ÖĞRENME BİRİMİ 7

TARİH VE STRİNG (METİN) İŞLEMLERİ

Neler Öğreneceksiniz?

Bu öğrenme birimi ile;

- Python dilinde tarih ve zaman nesnelerinin nasıl saklandığını öğrenecek,
- Datetime modülünü kullanmayı öğrenecek,
- Tarih ve zaman verileri ile işlem yapabileceksiniz.

Anahtar Kelimeler:

Tarih, zaman, nesne, metin biçimlendirme.

Hazırlık Çalışmaları

1. Programlamada nesne kavramını araştırınız.
2. Diğer programlama dillerinde tarih işlemlerinin nasıl yapıldığını araştırıp bilgi sahibi olunuz.

7. TARİH VE METİN İŞLEMLERİ

7.1. Tarih Nesnesi

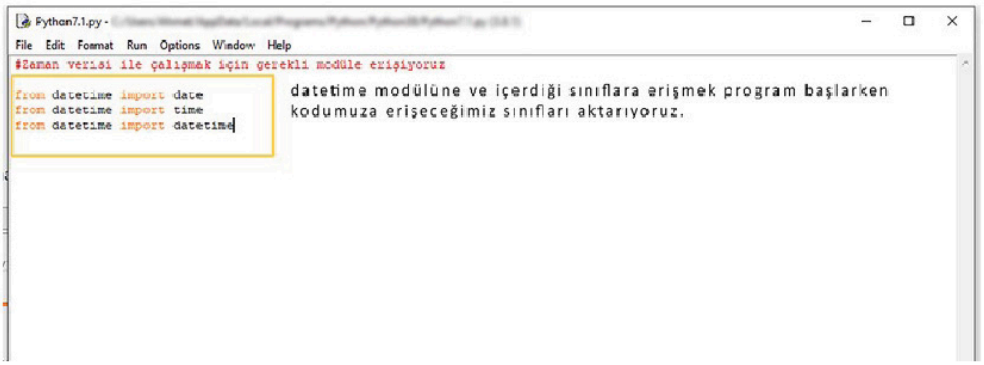
Python dilinde zaman bilgisi tutmak için kendine ait bir veri tipi bulunmaz. Python zaman ve tarih bilgilerini bir değişkene atanmış veri değil de oluşturulmuş bir nesne olarak görüp işler. Bu nedenle zaman nesnesi ile çalışmak için programlama dili ile birlikte gelen **datetime** modülünü kullanmak gerekir. **datetime** modülü; zaman, saat ve tarihlerle ilgili işlemler için çeşitli fonksiyonlar ve özellikler sağlayan sınıfları içerir.

Sınıflar, barındırdıkları nesnelerin özelliklerinin ve davranışlarının tanımlandığı genel şablonlardır.

Python dilinin mevcut sürümü olan Python 3.8.2 versiyonunda datetime modülünün barındırdığı, programlarınızda kullanabileceğiniz sınıflar şunlardır:

1. **datetime.date:** Tarihle ilgili nitelikleri ve fonksiyonları barındıran sınıftır. year (yıl), month (ay) ve day (gün) özelliklerini içerir.
2. **datetime.time:** Zamanla ilgili nitelikleri ve fonksiyonları barındıran sınıftır. hour (saat), minute (dakika), second (saniye) , microsecond (mikrosaniye) ve tzinfo (saat dilimi) özelliklerini içerir.
3. **datetime.datetime:** date ve time sınıflarının birleşiminden ve ilave birkaç fonksiyondan oluşur. Örneklerde sıklıkla kullanılacak sınıf budur.
4. **datetime.timedelta:** İki date, time veya datetime nesnesi arasındaki zaman farkını mikrosaniye cinsinden veren sınıftır.
5. **datetime.tzinfo:** date ve time sınıflarının saat dilimi özelliklerini tutmak için oluşturulmuş abstract (temel) sınıftır.

Python dili ile modüllere erişmeyi ve içerdiği fonksiyonları kullanmayı geçen bölümde öğrenmiştiniz.



