

11

Tarih ve Zaman

Birçok alanda olduğu gibi veri biliminde de tarih ve zaman verileri çok fazla kullanılır. Bu nedenle birçok programlama diliinde tarih ve zamanlarla daha kolay işlem yapabilmek için özel sınıflar oluşturulmuştur. Tarih ve zaman sınıflarını iyi öğrenmek özellikle zaman serisi analizlerinde çok işinize yarayacaktır.

11.1. Datetime Modülü

Python'da tarih ve zaman işlemleri yapmanın farklı yöntemleri vardır. Bunlardan ilki **datetime** modülünü ve bu modülde yer alan **datetime** veri türünü kullanmaktadır.

Datetime modülünü kullanarak metin formatındaki bir veriyi tarih-zaman nesnesine dönüştürebiliriz. Bunun için **.strptime()** metodu kullanılır.

```
from datetime import datetime as dt

tarih = '08-12-2017'

tarih_dt = dt.strptime(tarih, '%d-%m-%Y')

print(type(tarih_dt))
<class 'datetime.datetime'>

print(tarih_dt)
2017-12-08 00:00:00
```

PYTHON ile VERİ BİLİMİ

Tarih nesnesine dönüştürülecek metni `dt.strptime()` metodunu içinde ilk argüman olarak yazıyoruz. İkinci argüman olarak dönüştürülecek metnin hangi formatta olduğunu yazmamızıza program, gün, tarih ve yıl nereden arayacağını, bunların türünü ve bunlar arasında hangi işaretin olduğunu (/, -, . vb.) anlasın. Yukarıdaki kodda '%d-%m-%Y' ifadesini görüyorsunuz. %d, işaret 01,02,03,...,31 şeklinde ayın günlerini gösterir. %m, 01,02,...,12 şeklinde ayları, %Y ise 2016, 2017 formatında yılları gösterir. Tarih ve zaman format kısaltmaları C dilinden aktarılmış olup çoğu dilde ortak olarak kullanılmaktadır. Buradaki yazım sırası ve format dönüştürülecek metinle aynı olmalıdır.

Bir datetime nesnesini metin verisine çevirmek içinse `.strftime()` metodunu kullanılır. Şimdi bu metodunu kullanarak yukarıda yaptığımız işlemi tersine çevirelim. Bu defa tarih formattını farklı biçimde yazalım.

```
tarih_metin = dt.strftime(tarih_dt, '%Y.%m.%d')
print(tarih_metin)
```

2017.12.08

Şimdi de bir tarih listesindeki verilerin hepsini, for döngüsü kullanarak datetime nesnesine dönüştürelim. Fazla yer kapsamaması için listeyi kısa tuttum ama liste ne kadar uzun olursa olsun yöntem değişmeyecektir.

```
tarih_listesi = ['01.01.2017',
                  '02.01.2017',
                  '03.01.2017',
                  '29.12.2017',
                  '30.12.2017',
                  '31.12.2017']
```

```
from datetime import datetime as dt

for tarih in tarih_listesi:
    tarih_dt = dt.strptime(tarih, '%d.%m.%Y')
    print(tarih_dt)

2017-01-01 00:00:00
2017-01-02 00:00:00
2017-01-03 00:00:00
2017-12-29 00:00:00
2017-12-30 00:00:00
2017-12-31 00:00:00
```

Datetime formatına çevirdiğimiz verileri yazdırmak yerine bir liste olarak saklarsak ne olur?

TARİH ve ZAMAN

```
tarih_dt = []
for tarih in tarih_listesi:
    tarih_dt.append(dt.strptime(tarih, '%d.%m.%Y'))

print(tarih_dt)
[datetime.datetime(2017, 1, 1, 0, 0),
 datetime.datetime(2017, 1, 2, 0, 0),
 datetime.datetime(2017, 1, 3, 0, 0),
 datetime.datetime(2017, 12, 29, 0, 0),
 datetime.datetime(2017, 12, 30, 0, 0),
 datetime.datetime(2017, 12, 31, 0, 0)]

benzer şekilde, tarih_dt listesindeki datetime nesnelerini tekrar metin formatına çevirilebilir.

tarih_liste = []

for tarih in tarih_dt:
    tarih_liste.append(dt.strftime(tarih, '%d-%m-%Y'))

print(tarih_liste)
['01-01-2017',
 '02-01-2017',
 '03-01-2017',
 '29-12-2017',
 '30-12-2017',
 '31-12-2017']
```

