Metinlerle Çalışmak

Table of contents

Bölüm 3: Metinlerle Çalışmak: String İşlemleri

3.1 String Veri Tipi

Python'da **stringler (metinler)**, harfler, rakamlar, semboller ve boşluklar gibi karakterlerden oluşan dizilerdir. Stringler, programlamada kullanıcıdan veri almak, veriyi işlemek ve çıktı üretmek için sıklıkla kullanılır. Örneğin, bir kullanıcının adını, bir web sitesinin adresini veya bir mesajı saklamak için stringler kullanabiliriz.

3.1.1 Stringleri Tanımlama

Python'da stringleri tanımlamak için, metni tek tırnak (') veya çift tırnak (") içine alırız. Örneğin:

```
mesaj1 = 'Merhaba Dünya!'
mesaj2 = "Python programlama dili"
```

Hem `mesaj1` hem de `mesaj2` değişkenleri string veri tipini tutar. Tek tırnak veya çift tırnak kullanımı tamamen sizin tercihinize bağlıdır, ancak aynı string içinde aynı tür tırnak işaretini kullanmanız gerekir.

3.1.2 Çok Satırlı Stringler

Birden fazla satırdan oluşan stringler oluşturmak için üç tane tek tırnak (''') veya üç tane çift tırnak (""") kullanırız.

Örnek:

```
uzun_mesaj = """Bu,
çok satırlı
bir stringdir."""

print(uzun_mesaj)
```

Çıktı:

```
Bu,
çok satırlı
bir stringdir.
```

3.2 String Metotları

Python, stringler üzerinde birçok işlem yapmak için kullanabileceğiniz birçok **metot** sunar. Metotlar, bir nesne üzerinde belirli bir işlemi gerçekleştirmek için kullanılan fonksiyonlardır. String metotlarına erişmek için, string değişkeninin adını yazdıktan sonra nokta (.) koyarız ve ardından metodun adını yazarız.

3.2.1 Temel String Metotları

İşte bazı temel string metotları:

- **`len()`:** Bir stringin uzunluğunu (karakter sayısını) döndürür.

```
mesaj = "Merhaba"
uzunluk = len(mesaj)  # uzunluk değişkeninin değeri 7 olur
print(uzunluk)
```

- **`upper()`:** Bir stringi büyük harfe dönüştürür.

```
mesaj = "Merhaba"
buyuk_harf = mesaj.upper()  # buyuk_harf değişkeninin değeri "MERHABA" olur
print(buyuk_harf)
```

- **`lower()`:** Bir stringi küçük harfe dönüştürür.

```
mesaj = "Merhaba"
kucuk_harf = mesaj.lower()  # kucuk_harf değişkeninin değeri "merhaba" olur
print(kucuk_harf)
```

- **`capitalize()`:** Bir stringin ilk harfini büyük harfe, diğer harflerini küçük harfe dönüştürür.

```
mesaj = "merhaba dünya"
baslik = mesaj.capitalize()  # baslik değişkeninin değeri "Merhaba dünya" olur
print(baslik)
```

- **`title()`:** Bir stringdeki her kelimenin ilk harfini büyük harfe, diğer harflerini küçük harfe dönüştürür.

```
mesaj = "python programlama dili"
baslik = mesaj.title()  # baslik değişkeninin değeri "Python Programlama Dili" olur
print(baslik)
```

- **`strip()`:** Bir stringin başındaki ve sonundaki boşlukları siler.

```
mesaj = "   Merhaba Dünya!   "
temiz_mesaj = mesaj.strip()  # temiz_mesaj değişkeninin değeri "Merhaba Dünya!" olur
print(temiz_mesaj)
```

- **`replace()`:** Bir stringdeki belirli bir alt stringi başka bir alt string ile değiştirir.

```
mesaj = "Merhaba Dünya!"
yeni_mesaj = mesaj.replace("Dünya", "Python")  # yeni_mesaj değişkeninin değeri "Merhaba
print(yeni_mesaj)
```

- **`split()`:** Bir stringi belirli bir karaktere göre böler ve bir liste döndürür.

```
mesaj = "elma,armut,muz"
meyveler = mesaj.split(",")  # meyveler listesi ["elma", "armut", "muz"] olur
print(meyveler)
```

- **`find()`:** Bir stringdeki belirli bir alt stringin ilk bulunduğu indeksi döndürür. Eğer alt string bulunmazsa `-1` döndürür.

```
mesaj = "Merhaba Dünya!"
indeks = mesaj.find("Dünya")  # indeks değişkeninin değeri 7 olur
print(indeks)
```

- **`count()`:** Bir string içinde belirli bir alt stringin kaç kez geçtiğini sayar. Eğer alt string bulunmazsa 0 döndürür.

```
metin = "Merhaba Dünya!"

print(metin.count("a"))  # Çıktı: 3
```

3.2.2 String Formatlama (String Formatting)

String formatlama, stringler içine değişkenlerin değerlerini veya diğer ifadelerin sonuçlarını eklemek için kullanılır. Python'da string formatlama için birkaç farklı yöntem vardır. İşte en yaygın kullanılan üç yöntem:

1. `%` Operatörü ile Formatlama

Bu yöntem, C dilindeki `printf()` fonksiyonuna benzer. `%` operatörü, string içinde yer tutucu olarak kullanılır ve ardından değişkenlerin değerleri veya diğer ifadeler gelir.

