Python Kütüphaneleri – 1: NumPy

esra.aycan@ikcu.edu.tr

İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ

TÜBİTAK

4005 TÜBİTAK

Numpy

- Sayısal Python tabanlı işlemler için kullanılmakta
- Yüksek performanslı bilgi işlem ve veri analizi için gerekli bir temel paket
  - çok boyutlu diziler oluşturmak için ndarray
  - Döngü yazmak zorunda kalmadan tüm veri dizilerinde hızlı işlemler için standart matematik işlevleri
  - Dizi verilerini okumak/yazmak için araçlar
  - Lineer cebir araçları
  - vb.

ndarray vs liste listesi

- Diyelim ki 10 kişilik bir sınıfta üç sınav (2 ara sınav ve 1 final) notunuz var.
- grades =

```
[[79, 95, 60],
 [95, 60, 61],
 [99, 67, 84],
 [76, 76, 97],
 [91, 84, 98],
 [70, 69, 96],
 [88, 65, 76],
 [67, 73, 80],
 [82, 89, 61],
 [94, 67, 88]]
```

- 0. öğrenci final sınavı notu nasıl alınır?
  - grades [0][2]
- 2. öğrencinin notları nasıl alınır?
  - grades [2]
- 1 ara sınavda tüm öğrencilerin notları nasıl alınır?
- İlk üç öğrencinin (veya tüm kız öğrencilerin veya finalden kalanların) ara sınav notları nasıl alınır?
- Her sınavdan ortalama not nasıl alınır?
- Her öğrenci için (ağırlıklı) ortalama sınav notu nasıl alınır?

ndarray vs liste listesi

- gArray = array(examGrades)

```
In [3]: gArray
Out[3]:
array([[79, 95, 60],
[95, 60, 61],
[99, 67, 84],
...,
[67, 73, 80],
[82, 89, 61],
[94, 67, 88]])
```

```
In [5]: gArray[0,2]
Out[5]: 60

In [7]: gArray[2,:]
Out[7]: array([99, 67, 84])

In [8]: gArray[:, 0]
Out[8]: array([79, 95, 99, 76, 91, 70, 88,
67, 82, 94])

In [9]: gArray[:3, :2]
Out[9]:
array([[79, 95],
[95, 60],
[99, 67]])
```

ndarray

- ndarray homojen verilerin depolanması için kullanılır
  - yani, tüm öğeler aynı türde olmalıdır
- Her dizinin bir şekli ve bir tipi olmalıdır
- Uygun dilimleme, indeksleme ve verimli vektörleştirilmiş hesaplamayı destekler
- Döngüler için kaçının ve çok daha verimli

```
In [15]: type(gArray)
Out[15]: numpy.ndarray

In [16]: gArray.ndim
Out[16]: 2

In [17]: gArray.shape
Out[17]: (10, 3)

In [18]: gArray.dtype
Out[18]: dtype('int32')
```

ndarray

- np.array
- np.zeros
- np.ones
- np.eye
- np.arrange
- np.random

In [65]: np.array([[0,1,2],[2,3,4]])
Out[65]:
array([[0, 1, 2],
[2, 3, 4]])

In [66]: np.zeros((2,3))
Out[66]:
array([[0., 0., 0.],
[0., 0., 0.]])

In [67]: np.ones((2,3))
Out[67]:
array([[1., 1., 1.],
[1., 1., 1.]])

In [69]: np.eye(3)
Out[69]:
array([[1., 0., 0.],
[0., 1., 0.],
[0., 0., 1.]])

In [70]: np.arrange(0, 10, 2)
Out[70]: array([0, 2, 4, 6, 8])

In [295]: np.random.randint(0, 10, (3,3))
Out[295]:
array([[8, 7, 6],
[0, 8, 9],
[9, 0, 4]])

Numpy veri tipleri

- int8, int16, int32, int64
- float16, float32, float64, float128
- bool
- object
- String
- Unicode

64
bits

- gArray.astype

```
In [34]: gArray.astype(float64)
Out[34]:
array([[ 79., 95., 60.],
[ 95., 60., 61.],
...,
[ 82., 89., 61.],
[ 94., 67., 88.]])
```

```
In [79]: num_string = array(['1.0', '2.05', '3'])

In [81]: num_string
Out[81]:
array(['1.0', '2.05', '3'],
dtype='<U4')

In [82]: num_string.astype(float)
Out[82]: array([ 1. , 2.05, 3. ])
```

İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ

TÜBİTAK

4005 TÜBİTAK

Dizi operatörleri

- Diziler ve skalerler arasında
- Eşit boyutlu diziler arasında: eleman bazında işlem

In [**94**]: arr * arr
Out[**94**]:
array([[0, 1, 4],
[9, 16, 25]])

In [**95**]: arr / (arr+1)
Out[**95**]:
array([[0. , 0.5 , 0.66666667],
[0.75 , 0.8 , 0.83333333]])

In [**87**]: arr = array([[0,1,2],[3,4,5]])

In [**88**]: arr * 2
Out[**88**]:
array([[0, 2, 4],
[6, 8, 10]])

In [**90**]: arr ** 2
Out[**90**]:
array([[0, 1, 4],
[9, 16, 25]])

In [**91**]: 2 ** arr
Out[**91**]:
array([[1, 2, 4],
[8, 16, 32]], dtype=int32)

Dizi indeksleme ve dilimleme

- Python listesine biraz benzer, ancak çok daha esnek

```
In [152]: gArray
Out[152]:
array([[79, 95, 60],
       [95, 60, 61],
       [99, 67, 84],
       ...,
       [67, 73, 80],
       [82, 89, 61],
       [94, 67, 88]])
```

```
In [153]: gArray[0]
Out[153]: array([79, 95,
60])

In [154]: gArray[1:3]
Out[154]:
array([[95, 60, 61],
       [99, 67, 84]])

In [155]: gArray[0][2]
Out[155]: 60

In [156]: gArray[0,2]
Out[156]: 60
```

```
In [157]: gArray[:, 2]
Out[157]: array([60, 61, 84, 97, 98, 96,
76, 80, 61, 88])
```

İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ

TÜBİTAK

4005 TÜBİTAK

Dizi indeksleme ve dilimleme

```
In [152]: gArray
Out[152]:
array([[79, 95, 60],
       [95, 60, 61],
       [99, 67, 84],
       ...,
       [67, 73, 80],
       [82, 89, 61],
       [94, 67, 88]])
```

```
In [160]: gArray[:2, [0, 2]]
Out[160]:
array([[79, 60],
       [95, 61]])
```

```
In [175]: gArray[[0, 2], :]
Out[175]:
array([[79, 95, 60],
       [99, 67, 84]])
```

```
In [177]: gArray[[0, 2], [0, 1, 2]]
Traceback (most recent call last):
...
IndexError: shape mismatch: ...

In [178]: gArray[[0, 2], [0, 2]]
Out[178]: array([79, 84])
```

list

```
In [200]: gArray[[0,2]][:,[0,2]]
Out[200]:
array([[79, 60],
       [99, 84]])
```

```
In [272]: gArray[np.ix_([0, 2], [0, 2])]
Out[272]:
array([[79, 60],
       [99, 84]])
```

Dizi dilimleri görünümlerdir

```
In [202]: gArray[0,:]=100

In [203]: gArray
Out[203]:
array([[100, 100, 100],
       [ 95,  60,  61],
       [ 99,  67,  84],
       ...,
       [ 67,  73,  80],
       [ 82,  89,  61],
       [ 94,  67,  88]])
```

```
In [254]: arr2 = gArray.copy()

In [255]: arr2 is gArray
Out[255]: False

In [258]: arr2[1,:]=100

In [260]: gArray[1,:]
Out[260]: array([95, 60, 61])
```

Bir dizinin açıkça bir kopyasını oluşturmak için .copy() kullanın.

Boole indeksleme

```
# kız öğrenciler için kayıt seç
In [262]: female = [ True, False,
True,  True, False,  True, False,
False, False, False]

In [263]: gArray[female, :]
Out[263]:
array([[100, 100, 100],
       [ 99,  67,  84],
       [ 76,  76,  97],
       [ 70,  69,  96]])
```

```
# finalde # <= 70 olanlar için
# kayıt seç,
In [265]: gArray[gArray[:,
2]<70,:]
Out[265]:
array([[95, 60, 61],
       [82, 89, 61]])

# < 70 olan her şey 70 olarak
# değiştirilir
In [267]: gArray[gArray < 70] = 70

In [268]: gArray
Out[268]:
array([[100, 100, 100],
       [ 95,  70,  70],
       [ 99,  70,  84],
       ...,
       [ 70,  73,  80],
       [ 82,  89,  70],
       [ 94,  70,  88]])
```

İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ

TÜBİTAK

4005 TÜBİTAK

Python Kütüphaneleri – 2: Pandas

esra.aycan@ikcu.edu.tr

İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ

TÜBİTAK

4005 TÜBİTAK

Neden pandas?