Görsel 7.1: datetime modülünden sınıfları içe aktarma

```
from datetime import datetime
```

satırı ile istediğiniz sınıfı kodunuza aktarıp erişebileceğiniz gibi

```
import datetime
```

satırı ile bütün d**atetime** modülünü içe aktarıp erişebilirsiniz. Ancak fonksiyonlar bölümünden hatırlanacağı üzere kullanılacak tipin başına ait olduğu modül isminin de yazılması gerekir.

now( ) : datetime modülü içindeki datetime sınıfına ait bu fonksiyon içinde bulunulan andaki tarih ve saat bilgilerini verir.

Aşağıdaki örnekte bir tarih nesnesi oluşturulmuş ve o anki zaman bilgisi oluşturulan nesneye atanmıştır.

Örnek 1:

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.
an = datetime.now() #tarih nesnesi oluşturulup now() fonksiyonu ile zaman bilgisi atanıyor.


print("tarih ve saat = ",an)  #an nesnesi ekrana yazdırılıyor.
```

Çıktı:

```
tarih ve saat = 2020-06-11 11:38:14.170414
```

today( ) : now() fonksiyonu ile aynı işleve sahiptir. Bulunulan günün tarih ve saat bilgilerini verir. Eğer **datetime** sınıfı ile değil de **date** sınıfı ile beraber kullanılırsa sadece yıl, ay ve gün bilgisini verecektir. **now()** fonksiyonu saat bilgisi içerdiğinden date sınıfı ile beraber kullanılmaz.

Örnekte **date** modülü ile **today( )** fonksiyonu kullanılmıştır.

Örnek 2:

```
from datetime import date #date sınıfı içe aktarılıyor.
bugun = date.today()
print("bugun = ",bugun)
```

Çıktı:

```
bugun =  2020-06-11
```

İstenirse tarih nesnesinin içerdiği veri; yıl, ay, gün, saat, dakika, saniye ve mikrosaniye olarak çağrılıp kullanılabilir.

Örnekte tarih nesnesini oluşturan zaman verileri ayrı ayrı yazdırılmıştır. **weekday()** fonksiyonu ise haftanın kaçıncı gününde olunduğunu verir.

Örnek 3:

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.

bugun = datetime.today()

print("bugun = ",bugun)
print(bugun.weekday()) #haftanın kaçıncı günü - Pazartesi 0 ile Pazar 6 arası
print(bugun.year) # yıl bilgisi
print(bugun.month) # ay bilgisi
print(bugun.day)    # gün bilgisi
print(bugun.hour)    # saat bilgisi
print(bugun.minute)   # dakika bilgisi
print(bugun.second)    # saniye bilgisi
```

Çıktı:

```
bugun =  2020-06-11 12:20:25.946320
3
2020
6
11
12
20
25
```

Tarih nesnelerini, kendi tarih verilerinizi vererek de tanımlayabilirsiniz.

Örnekte yıl, ay ve gün verileri verilerek tarih tanımlanmıştır.

Örnek 4:

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.

dogum_tarihi = datetime(year=1923,month=10,day=29)
print("doğum tarihi = ",dogum_tarihi)
```

Çıktı:

```
dogum tarihi =  1923-10-29 00:00:00
```

Saat verilerini de tanımlama sırasında atayabilirsiniz.

Örnek 5:

```
from datetime import datetime #datetime sınıfı içe aktarılıyor.

dogum_tarihi = datetime(year=1923,month=10,day=29, hour=8, minute=30, second=35)
print("doğum tarihi = ",dogum_tarihi)
```

Çıktı:

```
dogum tarihi =  1923-10-29 08:30:35
```

Aşağıda programa girdi olarak verilen doğum günü verilerini tarih nesnesine çevirip bugün itibari ile aradaki farkı bulan program gösterilmiştir.

Örnek 6:

```
from datetime import datetime

bugun = datetime.today()
print("Bu günün tarihi", bugun)
yil = int(input("Lütfen doğduğunuz yılı girin:"))
ay = int(input("Lütfen doğduğunuz ayı girin:"))
gun = int(input("Lütfen doğduğunuz günü girin:"))
dogum = datetime(year=yil,month=ay,day=gun)
yas = bugun-dogum
print(yas)
```

Çıktı:

```
Bu günün tarihi 2020-06-11 12:57:33.983500
Lütfen doğduğunuz yılı girin:1979
Lütfen doğduğunuz ayı girin:2
Lütfen doğduğunuz günü girin:20
15087 days, 12:57:33.983500
```

Bu örnekte elde edilen yas verisi aslında yukarıda sınıflarda bahsedilen timedelta nesnesidir. İki tarih arasında aritmetiksel bir işlem yapıldığında sonuç timedelta yani zaman farkı nesnesi ile tutulur.

Yukarıdaki örneğe aşağıdaki satırları ekleyerek timedelta nesnesinin niteliklerine ulaşabilirsiniz.

```
print(yas)
print(yas.days," gün")
print(yas.seconds," saniye"ı
print(yas.microseconds," mikrosaniye")
```

Çıktı:

```
15087 days, 13:43:43.797852
15087  gün
49423  saniye
797852  mikrosaniye
```

timedelta nesnesi ile iki tarih arasındaki farka gün, saniye veya mikrosaniye biçiminde ulaşılabilir.

```
bugun = datetime.date.today()
dun = bugun - datetime.timedelta(days = 1)
yarin = bugun + datetime.timedelta(days = 1)
print("Dün :",dun)
print("Bugün :",bugun)
print("Yarın :", yarin)
```

Çıktı:

```
Dün : 2020-06-10
Bugün : 2020-06-11
Yarın : 2020-06-12
```

Örnekte datetime modülü içe aktarıldı ve bu modüldeki date ve timedelta nesneleri kullanıldı. today() fonksiyonu ile bugünün tarihi alındı. timedelta nesnesine 1 gün fark tanımlanıp bugünün tarihine eklendi ve çıkarıldı.

Aşağıdaki örnekte bugünden 150 gün sonra tarihin ne olacağı hesaplanmıştır.

Örnek 8:

```
import datetime

bugun = datetime.datetime.today()
fark = datetime.timedelta(days = 150)
gelecek = bugun + fark

print("150 gün sonrası :",gelecek)
print("Biçimlendirilmiş Tarih :",gelecek.strftime('%c'))
```

Çıktı:

```
150 gün sonrası : 2020-11-08 15:29:52.305214
Biçimlendirilmiş Tarih : Sun Nov  8 15:29:52 2020
```

Not: **Programın sonunda eldeki tarih verisini biçimlendirmek için strftime() isimli bir metot kullanıldı.**

7.2. Tarih Bilgisinin Biçimlendirilmesi

Görsel 7.2: Tarih bilgisi

strftime() : strftime() fonksiyonu date, datetime veya time nesnesini kullanarak bize tarih ve zamanı bildiren biçimlendirilmiş string bir değer verir. Böylece size tarih ve zaman bilgilerini ihtiyaçlarınız doğrultusunda biçimlendirme imkanı sunar.

Yukarıdaki örnekte '%c' biçimlendiricisi kullanılarak

```
Sun Nov  8 15:29:52 2020
```

formatında bir çıktı elde edildi. Biçimleyicileri bir arada kullanarak tarih verilerinizi istediğiniz gibi biçimlendirebilirsiniz.

Örnek 9:

```
import datetime


an = datetime.datetime.now()


print(an.strftime('%Y')) # Yıl
print(an.strftime('%X')) # Saat
print(an.strftime('%d')) # Gün - ayın kaçıncı günü
print(an.strftime('%A')) # Gün - İsim olarak
print(an.strftime('%B')) # Ay - İsim olarak
```

Çıktı:

```
2020
16:31:34
11
Thursday
June
```

Biçimlendiriciler kullanarak elinizdeki tarih verisini istediğiniz gibi biçimlendirebilirsiniz. Çıktıda ay ve gün isimlerinin İngilizce olarak yazıldığı görülebilir. Biçimlendirme işlemlerinde yerel yazım formatlarına uygun davranılmak isteniyorsa **locale** isimli modül çağrılarak programın çalıştığı bilgisayarın yerel dil ayarlarını almasını sağlayabilirsiniz.