Tarih ve zaman bilgisinin gösterilmesi konusunda geliştirilmiş bir standart ISO-8601 standardıdır. Bu standart göre tarihler YYYY-AA-GG şeklinde yıl, ay ve gün olarak yazılmıştır. Datetime modülünde yer alan `.isoformat()` metodunu kullanılarak datetime nesnelerini ISO-8601 formatında yazdırma mümkündür.

```
for tarih in tarih_dt:
    print(tarih.isoformat())
2017-01-01T00:00:00
2017-01-02T00:00:00
2017-01-03T00:00:00
2017-12-29T00:00:00
2017-12-30T00:00:00
2017-12-31T00:00:00
```

Göründüğü gibi datetime nesneleri gün, ay ve yıl bilgisinin yanında saniye, dakika saat ve saat dilimi bilgisi de içerir.

PYTHON İLE VERİ BİLİMİ

Datetime modülünde güncel zamanı görmeyen (ya da kullanmanın) yolu `.now()` ve `.utcnow()` metodlarını kullanmaktadır. Bunlardan ilki zamanı yerel saat diliminde, ikincisi ise UTC yani eşgüdümü evrensel zaman diliminde verir. UTC birçok ülkede baz alınan bilimsel zaman olarak kabul edilir.

```
yerel_zaman = dt.now()
utc_zaman = dt.utcnow()

print(yerel_zaman)
print(utc_zaman)
2017-12-08 22:01:40.343844
2017-12-08 19:01:40.343844
```

Zaman verileri ile çalışırken önemli olabilecek bir başka özellik de zaman dilimi (time zone) özellığıdır. Saat dilimlerini kullanılmak için pytz modülünün altında `timezone` fonksiyonunu kullanmamız gereklidir. Aşağıdaki örnekte görüldüğü gibi bir datetime nesnesine, `timezone` fonksiyonu ile saat dilimi bilgisi eklemek mümkündür. Saat dilimi özelliğinin, Dünyanın farklı bölgelerinden (örneğin farklı borsalarдан) çekilen verileri incelerken işinize yarayabilir.

```
from pytz import timezone

zaman = dt.strptime('18/07/2006 11:00', '%d/%m/%Y %H:%M')

TR_zaman = zaman.replace(tzinfo=timezone('Europe/Istanbul'))
IT_zaman = zaman.replace(tzinfo=timezone('Europe/Rome'))

print(TR_zaman)
print(IT_zaman)
2006-07-18 11:00:00+01:56
2006-07-18 11:00:00+00:56
```

Datetime modülünde bulunan `timedelta()` fonksiyonunu kullanarak bir tarihveye zamanlı belirli bir zaman (gün, ay, yıl, hafta vb.) sonra veya önceki tarih veya zamanı bulabiliyoruz.

```
from datetime import timedelta
from datetime import datetime as dt

bugun = dt.now()

fark = timedelta(days=100)
```

TARİH VE ZAMAN

```
gecmis = bugun - fark
gelecek = bugun + fark

print('Geçmiş: ' + str(gecmis) + '\nGelecek: ' + str(gelecek))
Geçmiş: 2017-09-05 22:55:05.791784
Gelecek: 2018-03-24 22:55:05.791784
Benzer şekilde datetime formatındaki iki tarih arasındaki farkı bulmak da mümkündür.

print(gelecek - gecmis)
200 days, 0:00:00
```

11.2. Pendulum Modülü

Tarih ve zaman işlemleri için kullanılabilen bir başka modül de pendulum modülüdür. Bu modülü kullanarak metin veri tipini tarih-zaman nesnesine çevirmek için `.parse()` fonksiyonu kullanılır. Önceki bölümde kullandığımız tarih listesini tekrar kullanarak liste elemanlarını pendulum tarih-zaman nesnesine çevirelim.

```
tarih_listesi = ['01.01.2017',
                  '02.01.2017',
                  '03.01.2017',
                  '29.12.2017',
                  '30.12.2017',
                  '31.12.2017']

import pendulum

for tarih in tarih_listesi:
    tar = pendulum.parse(tarih, tz ="Europe/Istanbul")
    print(tar)
2017-01-01T00:00:00+03:00
2017-02-01T00:00:00+03:00
2017-03-01T00:00:00+03:00
2017-12-29T00:00:00+03:00
2017-12-30T00:00:00+03:00
2017-12-31T00:00:00+03:00
```

Datetime modülünde olduğu gibi pendulum modülünde de saat dilimi için kullanılan bir metod vardır: `.in_timezone()`. Bu metod ile bir tarih-zaman nesni başka bir saat dilimine çevirebiliriz.

```
import pendulum
```