- Veri bilimcilerin kullandığı en popüler kütüphanelerden biri
- Verilerin yanlış hizalanmasını önlemek için etiketlenmiş eksenler
- Veri[:, 2] ağırlığı mı yoksa ağırlığı mı temsil ediyor2?

	height	Weight	Weight2	age	Gender
Amy	160	125	126	32	2
Bob	170	167	155	-1	1
Chris	168	143	150	28	1
David	190	182	NA	42	1
Ella	175	133	138	23	2
Frank	172	150	148	45	1

	salary	Credit score
Alice	50000	700
Bob	NA	670
Chris	60000	NA
David	-99999	750
Ella	70000	685
Tom	45000	660

Genel Bakış

- 2008'de Wes McKinney tarafından yaratıldı, şimdi Jeff Reback ve diğerleri tarafından sürdürülüyor.
- Ders kitaplarından birinin yazarı: Veri Analizi için Python
- Güçlü ve üretken Python veri analizi ve Yönetim Kitaplığı
- Panel Veri Sistemi
- Açık kaynaklı bir üründür.

Genel Bakış - 2

- R, MATLAB, SAS'a benzer veri analizi özellikleri sağlamak için Python Kitaplığı
- Veri yapısıyla çalışmayı hızlı, kolay ve anlamlı hale getirmek için zengin veri yapıları ve işlevler.
- NumPy üzerine inşa edilmiştir
- Pandalar tarafından sağlanan temel bileşenler:
  - Dizi
  - Veri çerçevesi

```
In [664]: from pandas import Series, DataFrame
In [665]: import pandas as pd
```

Seriler

- Tek boyutlu dizi benzeri nesne
- İlişkili dizinlerle birlikte (herhangi bir NumPy veri türünden) veri dizisini içerir.
- (İndeksler, dizeler veya tamsayılar veya diğer veri türleri olabilir.) Varsayılan olarak, seri, N = size -1 olmak üzere 0'dan N'ye indeksleme alacaktır.

```
In [666]: obj = Series([4, 7, -5, 3])

In [667]: obj
Out[667]:
0 4
1 7
2 -5
3 3
dtype: int64
```

```
In [668]: obj.values
Out[668]: array([ 4, 7, -5, 3], dtype=int64)

In [669]: obj.index
Out[669]: RangeIndex(start=0, stop=4, step=1)
```

Seriler

```
In [670]: obj2 = Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])

In [671]: obj2
Out[671]:
d 4
b 7
a -5
c 3
dtype: int64

In [672]: obj2.index
Out[672]: Index(['d', 'b', 'a', 'c'], dtype='object')

In [673]: obj2.values
Out[673]: array([ 4, 7, -5, 3], dtype=int64)

In [674]: obj2['a']
Out[674]: -5

In [675]: obj2['d']=10

In [677]: obj2[['d', 'c', 'a']]
Out[677]:
d 10
c 3
a -5
dtype: int64

In [692]: obj2[:2]
Out[692]:
d 10
b 7
dtype: int64

In [818]: obj2.a
Out[818]: -5
```

Seri – dizi/sözlük işlemleri

- dizin değeri bağlantısını koruyacak olan numpy dizi işlemleri de uygulanabilir

```
In [694]: obj2[obj2>0]
Out[694]:
d 10
b 7
c 3
dtype: int64
```

```
In [699]: obj2**2
Out[699]:
d 100
b 49
a 25
c 9
dtype: int64
```

```
In [700]: 'b' in obj2
Out[700]: True
```

```
In [702]: obj3 = Series({'a': 10, 'b': 5, 'c': 30})

In [703]: obj3
Out[703]:
a 10
b 5
c 30
dtype: int64
```

Seriler

```
In [704]: sdata = {'Texas': 10, 'Ohio': 20, 'Oregon': 15, 'Utah':
18}

In [705]: states = ['Texas', 'Ohio', 'Oregon', 'Iowa']

In [706]: obj4 = Series(sdata, index=states)

In [707]: obj4
Out[707]:
Texas 10.0
Ohio 20.0
Oregon 15.0
Iowa NaN
dtype: float64
```

