Numpy

Prof.Dr. Bahadır AKTUĞ
JFM212 Python ile Mühendislik Uygulamaları

**Kaynakça bölümünde verilen kaynaklardan derlenmiştir.*

Numpy

- Numpy, Python'daki dizilerden (list, tuple) farklı olarak her bir elemanının aynı türde olduğu sayısal dizileri içermektedir.
- Özellikle matematiksel işlemlerde numpy dizileri (arrays) önemli kolaylıklar sağlamaktadır.
- Numpy kütüphanesi Python dilinin doğrudan bir parçası olmayıp, ayrıca kurulması ve program içinde tanımlanması(import) gerekmektedir.
- Numpy dizi elemanlarına, dizi tanımlandıktan sonra liste elemanlarına benzer şekilde «index»'ler ile erişilebilir.
- Numpy dizileri tek boyutlu (vektör) veya çok boyutlu (matris) olabilir. Dizilerin boyutu eksen (axis)'ler ve rank'ler ile ifade edilir.

Numpy

- Numpy dizilerinin veri tipi "ndarray"dir. Ayrıca "numpy.array" olarak da isimlendirilmiştir.

- Temel ndarray komutları
  - ndarray.ndim
    - Eksen (boyut) sayısı. Dizilerin boyutu için aynı zamanda "rank" ifadesi de kullanılmaktadır
  - ndarray.shape
    - Dizinin boyutları. 2x3'lük bir matris için "shape" (2,3) şeklindedir.
  - ndarray.size
    - Bir dizideki toplam eleman sayısı. 2x3'lük bir matris için "size" 6'dır.

Numpy

- Temel ndarray komutları (devam)

  - ndarray.dtype
    - Dizi içindeki elemanların veri tipi. Temel Python veri tiplerine ilave olarak numpy'nin de numpy.int32, numpy.int16, ve numpy.float64 gibi veri tipleri bulunmaktadır.
  - ndarray.itemsize
    - Dizinin her bir elemanının "byte" cinsinden boyutu. Örneğin, "float64" türünde her bir elemanının boyutu 8, "complex32" türünde her bir elemanının boyutu 4'tür.
  - ndarray.data
    - Dizinin elemanlarının tümü. Genel olarak dizi elemanlarına "indisleme/dilimleme" yöntemiyle erişildiğinden pek ihtiyaç duyulmaz.

Numpy

```
>>> import numpy as np
>>> dizi = np.arange(12).reshape(3,4)
>>> dizi
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> dizi.shape
(3, 4)
>>> dizi.dtype
dtype('int32')
>>> dizi.itemsize
4
>>> dizi.ndim
2
>>> dizi.size
12
```

Numpy

- En sık kullanılan dizi komutlarından "shape" örnekleri:

```
import numpy as np

a = np.array([1, 2, 3])  # Bir boyutlu dizi
b = np.array([[1,2,3],[4,5,6]])   # İki boyutlu dizi

print(a.shape) → (3,) yazacaktır
print(b.shape) → (2,3) yazacaktır
```

Numpy

- Numpy dizileri birkaç şekilde oluşturulabilir:
  - Doğrudan np.array komutu ile standart Python listeleri ve demetleri (tuple) kullanılarak
  - numpy.arange, numpy.ones, numpy.eye, numpy.zeros, numpy.empty, numpy.full,numpy.random, numpy.linspace komutları kullanılarak
- numpy.array komutu ile Python dizilerinden numpy dizileri oluşturulurken veri tipi verilebilir.
- Verilmezse, veriye numpy otomatik olarak veri tipini belirler. Veri tipi verilmediğinde, tamsayılar için varsayılan veri tipi "int32/int64", ondalık sayılar için "float64" tür.

Numpy Veri Tipleri

- Yeni bir Numpy dizisi oluşturulduğunda, en uygun veri tipi Numpy tarafından seçilir. Ancak, istenirse belirli bir veri tipi seçeneği verilebilir:

```
import numpy as np
x = np.array([1, 2])
x.dtype → "int64" veya "int32"

x = np.array([1.0, 2.0])
x.dtype   → "float64"

x = np.array([1, 2], dtype=np.int64)
x.dtype  → "int64"
```

Numpy

```
import numpy as np
>>> a = np.array([1, 2, 3])
>>> b = np.array([[1,2,3],[4,5,6]],dtype='float64')
```

veya

```
>>> b = np.array([[1,2,3],[4,5,6]],dtype=np.float64)
>>> a.dtype
dtype('int32')
>>> a = np.array([.1, 2, 3])
>>> a.dtype
dtype('float64')
>>> b.dtype
dtype('float64')
```

