String Methodları

capitalize()	find()	isdecimal()	istitle()	partition()
capitalize()	format()	isdigit()	isupper()	replace()
center()	format_map()	isidentifier()	join()	rfind()
count()	index()	islower()	ljust()	rindex()
encode()	isalnum()	isnumeric()	lower()	rjust()
endswith()	isalpha()	isprintable()	lstrip()	rpartition()
expandtabs()	isascii()	isspace()	maketrans()	rsplit()
	zfill()	title()	startswith()	rstrip()
		translate()	strip()	split()
	zfill()	upper()	swapcase()	splitlines()

```
capitalize()
```

```
var = "mustafa'ın bugün dersi var."
x = var.capitalize()
print(x)
```

```
Mustafa'ın bugün dersi var.
```

- **capitalize()** metodunda cümlenin sadece baş harfi büyük yazılır; geri kalan ifadeler küçüktür.

```
casefold()
```

```
]: var = "Mustafa'ın Bugün Dersi VAr."
x = var.casefold()
print(x)
```

```
mustafa'ın bugün dersi var.
```

- **casefold()** metodunda cümlenin bütün kelimeleri küçük harfle yazılır (sadece baş harfi değil.)

```
string.center(length, character)
```

```
: var = "Mustafa"
x = var.center(20,"+")
print(x)
```

```
++++++Mustafa+++++++
```

- **center(uzunluk, karakter)** metodu ile karakteri soldan ve sağdan ortalamak için kullanılır. yukarıda örnekte 20 yazılmış ve 20 indexlik ifade içerisinde karakter oturtulmuş. “+” yazılarak da boş kalan yerler doldurulmuş.

```
string.count(value, start, end)
```

```
txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple", 10, 24)

print(x)
```

```
1
```

- **count()** metodu da str ifade içerisinde aranan ifadenin kaç tane olup olmadığını kontrol etmemizi sağlıyor. start ve end yazılmazsa tüm ifade içerisinde arama yapar.

```
string.endswith(value, start, end)
```

```
txt = "Hello, welcome to my world."

x = txt.endswith("my world.", 5, 11)

print(x)
```

```
False
```

- Dize belirtilen değerle sona ererse, yöntem True döndürür, aksi takdirde False.

```
string.expandtabs(tabsize)
```

```
var = "Mustafa\t Altaş"
print(var.expandtabs(20))
```

```
Mustafa              Altaş
```

- expantabs() metodu \t kaçış dizinin boşluk miktarını belirler.

```
string.find(value, start, end)
```

```
txt = "Hello, welcome to my world."
x = txt.find("o")
print(x)
```

```
4
```

- find() metodu ile aranana ifadenin veya karakterin soldan ilk yakaladığı index sayısını döndürür. eğer aranana ifade veya karakter yoksa -1 döndürür.

```
txt = "For only {price:.2f} dollars!"
print(txt.format(price = 49))
```

```
For only 49.00 dollars!
```

```
string.index(value, start, end)
```

```
txt = "Hello, welcome to my world."

x = txt.index("e")

print(x)
```

```
1
```

- index() metodunda eğer aranan ifade yoksa hata mesajı verecektir. ve aranan ifade soldan itibaren kontrol edilir.

```
string.isalnum()
```

```
var = "mustafa12"
x = var.isalnum()
print(x)
```

```
True
```

- isalnum() methodunda str ifade içerisinde alfabatik veya numaratik ifadeler varsa (ikisi de aynı anda olabilir) o zaman True döndürür.

```
string.isalpha()
Tüm karakterler alfabe harfleri (a-z) ise yöntem True döndürür.isalpha()
```

```
var = "mustafa12"
x = var.isalpha()
print(x)
```

```
False
```

- Tüm karakterler alfabe harfleri (a-z) ise yöntem True döndürür.isalpha()

```
var = "mustafa12"
x = var.isascii()
print(x)
```

```
True
```

- string.isascii()
  Tüm karakterler ascii karakterler (a-z) ise yöntem True döndürür.isascii()

```
string.isdecimal()
Tüm karakterler ondalık (0-9) ise yöntem True döndürür.isdecimal()
```

```
var = "12"
x = var.isdecimal()
print(x)
```

```
True
```

