BATCH : B150 Data Science

LESSON : **PANDAS**

DATE : 03.04.2023

SUBJECT : **Session 1- Introduction**

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ

PANDAS

Veri bilimi projeleri, verinin keşfedilmesi ve temizlenmesi ile başlar ve bu işlemler projelerin en çok zaman alan kısımlarıdır

Dolayısıyla verinin keşfi ve temizlenmesi sırasında işleri kolaylaştıracak bir takım kütüphanelere ihtiyaç duyulur.

Hatırlayacağınız üzere Numpy verilerle çalışmayı oldukça kolaylaştırmıştı.

Numpy'ın eksik kaldığı kısımlarda ise imdadımıza Pandas yetişiyor.

Ancak Pandas Numpy' ın bir alternatifi olarak değil, uzantısı olarak düşünülmelidir.

Pandas, Numpy'ın sütun adları ve homojen olmayan verilerle çalışamama gibi eksik kaldığı kısımlara ve daha fazlasına çözümler üretir.

Pandas ile veri analizi yaparken kullanacağımız temel veri yapıları Seriler ve DataFrame'lerdir.

PANDAS

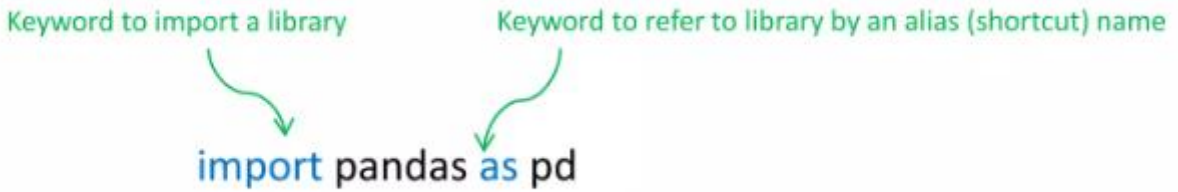
Pandas = Panel Data System

Pandas, Python programlama dili için yüksek performanslı, kullanımı kolay veri yapıları ve veri analiz araçları sağlayan açık kaynaklı bir kütüphanedir.

Pandas, Numpy'ın sütun adları ve homojen olmayan verilerle çalışamama gibi eksik kaldığı kısımlara çözümler üretir.

Pandas ile veri analizi yaparken kullanacağımız temel veri yapıları Seriler ve DataFrame'lerdir.

PANDAS



Used for:

- **Data Analysis**
- **Data Manipulation**
- **Data Visualization**

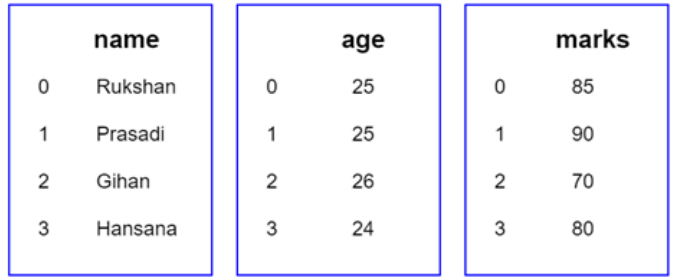
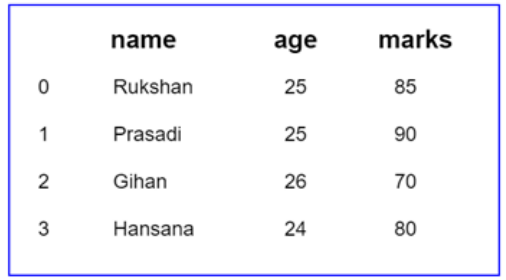
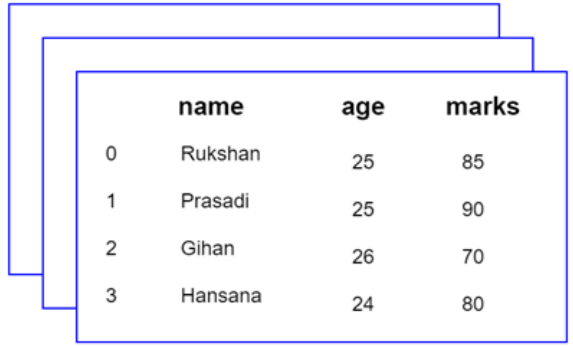
PANDAS

Pandas Data Types

- Series
- DataFrame
- Panel

Data Structure	Dimensions	Description
Series	1	1D labeled homogeneous array, sizeimmutable.
Data Frames	2	General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns.
Panel	3	General 3D labeled, size-mutable array.

PANDAS

Data Structure	Dimensionality	Format	View
Series	1D	Column	
DataFrame	2D	Single Sheet	
Panel	3D	Multiple Sheets	

PANDAS

Pandas Serileri

Pandas Serisi, NumPy dizi nesnelerinin üzerine inşa edilmiştir ve çok benzerler.

Herhangi bir veri tipinde veri tutabilen tek boyutlu etiketli bir dizidir.

Etiket değerlerine ise indeks denir.

Verinin kendisi sayılar, dizeler veya başka Python objelerinden oluşabilir.