Eksik değer

```
In [708]: pd.isnull(obj4)
Out[708]:
Texas False
Ohio False
Oregon False
Iowa True
dtype: bool

In [709]: pd.notnull(obj4)
Out[709]:
Texas True
Ohio True
Oregon True
Iowa False
dtype: bool

In [717]: obj4[obj4.notnull()]
Out[717]:
Texas 10.0
Ohio 20.0
Oregon 15.0
dtype: float64
```

Seri – otomatik hizalama

In [**707**]: obj4
Out[**707**]:
Texas 10.0
Ohio 20.0
Oregon 15.0
Iowa NaN
dtype: float64

In [**714**]: obj5
Out[**714**]:
Ohio 20
Oregon 15
Texas 10
Utah 18
dtype: int64

In [**715**]: obj5 + obj4
Out[**715**]:
Iowa NaN
Ohio 40.0
Oregon 30.0
Texas 20.0
Utah NaN
dtype: float64

Seri adı ve dizin adı

```
In [720]: obj4.name = 'population'

In [721]: obj4
Out[721]:
Texas 10.0
Ohio 20.0
Oregon 15.0
Iowa NaN
Name: population, dtype: float64
```

- Bir dizinin dizini farklı bir dizine değiştirilebilir.
- Dizin nesnesinin kendisi değişmezdir.

```
In [1014]: obj4.index[2]='California'
TypeError: Index does not support mutable operations

In [1016]: obj4.index
Out[1016]: Index(['Florida', 'New York', 'Kentucky', 'Georgia'],
dtype='object')
```

```
In [722]: obj4.index.name = 'state'

In [723]: obj4
Out[723]:
state
Texas 10.0
Ohio 20.0
Oregon 15.0
Iowa NaN
Name: population, dtype: float64

In [725]: obj4.index = ['Florida', 'New
York', 'Kentucky', 'Georgia']

In [726]: obj4
Out[726]:
Florida 10.0
New York 20.0
Kentucky 15.0
Georgia NaN
Name: population, dtype: float64
```

DataFrame (Veri Çerçevesi)

- DataFrame, bir elektronik tablo veya veritabanı tablosuna benzeyen satır ve sütunlardan oluşan tablo şeklinde bir veri yapısıdır.
- Sütunların bir sipariş koleksiyonu olarak kabul edilebilir.
- Her sütun farklı bir veri türü olabilir
- Hem satır hem de sütun indekslerine sahip olur

```
In [727]: data = {'state': ['Ohio', 'Ohio', 'Ohio',
'Nevada', 'Nevada'],
    ...: 'year': [2000, 2001, 2002, 2001, 2002],
    ...: 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}

In [728]: frame = DataFrame(data)

In [729]: frame
Out[729]:
   pop state year
0 1.5 Ohio 2000
1 1.7 Ohio 2001
2 3.6 Ohio 2002
3 2.4 Nevada 2001
4 2.9 Nevada 2002
```

Yeniden sıralı

DataFrame (Veri Çerçevesi)

```
In [727]: data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada',
'Nevada'],
   ...: 'year': [2000, 2001, 2002, 2001, 2002],
   ...: 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}

In [730]: frame2 = DataFrame(data, columns=['year', 'state',
'pop', 'debt'], index=['A', 'B', 'C', 'D', 'E'])

In [731]: frame2
Out[731]:
   year state pop debt
A 2000 Ohio 1.5 NaN
B 2001 Ohio 1.7 NaN
C 2002 Ohio 3.6 NaN
D 2001 Nevada 2.4 NaN
E 2002 Nevada 2.9 NaN
```

Same order

Initialized with NaN

- Sütunların/satırların sırası belirtilebilir.
- Veride olmayan sütunlarda NaN olacaktır.

DataFrame - iç içe sözlüklerden

```
In [838]: pop = {'Nevada': {2001: 2.9, 2002: 2.9}, 'Ohio': {2002: 3.6, 2001: 1.7, 2000: 1.5}}

In [840]: frame3 = DataFrame(pop)

In [841]: frame3
Out[841]:
      Nevada  Ohio
2000     NaN   1.5
2001     2.9   1.7
2002     2.9   3.6
```

İç anahtarların birliği (sıralı sırada)

Transpose

```
In [842]: frame3.T
Out[842]:
       2000  2001
       2002
Nevada  NaN   2.9
        2.9
Ohio    1.5   1.7
        3.6
```