PYTHON İLE VERİ BİLİMI

```
pendulum_liste = []

for tarih in tarih_listesi:
    tar = pendulum.parse(tarih, tz ="Europe/Istanbul")
    pendulum_liste.append(tar)

print(pendulum_liste)

[<Pendulum [2017-01-01T00:00:00+03:00]>,
 <Pendulum [2017-02-01T00:00:00+03:00]>,
 <Pendulum [2017-03-01T00:00:00+03:00]>,
 <Pendulum [2017-12-29T00:00:00+03:00]>,
 <Pendulum [2017-12-30T00:00:00+03:00]>,
 <Pendulum [2017-12-31T00:00:00+03:00]>]

for pend_tarih in pendulum_liste:
    print(pend_tarih.in_timezone('Europe/Paris'))

2016-12-31T22:00:00+01:00
2017-01-31T22:00:00+01:00
2017-02-28T22:00:00+01:00
2017-12-28T22:00:00+01:00
2017-12-29T22:00:00+01:00
2017-12-30T22:00:00+01:00
```

Benzer şekilde herhangi bir saat diliminde şu andaki tarih ve zamanı görmek için `.now()` metodu ve argüman olarak da istenen saat dilimi kullanılabilir.

```
print(pendulum.now('UTC'))
2017-12-14T21:05:48.918480+00:00

print(pendulum.now('Asia/Tokyo'))
2017-12-15T06:05:48.918480+09:00
```

Datetime modülünde iki zaman arasındaki farkı alabileceğimizi görmüştük. Pendulum modülünde iki tarih veya zaman arasındaki fark `.in_hours()`, `.in_days()` gibi metodlar kullanılarak istenilen zaman diliminde alınabilir.

```
tarih1 = pendulum_liste[0]
tarih2 = pendulum_liste[4]
print(tarih1)
```

206

```
2017-01-01T00:00:00+03:00
print(tarih2)
2017-12-30T00:00:00+03:00
fark = tarih2 - tarih1

print(fark.in_hours())
8712
print(fark.in_days())
363
print(fark.in_weeks())
51
print(fark.in_months())
11
```

Zaman farkını düzgün bir şekilde almak ve yazdırma için pendulum modülünde `in_words()` metodu bulunmaktadır ancak bu metod sadece İngilizce sonuç vermektedir.

```
print(fark.in_words())
11 months 4 weeks 1 day
```

TARİH VE ZAMAN

11.3. Pandas Modülünde Tarih ve Zaman

Pandas modülü birçok özelliğin yanı sıra zaman serileri ile çalışırken kolaylık sağlayacak özellikler sunmaktadır. Pandas modülünde de tarih zaman bilgilerini ISO 8601 formatında okutmak veya tanımlamak mümkündür.

Pandas modülünde metin veri tipini tarih zaman veri tipine (datetime object) çevirmek için `.to_datetime()` metodunu kullanılır.

```
gunler_2018 = pd.to_datetime(['2018-01-01', '2018-01-02', '2018-01-03',
                               '2018-01-04', '2018-01-05', '2018-01-06', '2018-01-07', '2018-01-08',
                               '2018-01-09', '2018-01-10'])

gunler_2018
DatetimeIndex(['2018-01-01', '2018-01-02', '2018-01-03', '2018-01-04', '2018-01-05',
               '2018-01-06', '2018-01-07', '2018-01-08', '2018-01-09', '2018-01-10'],
              dtype='datetime64[ns]', freq=None)
```

Şimdi örnek olarak <https://finance.yahoo.com> adresinden Tesla firmasının fiyat verilerini indirip grafiğini çizdirelim. Yahoo Finance sayfasından herhangi bir finansal enstrüman

207

PYTHON ile VERİ BİLİMİ

can verisini indirmek için yukarıdaki adrese gidip arama kısmına istediğiniz finansal aracın adını ya da sembolünü yazmanız yeterli. Açılan sayfada, yukarıda yer alan Historical Prices linkine gidip Download Data linkini kullanarak verileri bilgisayarınıza indirebilirsiniz. Şimdi bilgisayarımıza indirdiğimiz csv formatındaki dosyayı okuyarak grafiğini çizdirelim.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2013-01-22	34.56	35.54	34.26	35.18	35.18	1920200
1	2013-01-23	35.02	36.24	34.96	36.00	36.00	1564300
2	2013-01-24	36.00	37.72	35.84	36.99	36.99	1970400
3	2013-01-25	37.00	37.54	36.79	36.98	36.98	1287800
4	2013-01-28	36.86	38.71	36.86	38.02	38.02	1986000

Gördüğü gibi indirdiğimiz TESLA hisse senedinin bilgileri Date (tarih), Open (açılış), High (gün içi en yüksek), Low (gün içi en düşük), Close (kapanış), Adj Close (düzeltilmiş fiyat) ve Volume (Hacim) bilgilerinden oluşmaktadır.

