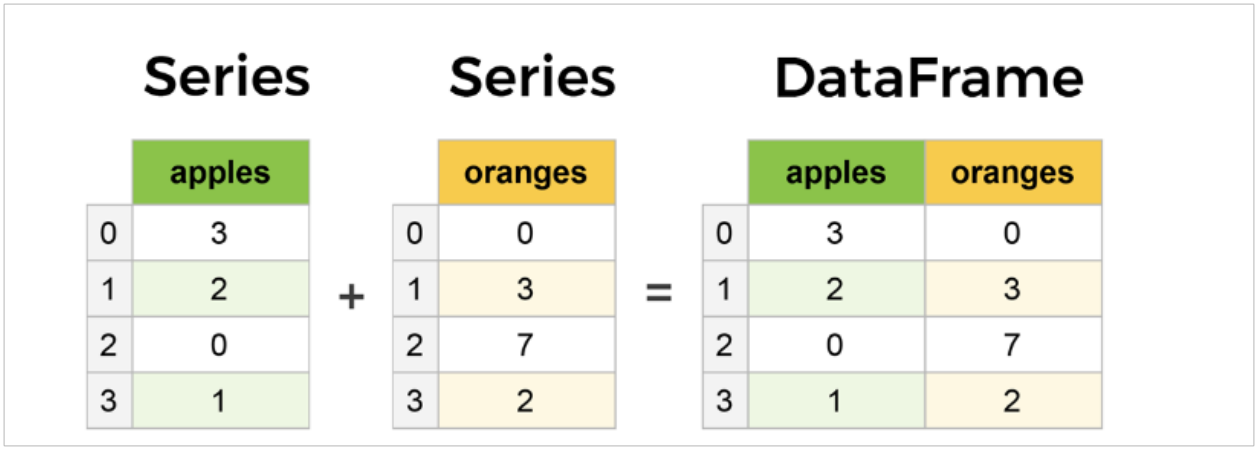
Introduction to Pandas

Pandas, veri manipülasyonu ve analizi için Python programlama dili için yazılmış ücretsiz bir yazılım kütüphanesidir. Özellikle, sayısal tabloları ve zaman serilerini işlemek için veri yapıları ve işlemleri sunar. Pandas esas olarak DataFrames şeklinde makine öğrenimi için kullanılır. Pandas, csv, excel vb. gibi çeşitli dosya formatlarındaki verilerin içe aktarılmasına izin verir. Ve groupby, join, merge, melt, concatenation gibi çeşitli veri manipülasyon işlemlerinin yanı sıra null değerleri doldurma, değiştirme veya imputing gibi veri temizleme özellikleri sağlar.

```
serr = pd.Series([11,21,13,41,15,61])
serr
```

```
0    11
1    21
2    13
3    41
4    15
5    61
dtype: int64
```

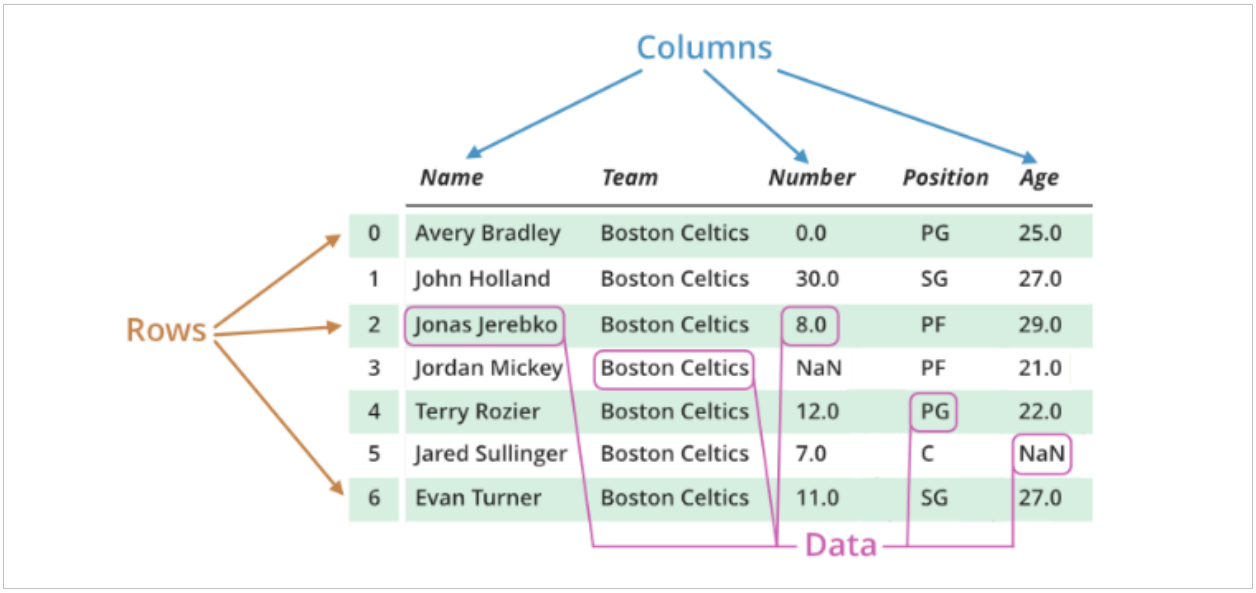
Pandas serisi, herhangi bir türdeki (tamsayı, dize, float, python nesneleri, vb.) verileri tutabilen tek boyutlu etiketli bir dizidir. DataFrames'i tam olarak anlamak için serilerin temellerini bilmeniz gerekir. Pandas serisini bir excel sayfasındaki etiketli bir sütun olarak düşünebilirsiniz.



