [7,8,9]])
b = np.array([[10,20],
              [30,40],
              [50,60]])
np.dot(a, b)
```

Out[ ]:

```
array([[ 220,  280],
       [ 490,  640],
       [ 760, 1000]])
```