- string.isdecimal()
  Tüm karakterler ondalık (0-9) ise yöntem True döndürür.isdecimal()

```
string.isdigit()
Tüm karakterler ondalık (0-9) ise yöntem True döndürür.isdecimal()
```

```
a = "\u0030" #unicode for 0
b = "\u00B2" #unicode for ²

print(a.isdigit())
print(b.isdigit())
```

```
True
True
```

```
string.isidentifier()
Dize geçerli bir tanımlayıcıysa, yöntem True döndürür, aksi takdirde False.isidentifier()

Dize yalnızca alfasayısal harfler (a-z) ve (0-9) veya alt çizgiler (_) içeriyorsa geçerli bir tanımlayıcı olarak kabul edilir.
Geçerli bir tanımlayıcı bir sayıyla başlayamaz veya boşluk içeremez.
```

```
a = "MyFolder"
b = "Demo002"
c = "2bring"
d = "my demo"

print(a.isidentifier())
print(b.isidentifier())
print(c.isidentifier())
print(d.isidentifier())
```

```
True
True
False
False
```

```
string.islower()
Tüm karakterler küçük harfle, aksi takdirde False ise yöntem True döndürür.islower()

Sayılar, semboller ve boşluklar denetlenmez, yalnızca alfabe karakterleri denetlenmez.
```

```
d = "my demo"
d.islower()
```

```
True
```

```
string.isnumeric()
Tanım ve Kullanım
Tüm karakterler sayısal (0-9), aksi takdirde False ise yöntem True döndürür.isnumeric()

² ve 3/4 gibi üsler de sayısal değerler olarak kabul edilir.

"-1" ve sayısal değerler olarak kabul edİlMEZ, çünkü dizedeki tüm karakterler sayısal olmalıdır ve ve değildir."1.5"-.
```

```
a = "\u0030" #unicode for 0
b = "\u00B2" #unicode for &sup2;
c = "10km2"
d = "-1"
e = "1.5"

print(a.isnumeric())
print(b.isnumeric())
print(c.isnumeric())
print(d.isnumeric())
print(e.isnumeric())
```

```
True
True
False
False
False
```

```
Tüm karakterler yazdırılabilirse yöntem True döndürür, aksi takdirde False.isprintable()

Yazdırılabilir karakter örneği satır başı ve satır beslemesi olabilir.
```

```
txt = "Hello!\nAre you #1?"

x = txt.isprintable()

print(x)
```

```
False
```

```
Bir dizedeki tüm karakterler boşluklar ise, aksi takdirde False, yöntem True döndürür.isspace()
```

```
txt = "    "

x = txt.isspace()

print(x)
```

```
True
```

```
string.istitle()


Bir metindeki tüm sözcükler büyük harfle başlıyorsa ve sözcüüğün geri kalanı küçük harfler ise, aksi takdirde False ise yöntem
True döndürür.istitle()

Semboller ve sayılar yoksayılır.
```

:
```
a = "HELLO, AND WELCOME TO MY WORLD"
b = "Hello"
c = "22 Names"
d = "This Is %'!?"

print(a.istitle())
print(b.istitle())
print(c.istitle())
print(d.istitle())
```

```
False
True
True
True
```

```
string.isupper()
Bir metindeki tüm sözcükler büyük harfle başlıyorsa ve sözcüüğün geri kalanı küçük harfler ise, aksi takdirde False ise yöntem
True döndürür.istitle()

Semboller ve sayılar yoksayılır.
```

```
a = "Hello World!"
b = "hello 123"
c = "MY NAME IS PETER"

print(a.isupper())
print(b.isupper())
print(c.isupper())
```

```
False
False
True
```

```
string.join(iterable)
Yöntem, bir yinelemedeki tüm öğeleri alır ve bunları tek bir dizede birleştirir.join()

Ayırıcı olarak bir dize belirtilmelidir.
```

```
myTuple = ("John", "Peter", "Vicky")

x = "#".join(myTuple)

print(x)
```

```
John#Peter#Vicky
```

```
x = "Mustafa"
"$".join(x)
```

```
'M$u$s$t$a$f$a'
```

```
myDict = {"name": "John", "country": "Norway"}
mySeparator = "TEST"

x = mySeparator.join(myDict)

print(x)
```

```
nameTESTcountry
```

```
string.ljust(length, character)
Yöntem, dolgu karakteri olarak belirtilen bir karakter (boşluk varsayılandır) kullanarak dizeyi sola hizalar.ljust()
```

```
txt = "banana"

x = txt.ljust(20, "O")

print(x)
```

```
banana00000000000000
```

```
string.lower()
Yöntem, tüm karakterlerin küçük harf olduğu bir dize döndürür.lower()

Semboller ve Sayılar yoksayılır.
```

```
txt = "Hello my FRIENDS"

x = txt.lower()

print(x)
```

```
hello my friends
```

```
string.lstrip(characters)
Yöntem önde gelen karakterleri kaldırır (boşluk kaldırılacak varsayılan baştaki karakterdir)lstrip()
```

```
txt = ",,,,,ssaaww.....banana"

x = txt.lstrip(",.asw")

print(x)
```

```
banana
```

```
string.maketrans(x, y, z)

x = Gerekli. Yalnızca bir parametre belirtilirse, bu, değiştirmenin nasıl gerçekleştirileceğini açıklayan bir sözlük olmalıdır. İki veya daha fazla parametre belirtilirse, bu parametrenin değiştirmek istediğiniz karakterleri belirten bir dize olması gerekir.

y = İsteğe bağlı. x parametresiyle aynı uzunlukta bir dize. İlk parametredeki her karakter, bu dizideki karşılık gelen karakterle değiştirilecektir.

z = İsteğe bağlı. Orijinal dizeden hangi karakterlerin kaldırılacağını açıklayan bir dize.
```

```
txt = "Good night Sam!"
x = "mSa"
y = "eJo"
z = "odnght"
mytable = txt.maketrans(x, y, z)
print(txt.translate(mytable))
```

```
G i Joe!
```

```
string.partition(value)
Yöntem belirtilen bir dizeyi arar ve dizeyi üç öğe içeren bir diziye böler.partition()

İlk öğe belirtilen dizeden önceki bölümü içerir.

İkinci öğe belirtilen dizeyi içerir.

Üçüncü öğe dizeden sonraki bölümü içerir.
```

```
txt = "I could eat bananas all day"

x = txt.partition("bananas")

print(x)
```

```
('I could eat ', 'bananas', ' all day')
```

```
string.replace(oldvalue, newvalue, count)
Yöntem, belirtilen bir tümceciği belirtilen başka bir tümcecikle değiştirir.replace()
```

```
txt = "one one was a race horse, two two was one too."

x = txt.replace("one", "three", 2)

print(x)
```

```
three three was a race horse, two two was one too.
```

```
string.rindex(value, start, end)ü
Yöntem, belirtilen değerin son örneğini bulur.rindex()

Değer bulunamazsa yöntem bir özel durum oluşturur.rindex()
```

```
txt = "Hello, welcome to my world."

print(txt.rfind("q"))
print(txt.rindex("q"))
```

```
-1

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-85-3fbde3bff713> in <module>
      2
      3 print(txt.rfind("q"))
----> 4 print(txt.rindex("q"))

ValueError: substring not found
```

```
string.rjust(length, character)
Yöntem, dolgu karakteri olarak belirtilen bir karakter (boşluk varsayılandır) kullanarak dizeyi sağa hizalar.rjust()
```

```
txt = "banana"

x = txt.rjust(20, "O")

print(x)
```

```
OOOOOOOOOOOOOObanana
```

```
txt = "banana"

x = txt.rjust(20)

print(x)
```

```
              banana
```

```
string.rpartition(value)
Yöntem, belirtilen dizenin son oluşumunu arar ve dizeyi üç öğe içeren bir diziye böler.rpartition()
```

```
txt = "I could eat bananas all day, bananas are my favorite fruit"

x = txt.rpartition("bananas")

print(x)
```

```
('I could eat bananas all day, ', 'bananas', ' are my favorite fruit')
```

```
txt = "I could eat bananas all day, bananas are my favorite fruit"

x = txt.rpartition("apples")

print(x)
```

```
('', '', 'I could eat bananas all day, bananas are my favorite fruit')
```

```
string.swapcase()
Yöntem, tüm büyük harflerin küçük harf olduğu ve bunun tersi de olsa bir dize döndürür.swapcase()
```