Pandas modülünde, yukarıdaki gibi zaman serilerinde çalışırken tarih sütununu indeks olarak belirtebiliriz. Bu bize analiz ve çizimlerde oldukça kolaylık sağlayacaktır.

```
tesla.index = tesla['Date']
tesla.index.name = 'Date'
tesla.head(3)
```

	Date	Open	High	Low	Close	Adj Close	Volume
	2013-01-22	2013-01-22	34.56	35.54	34.26	35.18	1920200
	2013-01-23	2013-01-23	35.02	36.24	34.96	36.00	1564300
	2013-01-24	2013-01-24	36.00	37.72	35.84	36.99	1970400

Şimdi tarih (Date) sütunu hem veri çerçevesinin indeksi hem de bir sütun olarak tekrar ediyor. Bu nedenle sütunu atabiliyoruz.

```
tesla.drop('Date', axis = 1, inplace=True)
tesla.head(3)
```

TARİH ve ZAMAN

Date	Open	High	Low	Close	Adj Close	Volume
2013-01-22	34.56	35.54	34.26	35.18	35.18	1920200
2013-01-23	35.02	36.24	34.96	36.00	36.00	1564300
2013-01-24	36.00	37.72	35.84	36.99	36.99	1970400

Aşında bazı argümanları kullanarak tarih bilgisini daha dosyayı okurken indeks olarak belirtebilirdik. Aşağıdaki komut, buraya kadar yaptıklarımızın aynısını yapacaktır:

```
tesla = pd.read_csv("C:/Users/.../TSLA.csv", index_col='Date',
parse_dates=True)
```

```
tesla.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
DatetimeIndex: 1259 entries, 2013-01-22 to 2018-01-19
```

```
Data columns (total 6 columns):
```

```
Open    1259 non-null float64
```

```
High    1259 non-null float64
```

```
Low    1259 non-null float64
```

```
Close   1259 non-null float64
```

```
Adj Close 1259 non-null float64
```

```
Volume  1259 non-null int64
```

```
dtypes: float64(5), int64(1)
```

```
memory usage: 68.9 KB
```

```
tesla.index
```

```
DatetimeIndex(['2013-01-22', '2013-01-23', '2013-01-24', '2013-01-25', '2013-01-28', '2013-01-29', '2013-01-30', '2013-01-31', '2013-02-01', '2013-02-04', ..., '2018-01-05', '2018-01-08', '2018-01-09', '2018-01-10', '2018-01-11', '2018-01-12', '2018-01-16', '2018-01-17', '2018-01-18', '2018-01-19'], dtype='datetime64[ns]', name='Date', length=1259, freq=None)
```

İstediğimiz bir tarihe ilişkin bilgileri görelim:

```
tesla.loc['2016-07-18']
```

```
Open      2.196400e+02
```

```
High      2.270900e+02
```

```
Low      2.183000e+02
```

```
Close     2.262500e+02
```

```
Adj Close 2.262500e+02
```

```
Volume   3.412100e+06
```

```
Name: 2016-07-18 00:00:00, dtype: float64
```

PYTHON İLE VERİ BİLİMI

Yukarıdaki gibi 2016-07-18 yerine istediğimiz tarihi aşağıdaki şekillerde de yazabilirim.

```
tesla.loc['July 18, 2016']
```

```
tesla.loc['2016-Jul-18']
```

Gün yerine ay veya yıl bilgisi de girebiliriz:

```
tesla.loc['2016-07']: 2016 yılı Temmuz ayına ait bilgiler
```

```
tesla.loc['2016']: 2016 yılına ait tüm bilgiler
```

İstedigimiz iki tarih arasındaki bilgileri de bu iki tarih arasında : yazarak seçebiliriz.

```
tesla.loc['2018-01-10':'2018-01-18']
```

Date	Open	High	Low	Close	Adj Close	Volume
10-01-18	332.20	337.00	330.00	334.80	334.80	4309900
11-01-18	335.24	344.81	333.26	337.95	337.95	6645500
12-01-18	338.63	340.41	333.67	336.22	336.22	4825100
16-01-18	337.54	345.00	334.80	340.06	340.06	6474300
17-01-18	340.47	349.00	339.75	347.16	347.16	7103500
18-01-18	345.67	352.30	343.74	344.57	344.57	5654600

