py [coders]

KARAKTER DIZILERI

2019 = {“HAFTA”:4}

Karakter Dizileri

1

```
>>> "Merhaba Zalim Dünya!"
```

Önceki slaytta dikkat ettiyseniz “Merhaba Zalim Dünya!” adlı karakter dizisini tırnak içinde gösterdik. Bu da çok önemli bir bilgidir. Eğer bu cümleyi tırnak içine almazsak programımız hata verecektir:

```
>>> Merhaba Zalim Dünya!

  File "<stdin>", line 1
    Merhaba Zalim Dünya!
                  ^
SyntaxError: invalid syntax
```

Karakter Dizileri

1

Mesela şu, içi boş bir karakter dizisidir:

```
>>> ""
```

Şu da içinde bir adet boşluk karakteri barındıran bir karakter dizisi...

```
>>> " "
```

Bu ikisi arasındaki farka dikkat ediyoruz: Python'da 'boş karakter dizisi' ve 'bir adet boşluktan oluşan karakter dizisi' birbirlerinden farklı iki kavramdır.

Karakter Dizileri

1

Eğer herhangi bir verinin karakter dizisi olup olmadığı konusunda tereddütünüz varsa, type() adlı bir fonksiyondan yararlanarak o verinin tipini sorgulayabilirsiniz. Bu fonksiyonu şöyle kullanıyoruz:

```
>>> type("Elma")

<class 'str'>
```

Karakter Dizileri

1

Esasında, henüz bilgimiz kısıtlı da olsa karakter dizileriyle yine de ufak tefek bazı şeyler yapamayacak durumda değiliz. Mesela şu anki bilgilerimizi ve görür görmez size tanıdık gelecek bazı basit parçaları kullanarak, karakter dizilerini birbirleriyle birleştirebiliriz:

```
>>> "istihza" + ".com"

'istihza.com'
```

```
>>> "Fırat" + "Özgül"

'FıratÖzgül'
```

```
>>> "Fırat" + " " + "Özgül"

'Fırat Özgül'
```

Karakter Dizileri

1

+ işareti dışında karakter dizileri ile birlikte * (çarpı) işaretini de kullanabiliriz.
O zaman şöyle
bir etki elde ederiz:

```
>>> "w" * 3

'www'

>>> "yavaş " * 2

'yavaş yavaş '

>>> "-" * 10

'----------'

>>> "uzak" + " " * 5 + "çok uzak..."

'uzak     çok uzak...'
```

Karakter Dizileri Metodları

1

replace()

Türkçede ‘değiştirmek, yerine koymak’ gibi anlamlar taşır. Yani bu metodu kullanarak bir karakter dizisi içindeki karakterleri başka karakterlerle değiştirebileceğiz.

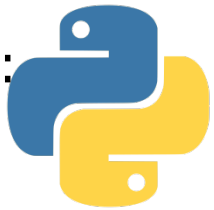
```
>>>kardiz = ”elma”

>>>kardiz.replace(”e” , ”E”)
```

2

split()

Bu metodun görevi karakter dizilerini belli noktalardan bölmektir. Zaten split kelimesi Türkçede ‘bölmek, ayırmak’ gibi anlamlara gelir. İşte bu metot, üzerine uygulandığı karakter dizilerini parçalarına ayırır. Örneğin:



Karakter Dizileri Metodları

1 split()

```
>>>kardiz = "İstanbul Büyükşehir Belediyesi”

>>> kardiz.split()
```

Ya da;

```
>>> kardiz = "Bolvadin, Kilis, Siverek, İskenderun,
İstanbul"

>>> kardiz = kardiz.split(",")

>>> print(kardiz)
```

Karakter Dizileri Metodları

1 join()

join() metodunun görevi bölünmüş karakter dizisi gruplarını birleştirmektir. Bu metot görevini yerine getirirken, yani karakter dizisi gruplarını birleştirirken bir birleştirme karakterine ihtiyaç duyar. Bizim örneğimizde bu birleştirme karakteri bir adet boşluktur. Durumu daha iyi anlayabilmek için örneğimizi incelleyelim:

```
>>> kardiz = "Beşiktaş Jimnastik Kulübü"
>>> bölünmüş = kardiz.split()
>>> print(bölünmüş)
['Beşiktaş', 'Jimnastik', 'Kulübü']

>>> kardiz = " ".join(bölünmüş)
>>> print(kardiz)
Beşiktaş Jimnastik Kulübü
```

Karakter Dizileri Metodları

1 join()

```
>>> birleştirme_karakteri = " "
>>> birleştirme_karakteri.join(bölünmüş)
```

Burada da tıpkı öteki metotlarda olduğu gibi, join() metodunu bir karakter dizisi üzerine uyguladık. Bu karakter dizisi bir adet boşluk karakteri. Ayrıca gördüğünüz gibi join() metodu bir adet de parametre alıyor.