Numpy

- Numpy dizisi oluşturan bazı özel fonksiyonlar :

a = np.zeros((2,2)) # 2x2'lik sıfır matrisi oluşturur.

b = np.ones((1,2))   # # 1x2'lik birler matrisi oluşturur

c = np.full((2,2), 7) # Sabit sayıdan oluşan 2x2'lik bir matris oluşturur

d = np.eye(2)   # 2x2'lik birim matris oluşturur

e = np.random.random((2,2)) # Rasgele sayılardan oluşan 2x2'lik birim matris oluşturur

Numpy

- Numpy dizisi oluşturan bazı özel fonksiyonlar :

```
>>> import numpy as np
>>> np.arange(2,10,2)
array([2, 4, 6, 8])
>>> np.arange(20,0,-5)
array([20, 15, 10,  5])
>>> np.linspace(0,50,9)
array([  0.  ,   6.25,  12.5 ,  18.75,  25.  ,  31.25,  37.5 ,  43.75,  50.  ])
>>> np.linspace(0,50,11)
array([  0.,   5.,  10.,  15.,  20.,  25.,  30.,  35.,  40.,  45.,  50.])
```

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Numpy dizilerinin alt dizilerine dilimleme ile ulaşılabilir:

```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
b = a[:2, 1:3]
b = [[2 3], [6 7]]
```

- Buna karşın, oluşturulan altdizide bir değişiklik yapılırsa ana dizinin de değişeceğini göz önünde bulundurulmalıdır.

b[0, 0] = 77 yapılırsa a[0, 1] → 77 olur

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Numpy dizi elemanlarına toplu atama da yapılabilir:

```
>>> a = np.arange(10)*2
>>> a
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
>>> a[:9:3] = 50
>>> a
array([50,  2,  4, 50,  8, 10, 50, 14, 16, 18])
>>>
```

- ":" operatörünün kullanımı Python standart dizi dilimleme işlemlerindeki gibidir.

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Dizi boyutlarından daha az indis verilirse, indisler sırasıyla "eksenlere" atanır ve eksik kalan boyutlarda ":" operatörü varmış gibi kabul edilir.

```
>>> a = np.array([[2,10,2],[3,7,9],[4,8,1]])
>>> a[-1]
array([4, 8, 1])
>>> a[2,:]
array([4, 8, 1])
>>> a[:,-1]
array([2, 9, 1])
```

Numpy Dizileri Üzerinde Döngü

- Numpy dizileri üzerinde döngü kurulmak istenirse ilk eksen (satır) üzerinde döngü oluşur.

```
>>> a
array([[ 2, 10,  2],
       [ 3,  7,  9],
       [ 4,  8,  1]])
>>> for x in a:
...     print(x)
...
[ 2 10  2]
[3 7 9]
[4 8 1]
```

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Numpy dizilerinin alt dizilerine dilimleme yanında doğrudan satır ve sütun numaraları ile de ulaşılabilir.
- Ancak, satır ve sütun numaraları ile erişim yapılırsa, matris boyutları farklı olur.

```
import numpy as np

a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

row_r1 = a[1, :]    # Rank 1 view of the second row of a
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
row_r1.shape → (4,)
row_r2.shape → (1, 4)
```

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Aynı durum sütun numaraları için de geçerlidir.

```
import numpy as np

a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
col_r1.shape → (3,)
col_r2.shape → (3, 1)
```

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Numpy dizilerine bir başka erişim ise «tamsayı dizileri» ile yapılır:

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
a[[0, 1, 2], [0, 1, 0]] → [1 4 5]
```

- Yukarıdaki indisleme aşağıdaki ile aynıdır:

```
b = np.array([a[0, 0], a[1, 1], a[2, 0]]) → [1 4 5]
```

- Buna karşın, dizinin aynı elemanının birden fazla kullanılmasına izin verir:

```
a[[0, 0], [1, 1]]  → [2 2]
np.array([a[0, 1], a[0, 1]]) → [2 2]
```

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Bir Numpy dizisi başka bir Numpy dizisinin indislerini oluşturabilir:

```
import numpy as np
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
b = np.array([0, 2, 0, 1])
a[np.arange(4), b]  → [ 1  6  7 11]
```

- İstenirse ana dizinin bu elemanları bu şekilde değiştirilebilir:

```
a[np.arange(4), b] += 10
a →  array([[11,  2,  3],
            [ 4,  5, 16],
            [17,  8,  9],
            [10, 21, 12]])
```

Numpy Dizilerinin Dilimlenmesi (Slicing)

- Numpy dizilerinde «Boolean» filtreleme ve indisleme yapılabilir:

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2) → a'nın 2'den büyük elemanlarını bulur
bool_idx → [[False False]
            [ True  True]
            [ True  True]]
a[bool_idx] →[3 4 5 6]
```

- Aynı işlem daha kısa bir şekilde de yapılabilir:

```
print a[a > 2]    → [3 4 5 6]
```

Farklı Boyutlardaki Numpy Dizilerinin Birleştirilmesi (Stacking)

- Farklı boyutlardaki Numpy dizileri yatay veya düşey olarak birleştirilebilir:

```
>>> a = np.array([[2,4,6],[7,1,9]])
>>> b = np.array([[8,3,2],[1,6,0]])
>>> np.vstack((a,b))
array([[2, 4, 6],
       [7, 1, 9],
       [8, 3, 2],
       [1, 6, 0]])
>>> np.hstack((a,b))
array([[2, 4, 6, 8, 3, 2],
       [7, 1, 9, 1, 6, 0]])
```

Vektör / Matris İşlemleri

- Numpy temel vektör ve matris işlemleri için hem operatör (operatör overloading) hem de fonksiyon düzeyinde araçlar sunar:

```
import numpy as np
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)
```

x + y veya np.add(x, y) → Eleman eleman toplama
x - y veya np.subtract(x, y) → Eleman eleman çıkarma
x * y veya np.multiply(x, y) → Eleman elaman çarpma
x / y veya np.divide(x, y) → Eleman elaman bölme
np.sqrt(x) → Eleman elaman karekök

Vektör / Matris İşlemleri

- Normal 4 işlem operatörlerinin Numpy dizileri için anlamının eleman-elaman işlem olduğunu görmüştük.
- Matris çarpımı için ise «dot» fonksiyonu kullanılır:

```
import numpy as np
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])
v = np.array([9,10])
w = np.array([11, 12])
```

v.dot(w) veya np.dot(v, w) → Vektör iç çarpım
x.dot(v) veya np.dot(x, v) → Matris / Vektör çarpım
x.dot(y) veya np.dot(x, y) → Matris / Matris çarpımı

Vektör / Matris İşlemleri

- Matris transpoz, satır/sütun toplamları için de Numpy pratik araçlar sunmaktadır:

```
import numpy as np
x = np.array([[1,2], [3,4]])
x.T → [[1 3], [2 4]]
```

- Vektörler için de 'T' kullanılabilmekle birlikte herhangi bir etkisi yoktur.

```
x = np.array([[1,2],[3,4]])
np.sum(x) → 10 Tüm elemanların toplamı
np.sum(x, axis=0)  → [4 6] sütunların toplamı
np.sum(x, axis=1)  → [3 7] satırların toplamı
```

Vektör / Matris İşlemleri

- Numpy, özellikle büyük matrislerle çalışırken bazı pratik araçlar sunar:

```
import numpy as np
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))  # Stack 4 copies of v on top of each other
vv → [[1 0 1]
      [1 0 1]
      [1 0 1]
      [1 0 1]]
y = x + vv →  [[ 2  2  4
                 [ 5  5  7]
                 [ 8  8 10]
                 [11 11 13]]
```

Kaynakça

1 Wentworth, P., Elkner, J., Downey, A.B., Meyers, C. (2014). How to Think Like a Computer Scientist: Learning with Python (3nd edition).
2 Pilgrim, M. (2014). Dive into Python 3 by. Free online version: DiveIntoPython3.org ISBN: 978-1430224150.
3 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
4 Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/.
5 Millman, K.J., Aivazis, M. (2011). Python for Scientists and Engineers, Computing in Science & Engineering, 13, 9-12.
6 John D. Hunter (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95.
7 Travis E. Oliphant (2007). Python for Scientific Computing, Computing in Science & Engineering, 9, 10-20.
8 Goodrich, M.T., Tamassia, R., Goldwasser, M.H. (2013). Data Structures and Algorithms in Python, Wiley.
9 http://www.diveintopython.net/
10 https://docs.python.org/3/tutorial/
11 http://www.python-course.eu
12 https://developers.google.com/edu/python/
13 http://learnpythonthehardway.org/book/