DataFrame – indeks, kolon, değerler

```
In [847]: frame3.index
Out[847]: Int64Index([2000, 2001, 2002], dtype='int64')

In [848]: frame3.columns
Out[848]: Index(['Nevada', 'Ohio'], dtype='object')

In [849]: frame3.values
Out[849]:
array([      [ nan, 1.5],
             [ 2.9, 1.7],
             [ 2.9, 3.6]])
```

```
In [850]: frame3.index.name = 'year';
frame3.columns.name='state'

In [851]: frame3
Out[851]:
state Nevada Ohio
year
2000 NaN 1.5
2001 2.9 1.7
2002 2.9 3.6
```

İndeksleme, seçim ve filtreleme

- Seriler ve DataFrame, etiket tabanlı dizinlerle veya Numpy Dizisine benzer konum tabanlı dizinler kullanılarak dilimlenebilir/erişilebilir

```
In [906]: S = Series(range(4), index=['zero', 'one', 'two', 'three'])

In [907]: S['two']
Out[907]: 2

In [908]: S[['zero', 'two']]
Out[908]:
zero 0
two 2
dtype: int32

In [909]: S[2]
Out[909]: 2

In [910]: S[[0,2]]
Out[910]:
zero 0
two 2
dtype: int32

In [911]: S[:2]
Out[911]:
zero 0
one 1
dtype: int32

In [913]: S['zero':'two']
Out[913]:
zero 0
one 1
two 2
dtype: int32
```

Dahil

```
In [917]: S[S > 1]
Out[917]:
two 2
three 3
dtype: int32

In [995]: S[-2:]
Out[995]:
two 2
three 3
dtype: int32
```

DataFrame – kolon içeriğine erişim

- DataFrame'deki bir sütun, sözlük benzeri gösterimle veya nitelik olarak bir Seri olarak alınabilir.
- Seri indeksi ve adı uygun şekilde tutulmalı/ayarlanmalı:

```
In [734]: frame['state']
Out[734]:
0 Ohio
1 Ohio
2 Ohio
3 Nevada
4 Nevada
Name: state, dtype: object
```

```
In [733]: frame.state
Out[733]:
0 Ohio
1 Ohio
2 Ohio
3 Nevada
4 Nevada
Name: state, dtype: object
```

```
In [805]: type(frame['state'])
Out[805]: pandas.core.series.Series
```

DataFrame – satırlara erişim

- dizinleri kullanmak için **loc** ve konumları kullanmak için **iloc**

```
In [792]: frame2
Out[792]:
   year state pop debt
A 2000 Ohio 1.5 NaN
B 2001 Ohio 1.7 NaN
C 2002 Ohio 3.6 NaN
D 2001 Nevada 2.4 NaN
E 2002 Nevada 2.9 NaN
```

```
In [801]: frame2.loc['A']
Out[801]:
year 2000
state Ohio
pop 1.5
debt NaN
Name: A, dtype: object

In [804]: type(frame2.loc['A'])
Out[804]: pandas.core.series.Series
```

```
In [819]: frame2.loc[['A', 'B']]
Out[819]:
   year state pop debt
A 2000 Ohio 1.5 NaN
B 2001 Ohio 1.7 NaN

In [820]: type(frame2.loc[['A', 'B']])
Out[820]: pandas.core.frame.DataFrame
```

DataFrame – kolonları değiştirmek

```
In [829]: frame2['debt'] = 0

In [830]: frame2
Out[830]:
   year state pop debt
A 2000 Ohio 1.5 0
B 2001 Ohio 1.7 0
C 2002 Ohio 3.6 0
D 2001 Nevada 2.4 0
E 2002 Nevada 2.9 0
```

```
In [831]: frame2['debt'] = range(5)

In [832]: frame2
Out[832]:
   year state pop debt
A 2000 Ohio 1.5 0
B 2001 Ohio 1.7 1
C 2002 Ohio 3.6 2
D 2001 Nevada 2.4 3
E 2002 Nevada 2.9 4
```

```
In [833]: val = Series([10, 10, 10],
index = ['A', 'C', 'D'])

In [834]: frame2['debt'] = val

In [835]: frame2
Out[835]:
   year state pop debt
A 2000 Ohio 1.5 10.0
B 2001 Ohio 1.7 NaN
C 2002 Ohio 3.6 10.0
D 2001 Nevada 2.4 10.0
E 2002 Nevada 2.9 NaN
```

DataFrame – kolonları silmek

```
In [836]: del frame2['debt']

In [837]: frame2
Out[837]:
```

	year	state	pop
A	2000	Ohio	1.5
B	2001	Ohio	1.7
C	2002	Ohio	3.6
D	2001	Nevada	2.4
E	2002	Nevada	2.9