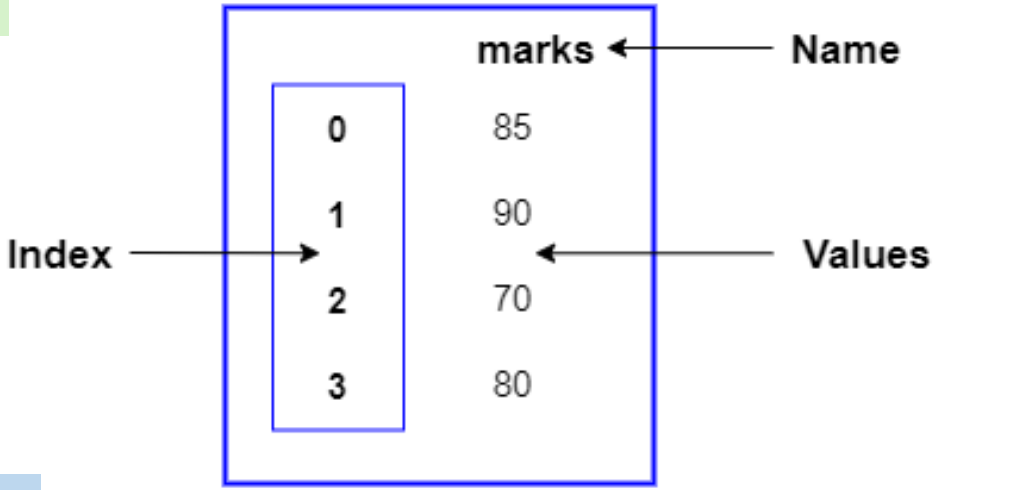
Serileri oluşturmak için ise listeler, sıralı diziler ya da sözlükler kullanılabilir.

PANDAS

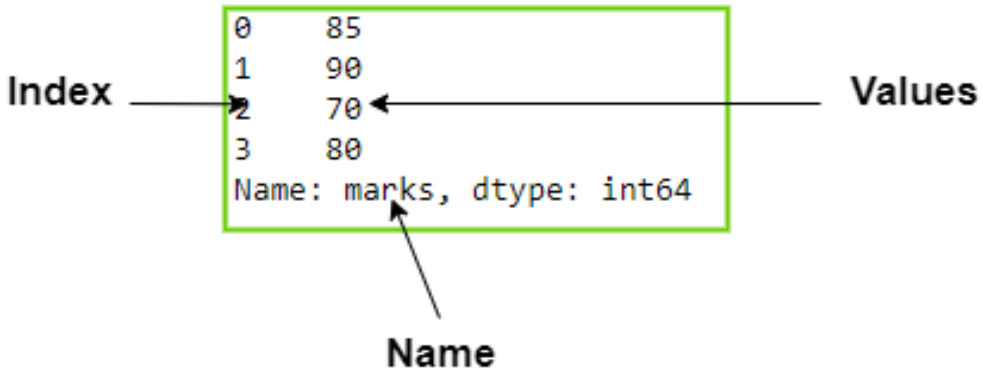
Series

- One-dimensional data (Values)
- Index

pandas.Series(data, index, dtype, copy)



pd.Series([85, 90, 70, 80], name='marks')



