

Python ile Veri Analizi

[Gösterge Paneli](#) / [Dersler](#) / [Çeşitli](#) / [Python ile DA](#) / [pandalar](#) / [Pandalar Bölüm-2](#)

Pandalar Bölüm-2

Yapılacaklar: Aktivitenin sonuna kadar gidin

Metin ve Zaman Yöntemleri

Birçok veri parçası, salt sayılar yerine metin biçimindedir. Bu, dizinin incelenmeden, algoritmalar tarafından işlenmeden veya halka gösterilmeden önce temizlenmesi ve ön işleme tabi tutulması gerektiği anlamına gelir. Neyse ki pandas kitaplığında, dize verileriyle çalışmayı kolaylaştıran dize işlemeye ayrılmış bir bölüm vardır.

Diziler ve Dizinler, dizinin her bir ögesi üzerinde çalışmayı kolaylaştıran bir dizi dizi işleme yöntemiyle donatılmıştır. Belki de en önemlisi, bu yöntemler eksik/NA değerlerini otomatik olarak hariç tutar. **str** Bunlara öznitelik aracılığıyla erişilir .

```
df=sns.load_dataset('titanic')
df.head(3)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True

- lower()**: Dizeleri küçük harfe dönüştürür

```
df.embark_town =df.embark_town.str.lower()
df.embark_town.head(3)
```

```
0    southampton
1    cherbourg
2    southampton
Name: embark_town, dtype: object
```

- upper()** : Dizeleri büyük harfe dönüştürür.

```
df.sex=df.sex.str.upper()
df.sex.head(3)
```

```
0    MALE
1    FEMALE
2    FEMALE
Name: sex, dtype: object
```

- islower()** : Her dizedeki tüm karakterlerin küçük harf olup olmadığını kontrol eder. Boole değerini döndürür

```
df.sex.head(2).str.islower()
```

```
0    True
1    True
Name: sex, dtype: bool
```

- **isupper()** : Her dizedeki tüm karakterlerin büyük harf olup olmadığını kontrol eder. Boole değerini döndürür

```
df.sex.head(2).str.isupper()
```

```
0    False
1    False
Name: sex, dtype: bool
```

- **isdigit()** : Check whether all characters in each string are digits.

```
df.pclass.astype('string').str.isdigit().head(2)
```

```
0    True
1    True
Name: pclass, dtype: boolean
```

- **isnumeric()**: Checks whether all characters in each string are numeric. Returns Boolean.

```
df.pclass.astype('string').str.isnumeric().head(2)
```

```
0    True
1    True
Name: pclass, dtype: boolean
```

- **replace()**: Replaces the value **a** with the value **b**

```
df['age'] = df['age'].replace(np.nan, 'UNKNOWN')
```

- **contains()**: Returns a Boolean value True for each element if the substring contains in the element, else False.

```
df[df.sibsp.astype('string').str.contains('1')].head(2)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alc
0	0	3	MALE	22.0	1	0	7.2500	S	Third	man	True	NaN	southampton	no	Fa
1	1	1	FEMALE	38.0	1	0	71.2833	C	First	woman	False	C	cherbourg	yes	Fa

- **split()**: Splits each string with the given pattern

```
df.sex.str.split('MALE').head(2)
```

```
0    [, ]
1    [FE, ]
Name: sex, dtype: object
```

- **strip()**: Helps strip whitespace(including newline) from each string

```
df.sex.str.strip('LE').head(2)
```

```
0    MA
1  FEMA
Name: sex, dtype: object
```

- **find()** : Returns the first position of the first occurrence of the pattern

```
df.embark_town.str.find('southampton').head(2)
```

```
0    0.0
1   -1.0
Name: embark_town, dtype: float64
```

- **findall()** : Returns a list of all occurrence of the pattern.

```
df.embark_town.str.findall('southampton').head(2)
```

```
0    [southampton]
1                []
Name: embark_town, dtype: object
```

Bu soru buraya gömülemez.

How do I use string methods in pandas?



Time-series data is currently used in a wide range of industries for time series forecasting, seasonality analysis, trend detection, and making critical business and research choices. As a result, it is critical for a data scientist or data analyst to accurately understand time series data.

- **datetime** modül, Python'da zaman serisi verileriyle çalışmak için temel nesneleri içerir.

```
df=sns.load_dataset('flights')
df.head(3)
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132

- **to_datetime()** **Timestamp** yöntem, bir nesneyi döndüren birçok farklı türde tarih temsilini ayrıştırır .

```
df['year']=pd.to_datetime(df['year'],format='%Y' )
df.year.head(3)
```

```
0    1949-01-01
1    1949-01-01
2    1949-01-01
Name: year, dtype: datetime64[ns]
```

- **strftime** - nesneyi belirli bir formata göre bir dizgeye dönüştürün

```
from datetime import datetime
current_date=datetime.now()
current_date
```

```
datetime.datetime(2022, 4, 8, 14, 50, 37, 470978)
```

```
date=current_date.strftime('%d'+' '+'%b'+' '+'%Y'))
date
```

```
'08 Apr 2022'
```

- **strptime** - bir dizgiyi **datetime** karşılık gelen bir formatta verilen bir nesneye ayrıştırın

```
datetime.strptime(date, '%d %b %Y')
```

```
datetime.datetime(2022, 4, 8, 0, 0)
```

- **timedelta**- zaman farkı verir

```
from datetime import timedelta
two_days_before=current_date.now()-timedelta(days=2)
two_days_before
```

```
datetime.datetime(2022, 4, 6, 14, 52, 13, 368791)
```

Bu soru buraya gömülemez.