Karakter Dizileri Metodları

1 lower()

lower() metodu, karakter dizisindeki bütün harfleri küçük harfe çeviriyor. Örneğin:

```
>>> kardiz = "ELMA"
>>> kardiz.lower() 'elma'

>>> kardiz = "arMuT"
>>> kardiz.lower() 'armut'

>>> kardiz = "PYTHON PROGRAMLAMA"
>>> kardiz.lower() 'python programlama'
```

upper()

upper() metodu, karakter dizisindeki bütür harfleri büyükharfe çeviriyor. Örneğin:

```
>>> kardiz = "istanbul"
>>> kardiz.upper() 'ISTANBUL'
```

```
iller = "istanbul, izmir, siirt, mersin"
iller = iller.replace("i", "İ").upper()
print(iller)
```

Karakter Dizileri Metodları

1 islower()

Bildiğiniz gibi, lower() metodu bir karakter dizisini tamamen küçük harflerden oluşacak şekle getiriyordu. islower() metodu ise bir karakter dizisinin tamamen küçük harflerden oluşup oluşmadığını sorguluyor.

```
>>> kardiz = "istihza"
>>> kardiz.islower() True

>>> kardiz = "Ankara"
>>> kardiz.islower() False
'
```

isupper()

isupper() metodu da islower() metodunun yaptığ işin tam tersini yapar. Bildiğiniz gibi, upper() metodu bir karakter dizisini tamamen büyük harflerden oluşacak şekle getiriyordu. isupper() metodu ise bir karakter dizisinin tamamen büyük harflerden oluşup oluşmadığını sorguluyor:

```
>>> kardiz = "İSTİHZA"
>>> kardiz.isupper()
True

>>> kardiz = "python"
>>> kardiz.isupper()
False
```

Karakter Dizileri Metodları

1 endswith()

Bu metot yardımıyla bir karakter dizisinin hangi karakter dizisi ile bittiğini sorgulayabiliyoruz. Yani örneğin:

```
>>> kardiz = "istihza"
>>> kardiz.endswith("a")
True
>>> kardiz.endswith("z")
 False
```

startswith()

endswith() metodunun yaptığı işin tam tersini yapar.

startswith() metodu ise bir karakter dizisinin hangi karakter veya karakterlerle başladığını denetler:

```
>>> kardiz = "python"
>>> kardiz.startswith("p")
True
>>> kardiz.startswith("a")
False
```

Karakter Dizileri Metodları

1 capitalize()

Şimdi göreceğimiz capitalize() metodu da upper() ve lower() metotlarına benzemekle birlikte onlardan biraz daha farklı davranır: capitalize() metodunun görevi karakter dizilerinin yalnızca ilk harfini büyütmektir. Örneğin:

```
>>> a = "python"
>>> a.capitalize()
'Python'

>>> a = "python programlama dili"
>>> a.capitalize()
'Python programlama dili'
```

Karakter Dizileri Metodları

1 title()

Bu metot biraz önce öğrendiğimiz capitalize() metoduna benzer. Bildiğiniz gibi capitalize() metodu bir karakter dizisinin yalnızca ilk harfini büyütüyordu. title() metodu da karakter dizilerinin ilk harfini büyütür. Ama capitalize() metodundan farklı olarak bu metot, birden fazla kelimeden oluşan karakter dizilerinin her kelimesinin ilk harflerini büyütür.

```
>>> a = "python programlama dili"
>>> a.capitalize()
'Python programlama dili'
>>> a.title()
'Python Programlama Dili'
```

Karakter Dizileri Metodları

1

swapcase()

swapcase() metodu da büyük-küçük harfle ilgili bir metottur.
Bu metot bir karakter dizisi içindeki büyük harfleri küçük harfe;
küçük harfleri de büyük harfe dönüştürür. Örneğin:

```
>>> kardiz = "python"
>>> kardiz.swapcase()
'PYTHON'

>>> kardiz = "PYTHON"
>>> kardiz.swapcase()
'python'

>>> kardiz = "Python”
 >>> kardiz.swapcase()
'pYTHON'
```

Karakter Dizileri Metodları

1 casefold()

Bu metot işlev olarak lower() metoduna çok benzer. Hatta Türkçe açısından, bu metodun lower() metodundan hiçbir farkı yoktur. Ancak bazı başka dillerde, bu metot bazı harfler için lower() metodunun verdiğinden farklı bir çıktı verir. Örneğin Almancadaki ‘ß’ harfi bu duruma bir örnek olabilir:

```
>>> "ß".lower() 'ß' >>> "ß".casefold() 'ss'
```

Karakter Dizileri Metodları

1 strip()

```
>>> kardiz = " istihza "
>>> print(kardiz)
' istihza '

>>> kardiz.strip()
'istihza'
```

Gördüğünüz gibi, strip() metodunu kullanarak, karakter dizisinin orijinalinde bulunan sağlı sollu boşluk karakterlerini bir çırpıda ortadan kaldırdık.
strip() metodu yukarıdaki örnekte olduğu gibi parametresiz olarak kullanıldığında, bir karakter dizisinin sağında veya solunda bulunan belli başlı karakterleri kırpar. strip() metodunun öntanımlı olarak kırptığı karakterleri diğer slaytta inceleyeceğiz.