PANDAS

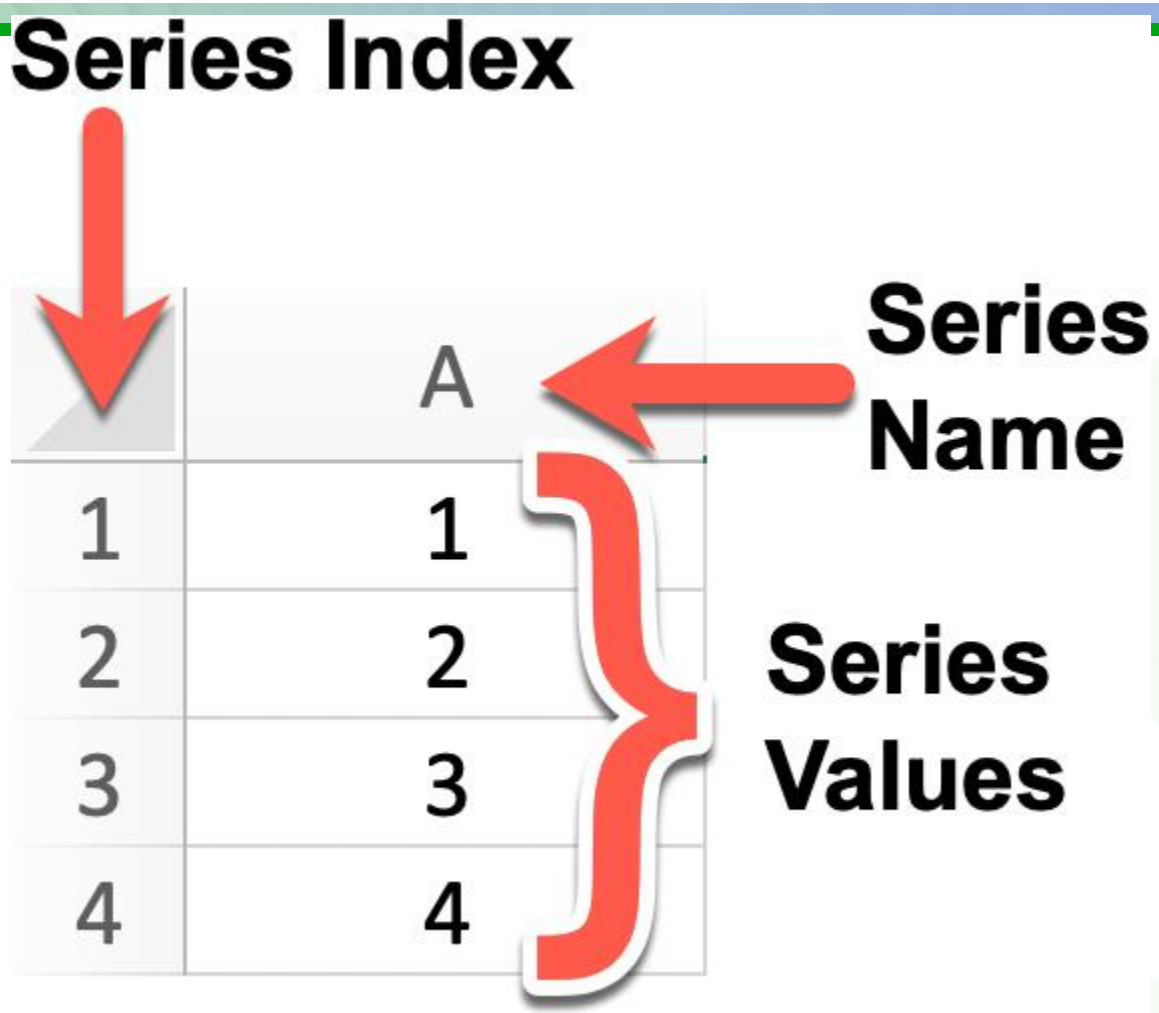
Pandas Series

10	23	56	17	52	61	73	90	26	72

```
s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])
```

```
a    0.469112
b   -0.282863
c   -1.509059
d   -1.135632
e    1.212112
dtype: float64
```

PANDAS



PANDAS

```
import pandas as pd
series1 = pd.Series([10,20,30])
print(series1)
```

giving an alias name to pandas

List

Series object

Output

index

```
0    10
1    20
2    30
dtype: int64
```

PANDAS

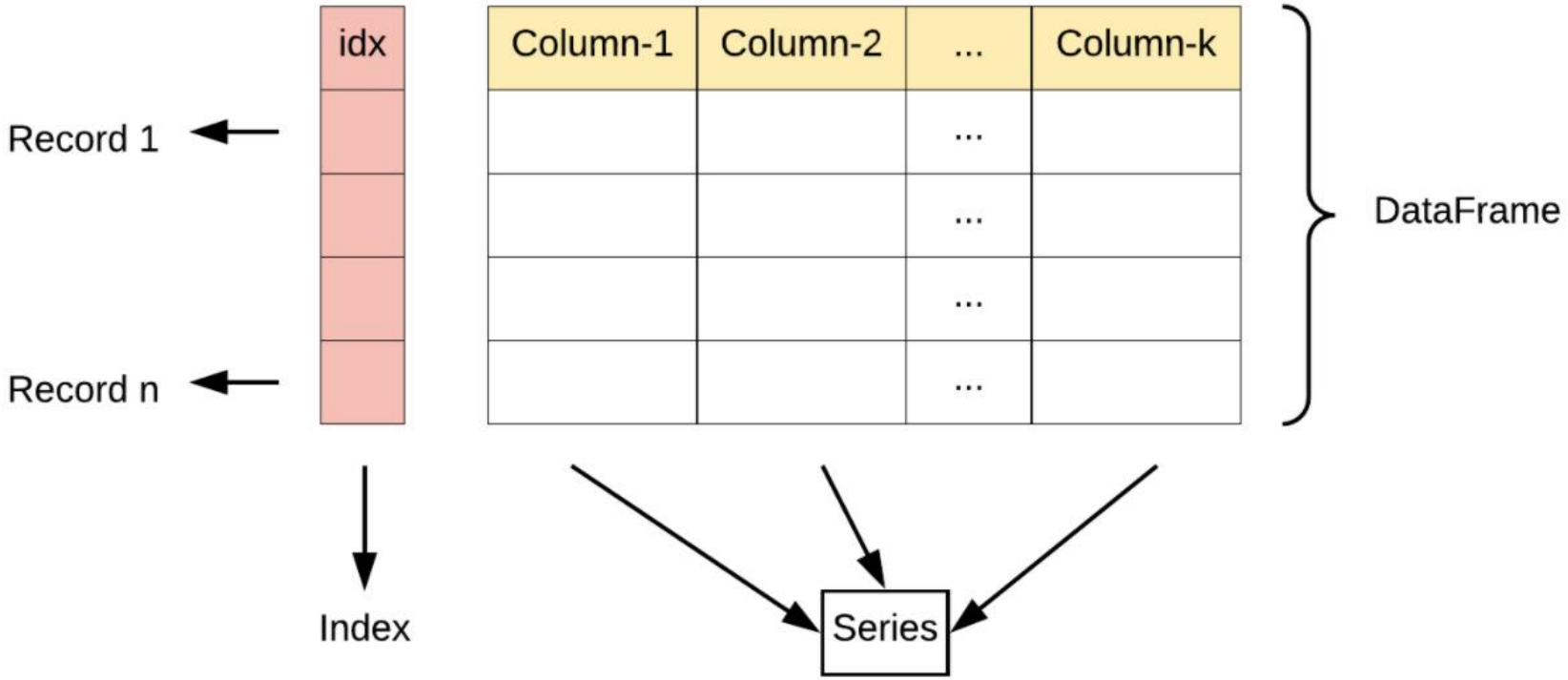
Pandas DataFrame

Pandas Dataframe, satırları ve sütunları olan iki boyutlu etiketli veri yapısıdır.

Pandas DataFrame'deki her sütun bir Pandas Serisidir.

Verinin kendisi sayılar, dizeler veya başka Python objelerinden oluşabilir.

Serileri oluşturmak için ise listeler, sıralı diziler ya da sözlükler kullanılabilir.