Bir seride, eksen etiketleri indeks olarak adlandırılır. Seriler yalnızca indeksli tek bir liste içerebilirken, DataFrames birden fazla seriden oluşabilir.

Data Frame Basics-1 (Attributes)

DataFrame, verilerin satırlar ve sütunlar halinde hizalandığı iki boyutlu bir veri yapısıdır. Üç temel bileşen; veriler, satırlar ve sütunlar Pandas DataFrame'i oluşturur.



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Bu derste öğreneceksiniz:

DataFrame(): İki boyutlu, boyutu değiştirilebilir, potansiyel olarak heterojen tablo verileri. Veri yapısı ayrıca etiketli eksenler (satırlar ve sütunlar) içerir. Birincil pandas veri yapısı

Series(): Eksen etiketleri içeren tek boyutlu ndarray (zaman serileri dahil).

```
series =pd.Series(list("techproeducation"))
series
```

```
0     t
1     e
2     c
3     h
4     p
5     r
6     o
7     e
8     d
9     u
10    c
11    a
12    t
13    i
14    o
15    n
dtype: object
```

- **.sort_index :** Sort Series by index labels

```
series.sort_index(ascending=False)
```

```
15    n
14    o
13    i
12    t
11    a
10    c
9     u
8     d
7     e
6     o
5     r
4     p
3     h
2     c
1     e
0     t
dtype: object
```

- **.sort_values :** Sort a Series in ascending or descending order by the values

```
series.sort_values()
```

```
11    a
2     c
10    c
8     d
1     e
7     e
3     h
13    i
15    n
6     o
14    o
4     p
5     r
0     t
12    t
9     u
dtype: object
```

- **.isin :** Return a boolean Series showing whether each element in the Series matches an element in the passed sequence of *values* exactly

```
series.isin(["o"])
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6      True
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14     True
15    False
dtype: bool
```

- **.keys :** return the index labels of the given series object

```
series.keys()
```

```
RangeIndex(start=0, stop=16, step=1)
```

- **.values :** Return Series as ndarray or ndarray-like depending on the dtype

```
series.values
```

```
array(['t', 'e', 'c', 'h', 'p', 'r', 'o', 'e', 'd', 'u', 'c', 'a', 't',
       'i', 'o', 'n'], dtype=object)
```

- **.items :** This method returns an iterable tuple (index, value)

```
list(series.items())
```

```
[(0, 't'),
 (1, 'e'),
 (2, 'c'),
 (3, 'h'),
 (4, 'p'),
 (5, 'r'),
 (6, 'o'),
 (7, 'e'),
 (8, 'd'),
 (9, 'u'),
 (10, 'c'),
 (11, 'a'),
 (12, 't'),
 (13, 'i'),
 (14, 'o'),
 (15, 'n')]
```

- **.read_csv()**: Read a comma-separated values (csv) file into DataFrame

```
dataframe=pd.read_csv("csv name.csv")
```

- **.head()**: This function returns the first *n* rows for the object based on position, default n=5

```
df=pd.DataFrame(np.arange(1, 59, 3).reshape(5,4), columns = ["var1", "var2", "var3", "var4"])
df
```

	var1	var2	var3	var4
0	1	4	7	10
1	13	16	19	22
2	25	28	31	34
3	37	40	43	46
4	49	52	55	58

```
df.head(2)
```

	var1	var2	var3	var4
0	1	4	7	10
1	13	16	19	22

.**tail**(): This function returns last *n* rows from the object based on position, default n=5.

```
df.tail(2)
```

	var1	var2	var3	var4
3	37	40	43	46
4	49	52	55	58

- **.sample :** Return a random sample of items from an axis of object

```
df.sample(n=2)
```

	var1	var2	var3	var4
0	1	4	7	10
1	13	16	19	22

- .**shape**: Return a tuple representing the dimensionality of the DataFrame.