Karakter Dizileri Metodları

1 strip()

‘ ‘	boşluk karakteri
\t	sekme (TAB) oluşturan kaçış dizisi
\n	satır başına geçiren kaçış dizisi
\r	imleci aynı satırın başına döndüren kaçış dizisi
\v	düşey sekme oluşturan kaçış dizisi
\f	yeni bir sayfaya geçiren kaçış dizisi

Yani eğer strip() metoduna herhangi bir parametre vermezsek bu metot otomatik olarak karakter dizilerinin sağında ve solunda bulunan yukarıdaki karakterleri kırpacaktır. Ancak eğer biz istersek strip() metoduna bir parametre vererek bu metodun istediğimiz herhangi başka bir karakteri kırpmasını da sağlayabiliriz.

```
>>> kardiz = "python"
>>> kardiz.strip("p")
```

Karakter Dizileri Metodları

1 **lstrip()/rstrip()**

lstrip() metodu bir karakter dizisinin sol tarafındaki gereksiz karakterlerden kurtulmamızı sağlar. Mesela bu bilgiyi yukarıdaki örneğe uygulayalım:

```
>>> "kazak".lstrip("k")
'azak'
```

2 Gördüğünüz gibi, lstrip() metodu yalnızca sol baştaki “k” harfiyle ilgilendi. Sağ taraftaki “k” harfine ise dokunmadı. Eğer sol taraftaki karakteri değil de yalnızca sağ taraftaki karakteri uçurmak istemeniz halinde ise rstrip() metodundan yararlanacaksınız:

```
>>> "kazak".rstrip("k")
'kaza'
```

Karakter Dizileri Metodları

1 count()

Tıpkı daha önce öğrendiğimiz sorgulayıcı metotlar gibi, count() metodu da bir karakter dizisi üzerinde herhangi bir değişiklik yapmamızı sağlamaz. Bu metodun görevi bir karakter dizisi içinde belli bir karakterin kaç kez geçtiğini sorgulamaktır. Bununla ilgili hemen bir örnek verelim:

```
>>> şehir = "Kahramanmaraş”
>>> şehir.count("a")
5
kelime = input("Herhangi bir kelime: ")
for harf in kelime:
        print("{} harfi {} kelimesinde {} kez
geçiyor!".format(harf,
                                        kelime,
                kelime.count(harf)))
```

Karakter Dizileri Metodları

1 **index(), rindex()**

Karakterlerin, bir karakter dizisi içinde hangi sırada bulunduğunu öğrenmek için index() adlı bir metottan yararlanabiliriz. Örneğin:

```
>>> kardiz = "python"
>>> kardiz.index("p")
0
>>> kardiz.index("n")
5
```

2 Python'ın, karakter dizisini soldan sağa doğru değil de, sağdan sola doğru okumasını da sağlayabilirsiniz. Bu iş için rindex() adlı bir metottan yararlanacağız.

```
>>> kardiz.rindex("a")
4
```

Karakter Dizileri Metodları

1

find, rfind()

find() ve rfind() metotları tamamen index() ve rindex() metotlarına benzer. find() ve rfind() metotlarının görevi de bir karakter dizisi içindeki bir karakterin konumunu sorgulamaktır:

```
>>> kardiz = "adana"

>>> kardiz.find("a")
0

>>> kardiz.rfind("a")
4
```

Karakter Dizileri Metodları

1 center()

Center kelimesi İngilizce'de 'orta, merkez, ortalamak' gibi anlamlara gelir. Bu anlama uygun olarak, center() metodunu karakter dizilerini ortalamak için kullanabilirsiniz. Örneğin:

```
for metot in dir(""):
    print(metot.center(15))
```

Gördüğünüz gibi center() metodu bir adet parametre alıyor. Bu parametre, karakter dizisine uygulanacak ortalama işleminin genişliğini gösteriyor. Bu parametrenin nasıl bir etki ortaya çıkardığını daha iyi anlayabilmek için isterseniz bir iki basit örnek verelim:

```
>>> kardiz = "python"

>>> kardiz.center(1)

'python'
```

Karakter Dizileri Metodları

1 rjust(), ljust()

Bu metotlar da tıpkı bir önceki center() metodu gibi karakter dizilerini hizalama vazifesi görür. rjust() metodu bir karakter dizisini sağa yaslarken, ljust() metodu karakter dizisini sola yaslar. Mesela şu iki kod parçasının çıktılarını inceleyin:

```
>>> for i in dir(""):
...             print(i.ljust(20))

>>> for i in dir(""):
...             print(i.rjust(20))

>>> kardiz = "tel no"
>>> kardiz.ljust(10, ".")

'tel no....'
```