Series 1

	Mango
0	4
1	5
2	6
3	3
4	1

+

Series 2

	Apple
0	5
1	4
2	3
3	0
4	2

+

Series 3

	Banana
0	2
1	3
2	5
3	2
4	7

=

DataFrame

	Mango	Apple	Banana
0	4	5	2
1	5	4	3
2	6	3	5
3	3	0	2
4	1	2	7

PANDAS

Pandas DataFrame

Name	Age	Gender	Score
Ali	32	Male	3.45
Ayşe	28	Female	4.6
Hasan	45	Male	3.9
Fatma	38	Female	2.78

```
d = {
"one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]),
"two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]),
}
```

```
df = pd.DataFrame(d)
```

```
   one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0
```

DataFrame

- DataFrame is a two-dimensional array with heterogeneous data. For example,

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

Data Type of Columns

Column	Type
Name	String
Age	Integer
Gender	String
Rating	Float

PANDAS

Pandas Data Structures

Series

index	*values*
A	6
B	3.14
C	-4
D	0

DataFrame

index ⟸ *columns* ⟹

	foo	bar	baz
A	x	6	True
B	y	10	True
C	z	NaN	False

PANDAS

Series

	apples
0	3
1	2
2	0
3	1

Series

	oranges
0	0
1	3
2	7
3	2

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

PANDAS

Creating Series

```
import pandas as pd
s1 = pd.Series([1, 2, 3, 4])
```

```
s2 = pd.Series([1, 2, 3, 4], index=['A', 'B', 'C', 'D'])
```

0	1
1	2
2	3
3	4

A	1
B	2
C	3
D	4

PANDAS

Series Create

- A one-dimensional ndarray
- A Python list
- A Python dictionary
- A scalar value

	1	2
INPUT	Array=np.arange(1,10,2) Ser=pd.Series(array) Ser	data = np.array(['a','b','c','d']) s = pd.Series(data,index=[100,101,102,103]) print s
OUTPUT	0 1 1 3 2 5 3 7 4 9 Dtype: int32	100 a 101 b 102 c 103 d dtype: object

PANDAS

Using a Python dictionary and Scalar

	1	2	3
IN	data = {'a' : 0., 'b' : 1., 'c' : 2.} s = pd.Series(data) print s	data = {'a' : 0., 'b' : 1., 'c' : 2.} s = pd.Series(data,index=['b','c','d','a']) print s	s = pd.Series(5, index=[0, 1, 2, 3]) print s
OUT	a 0.0 b 1.0 c 2.0 dtype: float64	b 1.0 c 2.0 d NaN a 0.0 dtype: float64	0 5 1 5 2 5 3 5 dtype: int64

an index must be provided.

PANDAS

Series Indexing-Slicing

	1	2
IN	s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])	s[:3]
OUT	a 1 b 2 c 3 d 4 e 5 dtype: int64	a 1 b 2 c 3 dtype: int64

PANDAS

Retrieve Data Using Label (Index)

```
a  1
b  2
c  3
d  4
e  5
dtype: int64
```

	1	2
IN	s['a']	s[['a','c','d']]
OUT	1	a 1 c 3 d 4 dtype: int64

PANDAS

Accessing Element

```
ser
```

```
1st    1
2nd    3
3rd    5
4th    7
5th    9
dtype: int32
```

```
ser[[0, 2, 4]] # or ser[['1st', '3rd', '5th']]
```

```
1st    1
3rd    5
5th    9
dtype: int32
```

```
ser
```

```
1st    1
2nd    3
3rd    5
4th    7
5th    9
dtype: int64
```

```
ser[ser > 3]
```

```
3rd    5
4th    7
5th    9
dtype: int64
```

PANDAS

```
ser
```

```
1st    1
2nd    3
3rd    5
4th    7
5th    9
dtype: int64
```

```
ser[2:] = 100
ser
```

```
1st      1
2nd      3
3rd    100
4th    100
5th    100
dtype: int64
```

```
ser
```

```
1st    1
2nd    3
3rd    5
4th    7
5th    9
dtype: int64
```

```
ser + ser
```

```
1st     2
2nd     6
3rd    10
4th    14
5th    18
dtype: int64
```

```
ser / 2
```

```
1st    0.5
2nd    1.5
3rd    2.5
4th    3.5
5th    4.5
dtype: float64
```

```
ser
```

```
1st    1
2nd    3
3rd    5
4th    7
5th    9
dtype: int64
```

```
ser.mean()
```

```
5.0
```

```
np.mean(ser)
```

```
5.0
```

PANDAS

Series

```
>>> import pandas as pd
>>> s = pd.Series(list('abcdef'))
>>> s
0    a
1    b
2    c
3    d
4    e
5    f
>>> s = pd.Series([2, 4, 6, 8])
>>> s
0    2
1    4
2    6
3    8
```

PANDAS

Series- Indexing

```
>>> s = pd.Series([2, 4, 6, 8],
index = ['f', 'a', 'c', 'e'])
>>>
>>> s
f    2
a    4
c    6
e    8
>>> s['a']
4
>>> s[['a', 'c']]
a    4
c    6
```

PANDAS

Series- Indexing

```
>>> s2 = pd.Series(range(4),
index = list('abab'))
>>> s2
a    0
b    1
a    2
b    3
>>> s['a]
>>>
>>> s['a']
4
>>> s2['a']
a    0
a    2
>>> s2['a'][0]
0
```

PANDAS

Series- Operations

```
>>> s
f    2
a    4
c    6
e    8
>>> s[s > 4]
c    6
e    8
>>> s>4
f    False
a    False
c     True
e     True
>>> s*2
f    4
a    8
c    12
e    16
```

PANDAS

Series- Incomplete Data

```
>>> sdata = {'b':100, 'c':150, 'd':200}
>>> s = pd.Series(sdata)
>>> s
b     100
c     150
d     200
>>> s = pd.Series(sdata, list('abcd'))
>>> s
a     NaN
b     100
c     150
d     200
>>> s*2
a     NaN
b     200
c     300
d     400
```