Örnek:

```
isim = "Ahmet"
yas = 30

mesaj = "Benim adım %s ve yaşım %d" % (isim, yas)
print(mesaj)  # Çıktı: Benim adım Ahmet ve yaşım 30
```

Bu örnekte, `%s` bir string yer tutucu, `%d` ise bir integer yer tutucudur. `%` işaretinden sonra parantez içinde yer tutuculara karşılık gelen değerler gelir.

2. `.format()` Metodu ile Formatlama

`.format()` metodu, string içinde süslü parantezler (`{}`) kullanarak yer tutucular oluşturur ve ardından bu yer tutuculara karşılık gelen değerler `.format()` metoduna argüman olarak verilir.

Örnek:

```
isim = "Ahmet"
yas = 30

mesaj = "Benim adım {} ve yaşım {}".format(isim, yas)
print(mesaj)  # Çıktı: Benim adım Ahmet ve yaşım 30
```

Bu örnekte, `{}` yer tutucuları, `.format()` metodunda sırasıyla `isim` ve `yas` değişkenlerine karşılık gelir.

3. f-Stringler (Python 3.6 ve Sonrası)

f-Stringler, string formatlamanın en yeni ve en okunabilir yöntemidir. Stringin başına `f` harfi eklenir ve süslü parantezler (`{}`) içine doğrudan değişkenlerin veya diğer ifadelerin adları yazılır.

Örnek:

```
isim = "Ahmet"
yas = 30

mesaj = f"Benim adım {isim} ve yaşım {yas}"
print(mesaj)  # Çıktı: Benim adım Ahmet ve yaşım 30
```

Bu örnekte, `{isim}` ve `{yas}` ifadeleri, sırasıyla `isim` ve `yas` değişkenlerinin değerlerine karşılık gelir.

3.2.3 String İndeksleme (Indexing) ve Dilimleme (Slicing)

Stringler, karakterlerden oluşan bir dizidir. Her karakterin, string içinde bir **indeksi (index)** vardır. İndeksler 0'dan başlar. Yani, bir stringin ilk karakterinin indeksi 0, ikinci karakterinin indeksi 1'dir ve bu şekilde devam eder.

Python'da, bir stringin belirli bir karakterine erişmek için indeksini köşeli parantez (`[]`) içine yazarız. Örneğin:

```
mesaj = "Merhaba"

ilk_harf = mesaj[0]   # ilk_harf değişkeninin değeri "M" olur
ikinci_harf = mesaj[1]  # ikinci_harf değişkeninin değeri "e" olur
son_harf = mesaj[6]   # son_harf değişkeninin değeri "a" olur

print(ilk_harf)
print(ikinci_harf)
print(son_harf)
```

Negatif İndeksleme:

Python'da, stringlere sondan başlayarak da indeksleme yapabiliriz. Bunun için negatif indeksler kullanırız. Son karakterin indeksi -1, sondan ikinci karakterin indeksi -2'dir ve bu şekilde devam eder.

```
mesaj = "Merhaba"

son_harf = mesaj[-1]   # son_harf değişkeninin değeri "a" olur
sondan_ikinci_harf = mesaj[-2]  # sondan_ikinci_harf değişkeninin değeri "b" olur

print(son_harf)
print(sondan_ikinci_harf)
```

Dilimleme (Slicing):

String'lerin belirli bir bölümünü almak için **dilimleme** işlemini kullanabiliriz. Dilimleme, başlangıç ve bitiş indeksleri belirtilerek yapılır. Başlangıç indeksi dahil, bitiş indeksi hariçtir.

```
mesaj = "Merhaba Dünya!"

ilk_uc_harf = mesaj[0:3]   # ilk_uc_harf değişkeninin değeri "Mer" olur
dunya = mesaj[8:13]      # dunya değişkeninin değeri "Dünya" olur

print(ilk_uc_harf)
print(dunya)
```

Başlangıç veya bitiş indeksini belirtmezsek, varsayılan olarak stringin başlangıcından veya sonuna kadar alınır.

```
mesaj = "Merhaba Dünya!"

ilk_alti_harf = mesaj[:6]  # ilk_alti_harf değişkeninin değeri "Merhab" olur
son_bes_harf = mesaj[7:]   # son_bes_harf değişkeninin değeri "Dünya!" olur

print(ilk_alti_harf)
print(son_bes_harf)
```

İndeks Hatası (IndexError):

Eğer stringin uzunluğundan büyük veya eşit bir indeks kullanırsanız, Python bir `IndexError` hatası verecektir. Örneğin:

```
mesaj = "Merhaba"
harf = mesaj[7]  # IndexError: string index out of range
```