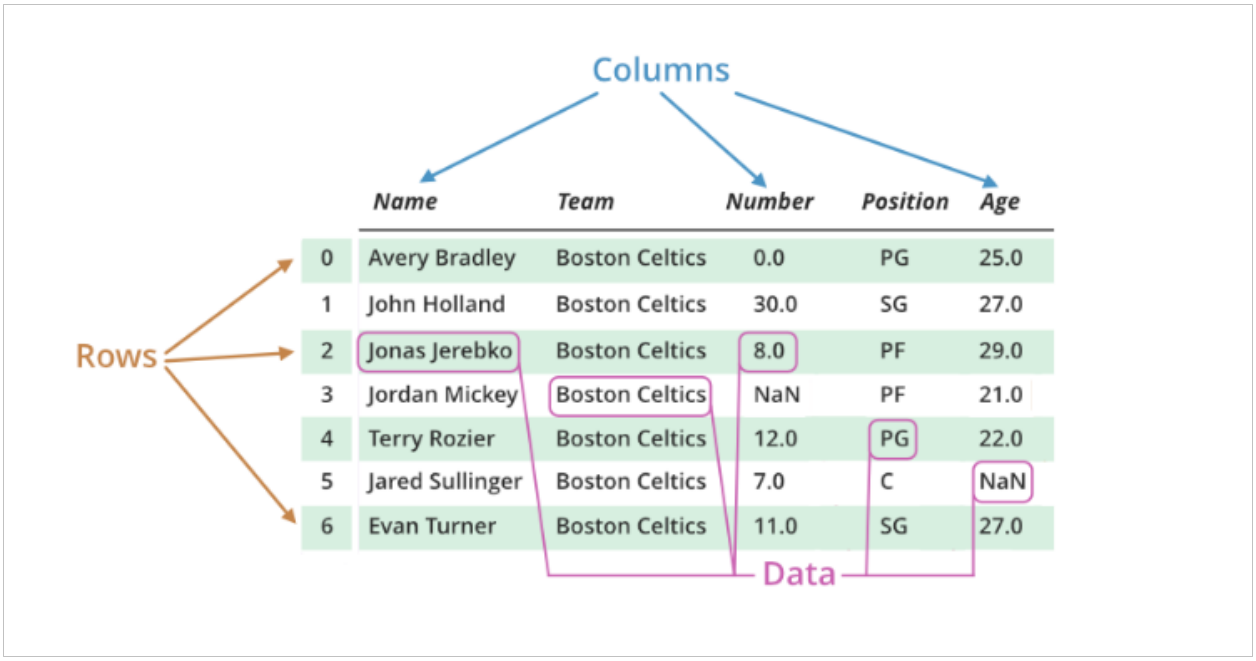
```
df.shape
```

```
(5, 4)
```

Data Frame Basics-2 (Index & Selecting)

Veri Çerçevesi Temelleri-2 (Dizin ve Seçme)

DataFrame, verilerin satırlar ve sütunlar halinde hizalandığı iki boyutlu bir veri yapısıdır. Üç temel bileşen; veriler, satırlar ve sütunlar Pandas DataFrame'i oluşturur.



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

```
np.random.seed(101)
df = pd.DataFrame(np.random.randint(9, size=(5, 4)),
                  index = 'A B C D E'.split(),
                  columns = 'W X Y Z'.split())
```

```
df
```

	W	X	Y	Z
A	1	6	7	8
B	4	8	5	0
C	5	8	1	3
D	8	3	3	2
E	8	3	7	0

- **df.index :** The basic object storing axis labels for all pandas objects.

```
df.index
```

```
Index(['A', 'B', 'C', 'D', 'E'], dtype='object')
```

- **df.columns :** The column labels of the DataFrame.

```
df.columns
```

```
Index(['W', 'X', 'Y', 'Z'], dtype='object')
```

- **df.shape :** Return a tuple representing the dimensionality of the DataFrame.

```
df.shape
```

```
(5, 4)
```

- **df.size :** Return an int representing the number of elements in this object.

```
df.size
```

```
20
```

- **df.ndim :** Return an int representing the number of axes / array dimensions.

```
df.ndim
```

```
2
```