PANDAS

Series- Automatic Alignment

```
>>> s2 = pd.Series([1, 2, 3],
index = ['c', 'b', 'a'])
>>> s2
c     1
b     2
a     3
>>> s
a     NaN
b     100
c     150
d     200
>>> s*s2
a     NaN
b     200
c     150
d     NaN
```

```
import pandas as pd
sr = pd.Series([15,20,40])
print(sr[2])
```

Positional index

Output

```
40
```

```
srCaps = pd.Series(['NewDelhi', 'WashingtonDC',
'London', 'Paris'], index=['India', 'USA', 'UK',
'France’])
print(srCaps['India’])
```

Label index

Output

```
'NewDelhi'
```

PANDAS

```
import pandas as pd
srCaps = pd.Series(['NewDelhi', 'WashingtonDC', 'London',
'Paris'],index=['India', 'USA', 'UK', 'France'])
print(srCaps[-1])
print(srCaps[1:3])
```

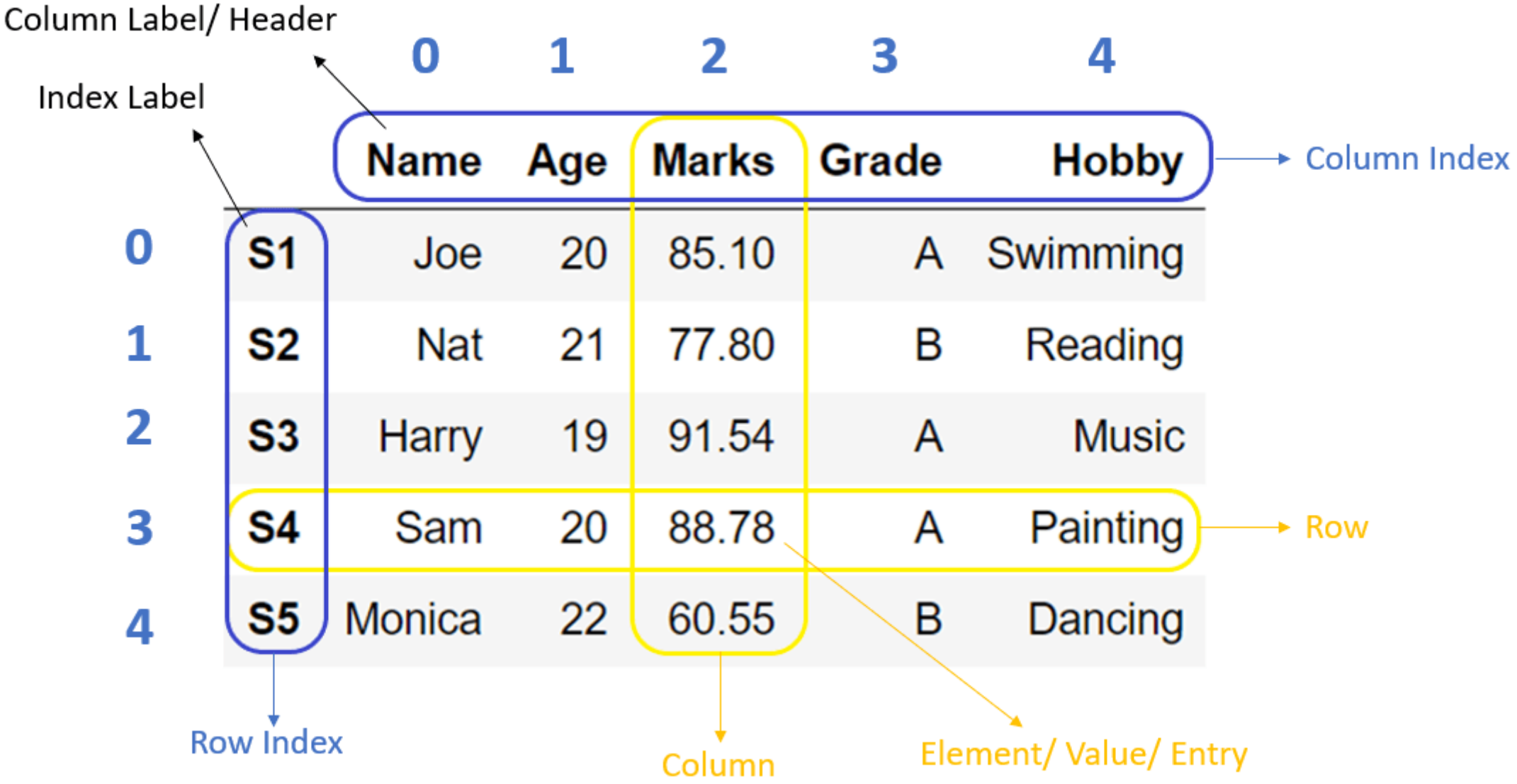