3.2.4 Escape Karakterleri

Bazı karakterleri string içinde doğrudan kullanamayız. Örneğin, çift tırnak içinde çift tırnak kullanmak istiyorsak, escape karakteri olan ters bölü (\) kullanmamız gerekir.

```
mesaj = "Bu bir \"tırnak işareti\" örneğidir."
print(mesaj)  # Çıktı: Bu bir "tırnak işareti" örneğidir.
```

Diğer yaygın escape karakterleri:

- `\n`: Yeni satır
- `\t`: Tab
- `\\`: Ters bölü

3.2.5 Stringlerin Değiştirilemezliği (Immutability)

Python'da stringler **değiştirilemez (immutable)** veri tipleridir. Yani, bir string oluşturulduktan sonra, karakterleri değiştirilemez.

```
mesaj = "Merhaba"
mesaj[0] = "m"  # TypeError: 'str' object does not support item assignment
```

String'leri değiştirmek için, yeni bir string oluşturmamız gerekir.

3.2.6 String Metotlarının Zincirlenmesi (Method Chaining)

Birçok string metodunu art arda çağırarak, string üzerinde birden fazla işlem yapabiliriz.

```
mesaj = "   MERHABA DÜNYA!   "
temiz_mesaj = mesaj.strip().lower()
print(temiz_mesaj)  # Çıktı: merhaba dünya!
```

Bu örnekte, `strip()` metodu ile baştaki ve sondaki boşluklar silinir, ardından `lower()` metodu ile string küçük harfe dönüştürülür.

3.2.7 `in` ve `not in` Operatörleri

Bir stringin başka bir string içinde olup olmadığını kontrol etmek için `in` ve `not in` operatörlerini kullanabiliriz.

```
mesaj = "Merhaba Dünya!"

if "Dünya" in mesaj:
    print("Mesajda 'Dünya' kelimesi geçiyor.")

if "Python" not in mesaj:
    print("Mesajda 'Python' kelimesi geçmiyor.")
```

3.2.8 ASCII ve Unicode

Bilgisayarlar, metinleri ve karakterleri sayılarla temsil eder. Bu temsil için kullanılan iki temel sistem vardır: ASCII ve Unicode.

- **ASCII (American Standard Code for Information Interchange):** İngilizce alfabesindeki harfler, rakamlar ve bazı özel karakterler için 128 farklı karakteri temsil eden bir kodlama sistemidir.
- **Unicode:** Dünya dillerindeki hemen hemen tüm karakterleri kapsayan, çok daha geniş bir kodlama sistemidir. Python 3, varsayılan olarak Unicode kullanır.

3.3 Bölüm Sonu Quiz

1. Aşağıdaki kod bloğunun çıktısı nedir?

```
mesaj = "  Python Programlama  "
print(len(mesaj.strip()))
```

2. Aşağıdaki kod bloğunun çıktısı nedir?

```
metin = "merhaba dünya"
print(metin.title())
```

3. Aşağıdaki kod bloğunda boşluğa hangi ifade gelmelidir?

```
isim = "Ali"
soyisim = "Veli"

mesaj = f"Benim adım {isim} _____________"
print(mesaj) # Çıktı: Benim adım Ali Veli
```

4. Aşağıdaki kod bloğunun çıktısı nedir?

```
kelime = "kalem"
print(kelime.replace("a", "e"))
```

5. Aşağıdaki kod bloğunun çıktısı nedir?

```
cümle = "Bu bir cümledir."
print(cümle.split()[2])
```

6. Aşağıdaki kod bloğunun çıktısı nedir?

```
kelime = "Bilgisayar"
print(kelime[3:6])
```

7. Aşağıdaki kod bloğunun çıktısı nedir?

```
kelime = "Programlama"
print(kelime[:-4])
```

8. Aşağıdaki kod bloğunun çıktısı nedir?

```
mesaj = "Python"

for i in range(len(mesaj)-1, -1, -1):
    print(mesaj[i], end="")
```

9. Aşağıdaki kod bloğunun çıktısı nedir?

```
mesaj = "Python Öğreniyorum"

if "Python" in mesaj:
    print("Evet")
else:
    print("Hayır")
```

10. Aşağıdaki kod bloğunun çıktısı nedir?

```
metin = "  Merhaba Dünya!  "
yeni_metin = metin.strip().replace("Dünya", "Python").upper()
print(yeni_metin)
```

11. Aşağıdaki kod bloğunda boşluğa hangi ifade gelmelidir?

```
kelime = "Programlama"
print(kelime[2:_______:-1])  # Çıktı: arg
```

12. Aşağıdaki çıktıyı veren Python kodunu yazın:

```
y
h
t
o
n
```

14. Kullanıcıdan bir string girdisi alın ve bu stringin palindrom olup olmadığını kontrol eden bir program yazın. (Palindrom, tersten okunduğunda da aynı olan kelime veya cümlelerdir.)