```
import locale
locale.setlocale(locale.LC_ALL, '')
```

Satırları programın çalıştığı bilgisayarın dil ayarları ile biçimlendirme yapmasını sağlar veya

```
import locale
locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')
```

çalışılan platformdan bağımsız olarak biçimleyicileri tek bir dile ayarlayabilirsiniz. Yukarıdaki satır programa Türkçe biçimlendirme ayarlarını kullanmasını söyler.

Örnek 10:

```
import datetime
import locale

locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')
an = datetime.datetime.now()

print(an.strftime('%Y')) # Yıl
print(an.strftime('%X')) # Saat
print(an.strftime('%d')) # Gün - ayın kaçıncı günü
print(an.strftime('%A')) # Gün - İsim olarak
print(an.strftime('%B')) # Ay - İsim olarak

print(an.strftime('%d %B %Y')) # gün ay ismi yıl şeklinde
print(an.strftime('%d.%m.%Y tarihinde buluşalım.')) # aralarına nokta konarak
```

Çıktı:

```
2020
16:53:54
11
Perşembe
Haziran
11 Haziran 2020
11.06.2020 tarihinde buluşalım.
```

Tabloda strftime() fonksiyonu ile kullanabileceğiniz biçimlendiricileri inceleyebilirsiniz (Tablo 7.1).

Tablo 7.1. Strftime() Fonksiyonu ile Kullanılabilen Biçimlendiriciler

Biçimlendirici	Açıklama	Örnek
%a	Kısaltılmış gün ismi	Çar
%A	Gün ismi	Çarşamba
%w	Sayı olarak haftanın kaçıncı günü (Pazar 0, Cumartesi 6)	3
%d	Sayı olarak ayın kaçıncı günü	30
%b	Kısaltılmış ay ismi	Haz
%B	Ay ismi	Haziran
%m	Sayı olarak ay (tek haneli ise başına 0 eklenerek)	06
%-m	Sayı olarak ay	6
%y	Yıl – son iki rakam olarak	20
%Y	Yıl	2020
%H	Saat (24-Saatlik format) – 0 eklenmiş sayı	07
%-H	Saat (24-Saatlik format)	18
%I	Saat (12-Saatlik format) 0 eklenmiş sayı	07
%-I	Saat (12-Saatlik format)	6
%p	AM / ÖÖ – PM / ÖS bilgisi	AM
%M	Dakika – 0 eklenmiş sayı	06
%-M	Dakika	6
%S	Saniye – 0 eklenmiş sayı	05
%-S	Saniye	5
%f	Mikrosaniye – 0 eklenmiş sayı	000000
%j	Yılın kaçıncı günü	273
%U	Yılın kaçıncı haftası (Pazar gününden başlayarak)	39
%W	Yılın kaçıncı haftası (Pazartesi gününden başlayarak)	39
%c	Yerel biçim ayarlarına göre tarih ve saat	Çar Haz 10 07:06:05 2020
%x	Yerel biçim ayarlarına göre tarih	06/10/20
%X	Yerel biçim ayarlarına göre tarih ve saat	07:06:05
%%	Karakter girdisi	%

Örnekte biçimlendiricilerin kullanımlarını inceleyebilirsiniz.

Örnek 11:

```
import datetime
import locale
locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')

bugun = datetime.datetime.now()
yarin = datetime.date(bugun.year, bugun.month, bugun.day+1)

print('Bugün :',bugun)
print(bugun.strftime('%d/%m/%Y'))
print('Yarın :',yarin)
print(yarin.strftime('%d/%m (%Y)'))

print(bugun.strftime('%m/%d/%Y, %H:%M:%S'))
print(bugun.strftime('%d %b, %Y'))
print(bugun.strftime('%d %B, %Y'))
print(bugun.strftime('%I%p'))
```

Çıktı:

```
Bugün : 2020-06-12 13:04:50.267157
12/06/2020
Yarın : 2020-06-13
13/06 (2020)
06/12/2020, 13:04:50
12 Haz, 2020
12 Haziran, 2020
01ÖS
```

7.2.1. String (Metin) Olarak Girilen Değerlerin Tarih Bilgisinin Biçimlendirilmesi

Tarih nesnesi oluşturmayı ve içeriğini biçimlendirerek yazdırmayı öğrendiniz. Şimdi metin hâlindeki veriyi tarih nesnesine çevirmeyi öğreneceksiniz. Diyelim ki elinizde input() fonksiyonu ile girilmiş veya başka bir kaynaktan gelmiş "29 Ekim 1923 saat 14:24:11" gibi bir metin verisi var. Bu veriyi çeşitli string fonksiyonları kullanarak anlamlı parçalara bölebilirsiniz ama bu çok meşakkatli bir işlem olacaktır.

strptime() : Bu işlemler için datetime modülünde yukarıdaki biçimlendiriciler ile beraber kullanılacak **strptime()** fonksiyonu bulunur. Bu fonksiyon tarih ve zaman bilgisi içeren herhangi bir metni veya karakter dizisini tarih nesnesine dönüştürür. Çalışırken bilgiyi içeren metni ve biçimlendiricileri parametre olarak alır.

Örnek 12:

```
import datetime
import locale

locale.setlocale(locale.LC_ALL, 'Turkish_Turkey.1254')

tarih_metni = '29 Ekim 1923 saat 14:32:11'
print("Metin halindeki tarih = ", tarih_metni)
tarih_nesnesi = datetime.datetime.strptime(tarih_metni,'%d %B %Y saat %H:%M:%S')
print("Tarih nesnesi = ",tarih_nesnesi) print(tarih_nesnesi.year)
print(tarih_nesnesi.month)
print(tarih_nesnesi.day)
```

Çıktı:

```
Metin halindeki tarih =  29 Ekim 1923 saat 14:32:11
Tarih nesnesi =  1923-10-29 14:32:11
1923
10
29
```

Sıra Sizde: Aşağıdaki metni oluşturacağınız bir tarih nesnesine atayacağınız programı yazıp deneyiniz. Metindeki formatın ABD tarih formatında olduğunu gözden kaçırmayınız.

Tarih metni : 'Oct. 30, 2014, 6:17 a.m.'

7.3. String (Metin) İşlemleri

Python'da stringler karakterleri temsil eden baytları içeren listelerdir. Python doğrudan karakterleri temsil eden bir veri tipi içermez. Bu nedenle tek bir karakteri temsil eden veri, tek elemanlı bir string dizisidir. String verileri kolayca düzenlemek ve değiştirmek için dil içinde gelen gömülü fonksiyonlar vardır. Python'da string verilerle nasıl çalışıldığı örnekler eşliğinde incelenebilir.

7.3.1. String Verileri Birleştirme

Aşağıdaki kod bloğunda iki karakter dizisi yani string veri oluşturulup birleştirilmiştir.

Örnek 13:

```
ad = "Cüneyt"
soyad = "Arkın"
ad_soyad = ad + soyad
print(ad_soyad)
```

Çıktı:

```
CüneytArkın
```

Ad, soyad arasına boşluk bırakılsın. Bunun için boş bir karakter içeren yeni değişken oluşturulsun.

```
ad = "Cüneyt"
soyad = "Arkın"
ara=" "
ad_soyad = ad + ara + soyad
print(ad_soyad)
```

Çıktı:

```
Cüneyt Arkın
```

Gördüğünüz gibi boşluk da bir karakter olarak algılanır.

7.3.2. String Veri İçindeki Bir Karaktere Erişme

Stringler, karakterlerden oluşmuş listeler olduğundan köşeli parantez ([]) ve index sayısı ile istenilen karaktere rahatça erişebilir.

Örnek 14:

```
sehir = "Gaziantep"
karakter = sehir[3] #dördüncü harfi alıyoruz
print(karakter) #i
print(sehir[0]) #G
print(sehir[5]) #n
```

Çıktı:

```
i
G
n
```

7.3.3. String Verinin Uzunluğu

Daha önce listelerde kullanılan len() fonksiyonu ile string verinin uzunluğunu öğrenebilirsiniz.

Örnek 15:

```
sehir = "Gaziantep"
print("sehir değişken boyutu: ", len(sehir)) #sehir değişkeninin boyutu
boyut=len(sehir)
son_karakter=sehir[boyut-1] # değişkendeki son karaktere erişilir.
print("Son karakter: ", son_karakter)
for harf in sehir:     #stringdeki bütün harflere for döngüsü ile erişilir.
    print(harf)
```

Çıktı:

```
sehir değişken boyutu: 9
Son karakter: p
G
a
z
i
a
n
t
e
p
```

7.3.4. String Veriyi Parçalama (Slice) ve Bölme (Split)

slice() fonksiyonu ile köşeli parantezleri kullanarak metinden parçalar alınabilir. Bunun için köşeli parantez içine çekilmek istenilen karakter aralığının indisini girmek yeterlidir.

Degisken[baslangicIndex : bitisIndex] şeklinde kullanılır. Burada, **baslangicIndex** dâhil, **bitisIndex** dâhil değildir.

Örnek 16:

```
veri = "Bilişim Teknolojileri"
# Köşeli parantez içerisine yazılan değerler dâhildir.
print(veri)
print(veri[2:8]) # string'in 2 ile 8 aralığındaki değerini alır.(2.index dahil, 8.index dahil değildir).
print(veri[2:]) # 2.indexten itibaren stringi bütünüyle alır.
print(veri[5:15]) #(5.index dâhil, 15.index dâhil değildir.)
print(veri[:12]) # 0. indexten 12. index'e kadar string'i parçalayacaktır.
```

Çıktı:

```
Bilişim Teknolojileri
lişim
lişim Teknolojileri
im Teknolo
Bilişim Tekn
```

split() fonksiyonu ile, metin verisi belirlenen karakterler baz alınarak bölünebilir. Bölünen metin parçaları dizi hâlinde verilecektir.

Örnek 17:

```
veri = "Bilişim Teknolojileri"
metin="ilkbahar,yaz,sonbahar,kış"

veri_bolum = veri.split(" ") #veri değişkenini boşluktan itibaren bölüyoruz.
metin_bolum = metin.split(",") #metin değişkenini virgüllerden bölüyoruz.
metin_bolum_ikieleman = metin.split(",",1) #1 parametresi vererek metni sadece 2 parçaya bölüyoruz

print(veri_bolum)
print(metin_bolum)
print(metin_bolum_ikieleman)
```

Çıktı:

```
['Bilişim', 'Teknolojileri']
['ilkbahar', 'yaz', 'sonbahar', 'kış']
['ilkbahar', 'yaz,sonbahar,kış']
```

7.3.5. String Veri İçinde Karakter Değiştirme, Karakter Ekleme ve Çıkarma

replace() fonksiyonu ile bir string içerisindeki herhangi bir karakter değiştirilebilir.

Örnek 18:

```
kelime = "Bilişim"
cumle = "Merhaba Dünya"
print(kelime)
print(kelime.replace("i","o")) #kelimedeki i'leri o karakteri ile değiştirelim.
print(cumle)
print(cumle.replace("Merhaba","Selam")) #cümledeki "Merhaba"'yı "Selam" ile değiştirelim.
```

Çıktı:

```
Bilişim
Boloşom
Merhaba Dünya
Selam Dünya
```

strip() fonksiyonu ile bir string içinden karakter çıkarılabilir.

Örnek 19:

```
cumle = "  bilişim teknolojilerine giriş  "
# baştaki ve sondaki boşluklar silinir.
print(cumle.strip())
# <bosluk>,b,i,l karakterleri silinir.
print(cumle.strip(" bil"))
# Parametre cümlede bulunmadığı için
# hiçbir karakter çıkarılmaz.
print(cumle.strip("prg"))
cumle2 = 'python çok kullanışlı'
print(cumle2.strip("pyt"))
```

Çıktı:

```
bilişim teknolojilerine giriş
şim teknolojilerine giriş
  bilişim teknolojilerine giriş
hon çok kullanışlı
```

join() fonksiyonu ile bir string verinin içerdiği her bir karakterden sonra yeni bir karakter eklenebilir.

Örnek 20:

```
#Formatı: "eklemek istenilen string ya da karakter değer".join(değişkenin kendisi
= elimizde olan string veri)
kelime = "Gaziantep"
print(".".join(kelime))
```

Çıktı:

```
G.a.z.i.a.n.t.e.p
```

7.3.6. String Veri İçinde Bir Karakterin Yerini veya Metnin Karakteri İçerip İçermediğini Bulma

find() fonksiyonu ile bir metin içinde aranılan karakterin kaçıncı indekste olduğu bulunabilir.

Örnek 21:

```
kelime = "Bilişim Teknolojileri"
# metin içinde 'm' nin indexini arıyoruz.
print(kelime.find('m'))
# metnin içinde 'no' nın indexini arıyoruz, burada indexi n'nin indexi(11)
olarak geri döndürecektir.
print(kelime.find('no'))
# 4. indexten başlayarak metin içinde i'yı tarayacaktır.
print(kelime.find('i',4))
# 1. index ile 7. index dahil olmak üzere aradaki metinde 'i' yı arayacaktır.
print(kelime.find('i',1,7))
# eğer aradığınız veri metinde yoksa -1 sonuç olarak döndürülecektir.)
print(kelime.find('z'))
```

Çıktı:

```
6
11
5
1
-1
```

in operatörü ile bir metin içinde herhangi bir karakterin olup olmadığını sorgulayabilirsiniz. Bu operatör geriye boolean , "True" ya da "False" bir değer döndürür. Aranan karakterler metin içinde bulunduğunda "True", aksi takdirde "False" döndürür.

Örnek 22:

```
kelime = "Gaziantep"
print("G" in kelime)
print("k" in kelime)
```

Çıktı:

```
True
False
```

Örnek 23:

```
kelime = "Gaziantep"
print("G" not in kelime)
print("k" not in kelime)
```

Çıktı:

```
False
True
```

7.3.7. String Veri İle Büyük ve Küçük Harf Değişimi Yapma

Python'da karakter dizilerinde büyük ve küçük harf değişikliği için kullanılabilecek aşağıdaki fonksiyonlar vardır.

upper() : Karakter dizisindeki bütün harfleri büyütür.

lower() : Karakter dizisindeki bütün harfleri küçültür.

capitalize() : Karakter dizisinin ilk harfini büyütür.

title() : Karakter dizisindeki her kelimenin ilk harfini büyütür.

swapcase() : Karakter dizisindeki büyük harfleri küçük, küçük harfleri büyük hâle getirir.

Örnek 24:

```
kelime = 'Bilişim teknolojileri'
print(kelime.upper()) # metni büyük harfe çevirir.
print(kelime.lower()) # metni küçük harfe çevirir.
print(kelime.capitalize()) # dizinin harfini büyütür.
print(kelime.title()) # kelimelerin ilk harflerini büyütür.
print(kelime.swapcase()) # büyük  küçük değişimi yapar.
```

Çıktı:

```
BILIŞIM TEKNOLOJILERI
bilişim teknolojileri
Bilişim teknolojileri
Bilişim Teknolojileri
bILIŞIM TEKNOLOJILERI
```

ÖLÇME VE DEĞERLENDİRME 7

1. Aşağıdakilerden hangisi datetime modülünün içerdiği sınıflardan değildir?

A) date
B) datetime
C) math
D) time
E) timedelta

2. Bulunulan günün tarih ve saat bilgilerini veren fonksiyon hangisidir?

A) print()
B) float()
C) pow()
D) now()
E) sin()

3. Bir metindeki bütün harfleri büyük yapan fonksiyon aşağıdakilerden hangisidir?

A) lower()
B) capitalize()
C) upper()
D) title()
E) print()

4. Aranılan karakterin bir metin içinde kaçıncı indekste olduğunu veren fonksiyon ..... fonksiyonudur.

5. Gün ismini "Çarşamba" olarak yazdıran biçimlendirici aşağıdakilerden hangisidir?

A) %b
B) %A
C) %a
D) %w
E) %d

NOT: **Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları veya faaliyetleri geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme birimine geçiniz.**