Negative index

Slicing

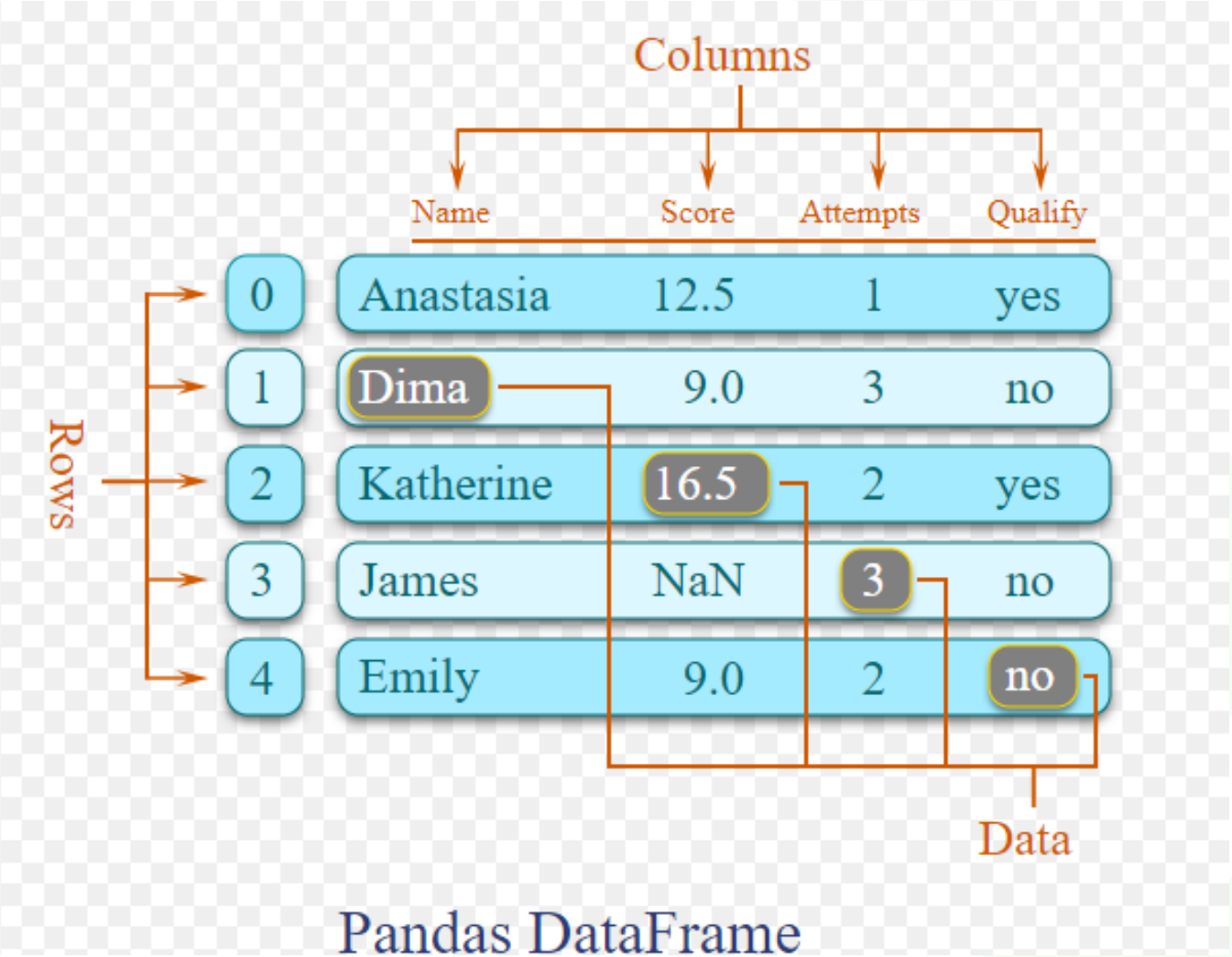
Output

```
Paris
USA    WashingtonDC
UK              London
dtype: object
```

PANDAS


Column Label/ Header
Index Label
0 1 2 3 4
Name Age Marks Grade Hobby
Column Index
0 S1 Joe 20 85.10 A Swimming
1 S2 Nat 21 77.80 B Reading
2 S3 Harry 19 91.54 A Music
3 S4 Sam 20 88.78 A Painting
4 S5 Monica 22 60.55 B Dancing
Row
Row Index
Column
Element/ Value/ Entry

PANDAS


Columns
Name
Score
Attempts
Qualify
Rows
0
Anastasia
12.5
1
yes
1
Dima
9.0
3
no
2
Katherine
16.5
2
yes
3
James
NaN
3
no
4
Emily
9.0
2
no
Data
Pandas DataFrame

PANDAS

```
df = pd.DataFrame ( { 'month' : [ 2, 5, 8, 10 ],
'year' : [ 2017, 2019, 2018, 2019 ],
'sale' : [ 60, 45, 90, 36 ] } )
```