- **df.reset_index :** Reset the index of the DataFrame, and use the default one instead. If the DataFrame has a MultiIndex, this method can remove one or more levels.

```
# Reset to default 0,1...n index
df.reset_index()
```

	index	W	X	Y	Z
0	A	1	6	7	8
1	B	4	8	5	0
2	C	5	8	1	3
3	D	8	3	3	2
4	E	8	3	7	0

- **df.set_index():** Set the DataFrame index using existing columns.

```
newindx='CA NY WY OR CO'.split()
newindx
```

```
['CA', 'NY', 'WY', 'OR', 'CO']
```

```
df['newidx']=newindx
```

```
df
```

	W	X	Y	Z	newidx
A	1	6	7	8	CA
B	4	8	5	0	NY
C	5	8	1	3	WY
D	8	3	3	2	OR
E	8	3	7	0	CO

```
df.set_index('newidx')
```

	W	X	Y	Z
newidx				
CA	1	6	7	8
NY	4	8	5	0
WY	5	8	1	3
OR	8	3	3	2
CO	8	3	7	0

- **df["col"]:** You can pass a list of columns to `[]` to select columns in that order. If a column is not contained in the DataFrame, an exception will be raised.

```
df['W']
```

```
A    1
B    4
C    5
D    8
E    8
Name: W, dtype: int32
```

- **df.iloc[]:** Purely integer-location based indexing for selection by position.

```
df.loc['B','W']
```

```
4
```

- **Conditional Indexing:**The condition inside the selection brackets `df[ df["Y"]<6 ]` checks for which rows the `Y` column has a value smaller than 6:

```
df[df.Y<6]
```

	W	X	Y	Z	newidx
B	4	8	5	0	NY
C	5	8	1	3	WY
D	8	3	3	2	OR

Data Frame Basics-3 (Properties)

Veri Çerçevesi Temelleri-3 (Özellikler)

```
import seaborn as sns
```

```
df=sns.load_dataset("penguins")
```

```
df.head(3)
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female

- df.info(): This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

df.info(): Bu yöntem, dizin türü ve sütunları, boş olmayan değerler ve bellek kullanımı dahil olmak üzere bir DataFrame hakkında bilgi yazdırır

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

df.describe(): NaN değerleri hariç olmak üzere, bir veri kümesinin dağılımının merkezi eğilimini, dağılımını ve şeklini özetleyenleri içeren tanımlayıcı istatistikler oluşturun

```
df.describe()
```

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
count	342.000000	342.000000	342.000000	342.000000
mean	43.921930	17.151170	200.915205	4201.754386
std	5.459584	1.974793	14.061714	801.954536
min	32.100000	13.100000	172.000000	2700.000000
25%	39.225000	15.600000	190.000000	3550.000000
50%	44.450000	17.300000	197.000000	4050.000000
75%	48.500000	18.700000	213.000000	4750.000000
max	59.600000	21.500000	231.000000	6300.000000

df.shape: DataFrame'in boyutluluğunu temsil eden bir tuple döndürür.

```
df.shape
```

```
(344, 7)
```

len(df): Bir dizenin uzunluğunu (karakter sayısını) verir. Sözlükler, listeler veya tuple'lar için girdi sayısını verir.

```
len(df)
```

```
344
```

df.col.value_counts(): Benzersiz değerlerin sayılarını içeren bir Seri döndürür.

```
df.species.value_counts()
```

```
Adelie       152
Gentoo       124
Chinstrap     68
Name: species, dtype: int64
```

df.mean(): İstenen eksen üzerindeki değerlerin ortalamasını döndürür.

```
df.mean()
```

```
bill_length_mm          43.921930
bill_depth_mm           17.151170
flipper_length_mm      200.915205
body_mass_g           4201.754386
dtype: float64
```

df.col.sum(): İstenen eksen üzerindeki değerlerin toplamını döndürür.

```
df.bill_depth_mm.sum()
```

```
5865.700000000001
```

df.col.unique(): Karma tablo tabanlı benzersiz. Benzersizler görünüm sırasına göre döndürülür. Bu sıralama yapmaz

```
df.species.unique()
```

```
array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)
```

df.isnull().sum(): Eksik değerleri tespit edin. Her sütundaki eksik değerlerin toplam sayısını döndürür.

```
df.isnull().sum()
```

```
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
dtype: int64
```

df.drop():Etiket adlarını ve ilgili ekseni belirterek veya doğrudan dizin veya sütun adlarını belirterek satırları veya sütunları kaldırın

```
df.drop("bill_length_mm", axis=1)
```

	species	island	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	16.1	213.0	5400.0	Male

344 rows × 6 columns

```
df = pd.DataFrame(np.arange(12).reshape(3, 4), columns=['A', 'B', 'C', 'D'])
df
```

	A	B	C	D
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

```
df.drop(['B', 'C'], axis=1)
```

	A	D
0	0	3
1	4	7
2	8	11