2017 yılı Ocak ayına ait açılış verilerini çekelim.

```
tesla['Open']['2017-January']
Date
2017-01-03    214.860001
2017-01-04    214.750000
...
2017-01-05    226.419998
2017-01-31    249.240005
Name: Open, dtype: float64
```

Pandas'ta zaman serileri ile çalışmanın güzel bir yönü de verileri farklı zaman birimlerinde kolayca hesaplayabilmektir. Bunun için `.resample()` metodu kullanılır. Bu metod, başka metodlarla birlikte kullanılarak istenilen değerler hesaplatılabilir. Buna da zincirleme fonksiyonları denir. Örneğin, tesla zaman serisinde, haftalık ortalama değerleri hesaplatmak: `tesla.resample('W').mean()`. Görüldüğü gibi önce resample metodunu ile günlük verileri haftalık verilere çevirip sonuca `.mean()` metodunu uyguluyoruz.

```
tesla_hafta = tesla.resample('W').mean()
tesla_hafta.head()
Open      High     Low     Close   Adj Close  Volume
```

TARİH VE ZAMAN

Date	Open	High	Low	Close	Adj Close	Volume	1686675	1276540	1333560	1900540	4352125
2013-01-27	35.645	36.762	35.464	36.290	36.290						
2013-02-03	37.769	38.303	37.194	37.861	37.861						
2013-02-10	38.644	39.228	38.252	38.752	38.752						
2013-02-17	38.374	38.856	37.600	38.014	38.014						
2013-02-24	37.217	38.180	36.477	37.272	37.272						

Yıllık ortamlar:

```
tesla_yillik = tesla['Close'].resample('A').mean()
```

```
tesla_yillik.head()
```

```
Date
```

```
2013-12-31    108.227280
```

```
2014-12-31    223.329088
```

```
2015-12-31    230.042898
```

```
2016-12-31    209.767262
```

```
2017-12-31    314.316295
```

```
Freq: A-DEC, Name: Close, dtype: float64
```

Benzer şekilde, zaman serilerine `.std()`, `.max()`, `.min()`, `.sum()` gibi farklı istatistiksel hesapları da uygulayabiliriz. Yukarıda, `.resample()` metodunda haftalık verilere ulaşmak için 'W' argümanını kullandık. Aşağıdaki tabloda, farklı zaman birimleri için kullanılmış gereken argümanlar yer almıştır.

Zaman Birimi	Argüman
Dakika	'min' veya 'T'
Saat	'H'
Gün	'D'
İş Günü	'B'
Hafta	'W'
Ay	'M'
Üç ay	'Q'
Yıl	'A'

Sadece birim zaman değil, iki hafta, üç gün, beş ay gibi istediğimiz sayıda dönem için de hesaplama yapırız.

```
tesla_ikihafta = tesla['Close'].resample('2W').mean()
```

```
tesla_ikihafta.head()
```

```
Date
```

```
2013-01-27    36.290000
```

```
2013-02-10    38.307000
```

PYTHON ile VERİ BİLİMİ

```
2013-02-24    37.684444  
2013-03-10    36.001001  
2013-03-24    36.815000  
Freq: 2W-SUN, Name: Close, dtype: float64
```

Bir başka faydalı metod da hareketli verilerle hesaplama yapmaya yarayan `.rolling()` metodudur. Bu metod ile örneğin hareketli ortalamalar hesaplanabilir. Rolling metodunun aldığı window argümanı istenen hesaplamanın kaç periyot için yapılacağını belirtir. Tesla hisse senedinin günlük kapanış fiyatının 12 günlük hareketli ortalamalarını hesaplayalım.

```
tesla_12gun = tesla['Open'].rolling(window=12).mean()  
print(tesla_12gun)  
Date  
2013-02-06    37.167500  
2013-02-07    37.553333  
2013-02-08    37.922500  
...  
Name: Open, Length: 1259, dtype: float64
```

Resample ve rolling metodlarını zincirleme olarak da kullanabiliriz. Örneğin, tesla kapanış verilerinin 5 haftalık hareketli ortalamalarını bulalım. Bunun için önce `.resample()` ile haftalık verileri hesaplatıp sonra rolling ile 5 haftalık hareketli ortalamaları bulabiliriz.

```
tesla_2w_hareketli = tesla['Close'].resample('W').rolling(window=5).mean()  
print(tesla_2w_hareketli)  
Date  
2013-01-27      NaN  
2013-02-03      NaN  
2013-02-10      NaN  
2013-02-17      NaN  
2013-02-24    37.638100  
2013-03-03    37.315700  
2013-03-10    37.208100  
...  
Freq: W-SUN, Name: Close, Length: 261, dtype: float64
```