DataFrame
df

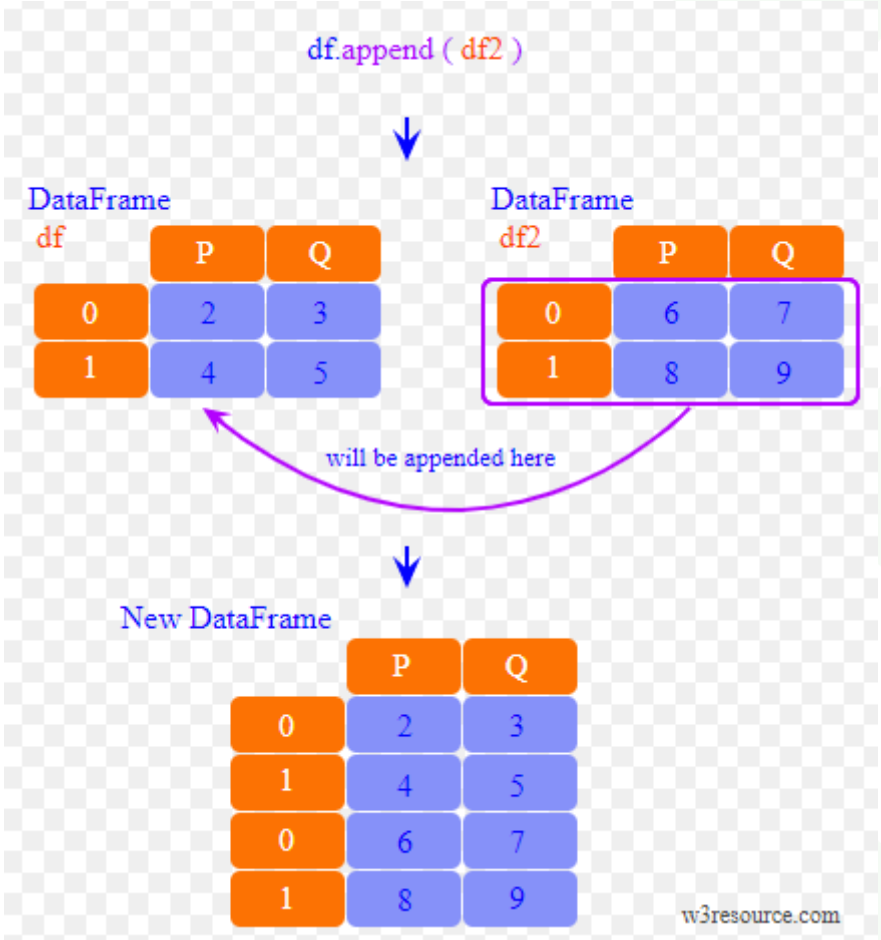
	month	year	sale
0	2	2017	60
1	5	2019	45
2	8	2018	90
3	10	2019	36

```
pd.DataFrame ( np.array ( ( [ 2, 3, 4 ], [ 5, 6, 7 ] ) ),
index = [ 'tiger', 'lion' ],
columns = [ 'one', 'two', 'three' ] )
```

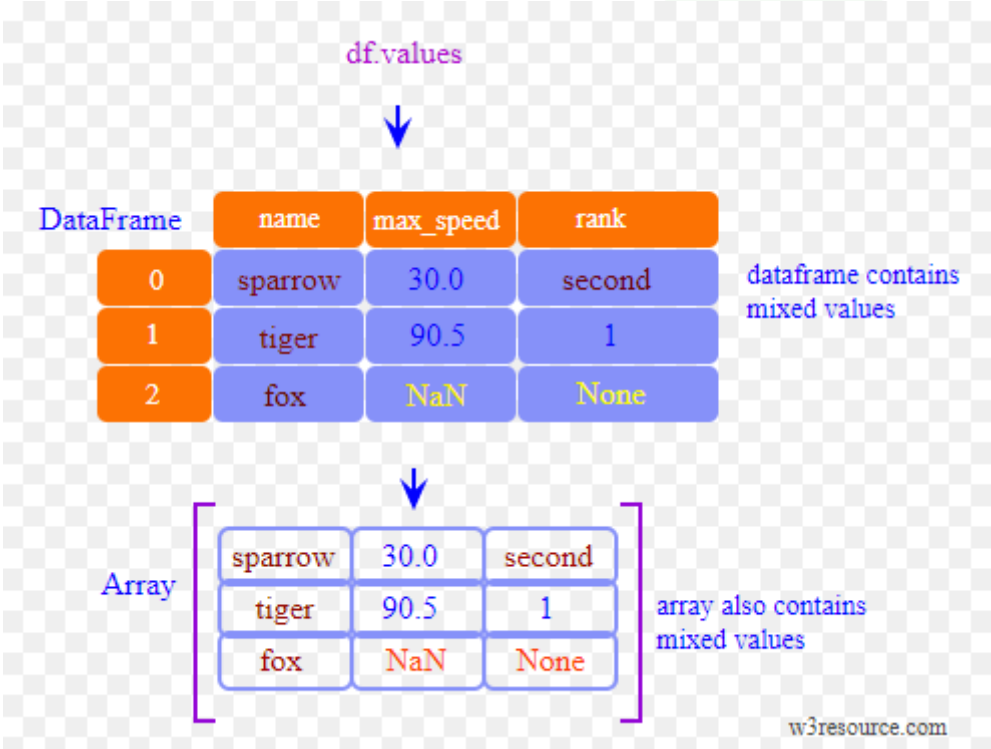
DataFrame

	one	two	three
tiger	2	3	4
lion	5	6	7

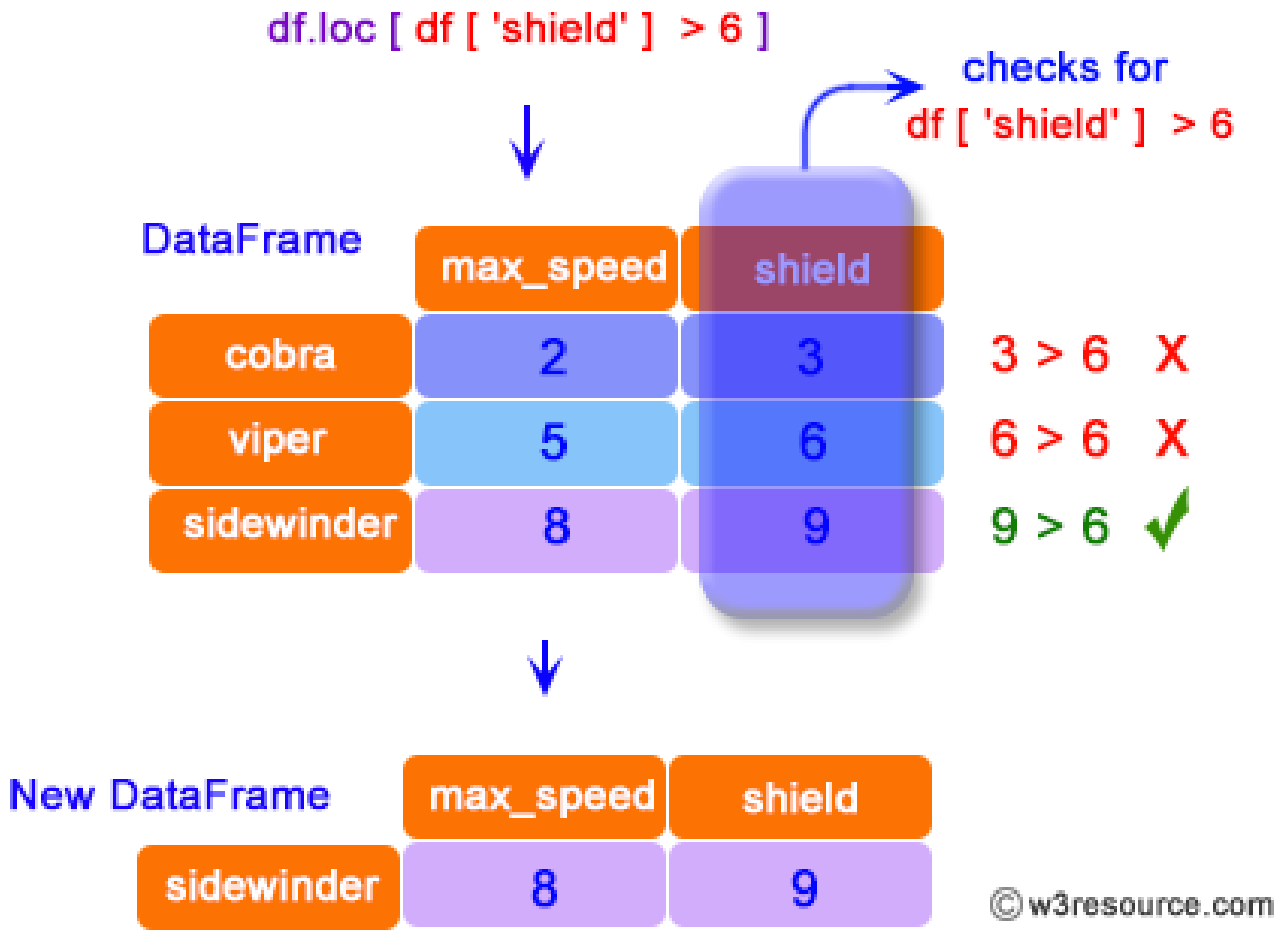
PANDAS


df.append (df2)
DataFrame
df
P
Q
0
2
3
1
4
5
DataFrame
df2
P
Q
0
6
7
1
8
9
will be appended here
New DataFrame
P
Q
0
2
3
1
4
5
0
6
7
1
8
9
w3resource.com

PANDAS


df.values
DataFrame
name
max_speed
rank
0
sparrow
30.0
second
1
tiger
90.5
1
2
fox
NaN
None
dataframe contains mixed values
Array
sparrow
30.0
second
tiger
90.5
1
fox
NaN
None
array also contains mixed values
w3resource.com

PANDAS


df.loc [df ['shield'] > 6]
checks for
df ['shield'] > 6
DataFrame
max_speed
shield
cobra
2
3
3 > 6 X
viper
5
6
6 > 6 X
sidewinder
8
9
9 > 6
New DataFrame
max_speed
shield
sidewinder
8
9
©w3resource.com

PANDAS

DataFrame

```
>>> data = {'state': ['FL', 'FL', 'GA', 'GA', 'GA'],
            'year':  [2010, 2011, 2008, 2010, 2011],
            'pop':   [18.8, 19.1, 9.7, 9.7, 9.8]}
>>> frame = pd.DataFrame(data)
>>> frame
    pop state  year
0  18.8    FL  2010
1  19.1    FL  2011
2   9.7    GA  2008
3   9.7    GA  2010
4   9.8    GA  2011
```

PANDAS

DataFrame

```
>>> pop_data = {'FL': {2010:18.8, 2011:19.1},
                'GA': {2008: 9.7, 2010: 9.7, 2011:9.8}}
>>> pop = pd.DataFrame(pop_data)
>>> pop
        FL   GA
2008   NaN  9.7
2010  18.8  9.7
2011  19.1  9.8
```

PANDAS

DataFrame

```
>>> obj = pd.Series(['blue', 'purple', 'red'],
index=[0,2,4])
>>> obj
0       blue
2     purple
4        red
>>> obj.reindex(range(4))
0       blue
1        NaN
2     purple
3        NaN
>>> obj.reindex(range(5), fill_value='black')
0       blue
1      black
2     purple
3      black
4        red
>>> obj.reindex(range(5), method='ffill')
0       blue
1       blue
2     purple
3     purple
4        red
```

PANDAS

DataFrame

```
>>> pop
      FL   GA
2008  NaN  9.7
2010 18.8  9.7
2011 19.1  9.8
>>> pop.sum()
FL    37.9
GA    29.2
>>> pop.mean()
FL    18.950000
GA     9.733333
>>> pop.describe()
              FL        GA
count   2.000000  3.000000
mean   18.950000  9.733333
std     0.212132  0.057735
min    18.800000  9.700000
25%    18.875000  9.700000
50%    18.950000  9.700000
75%    19.025000  9.750000
max    19.100000  9.800000
```

PANDAS

DataFrame

```
>>> pop
      FL   GA
2008  NaN  9.7
2010 18.8  9.7
2011 19.1  9.8
>>> pop < 9.8
       FL     GA
2008 False   True
2010 False   True
2011 False  False
>>> pop[pop < 9.8] = 0
>>> pop
      FL   GA
2008  NaN  0.0
2010 18.8  0.0
2011 19.1  9.8
```