In [92]:
```
txt = "Hello My Name Is PETER"

x = txt.swapcase()

print(x)
```

```
hELLO mY nAME iS peter
```

```
string.split(separator, maxsplit)
```

```
txt = "apple#banana#cherry#orange"

x = txt.split("#")

print(x)
```

```
['apple', 'banana', 'cherry', 'orange']
```

```
string.splitlines(keeplinebreaks)
Yöntem bir dizeyi listeye böler. Bölme satır sonlarında yapılır.splitlines()
```

```
txt = "Thank you for the music\nWelcome to the jungle"

x = txt.splitlines()

print(x)
```

```
['Thank you for the music', 'Welcome to the jungle']
```

```
string.startswith(value, start, end)
Dize belirtilen değerle başlarsa, yöntem True döndürür, aksi takdirde False.startswith()
```

```
txt = "Hello, welcome to my world."

x = txt.startswith("Hello")

print(x)
```

```
True
```

```
string.strip(characters)
Yöntem, baştaki boşlukları ve sondaki (sonundaki boşluklar) karakterleri kaldırır (boşluk, kaldırılacak varsayılan baştaki karakterdir)strip()
```

```
txt = ",,,,,rrttgg.....banana....rrr"

x = txt.strip(",.grt")

print(x)
```

```
banana
```

```
txt = "     banana     "

x = txt.strip()

print("of all fruits", x, "is my favorite")
```

```
of all fruits banana is my favorite
```

```
string.title()
Yöntem, her sözcükteki ilk karakterin büyük harf olduğu bir dize döndürür. Başlık veya başlık gibi.title()
```

```
txt = "Welcome to my 2nd world"

x = txt.title()

print(x)
```

```
Welcome To My 2Nd World
```

```
string.translate(table)
Yöntem, belirtilen bazı karakterlerin sözlükte veya eşleme tablosunda açıklanan karakterle değiştirildiği bir dize
döndürür.translate()

Eşleme tablosu oluşturmak için yöntemi kullanın. maketrans()

Sözlükte/tabloda bir karakter belirtilmezse, karakter değiştirilmez.

Sözlük kullanıyorsanız, karakterler yerine ascii kodları kullanmanız gerekir.
```

```
#use a dictionary with ascii codes to replace 83 (S) with 80 (P):
mydict = {83:  80}
txt = "Hello Sam!"
print(txt.translate(mydict))
```

```
Hello Pam!
```

```
string.upper()
Yöntem, tüm karakterlerin büyük harfle olduğu bir dize döndürür.upper()

Semboller ve Sayılar yoksayılır.
```

```
txt = "Hello my friends"

x = txt.upper()

print(x)
```

```
HELLO MY FRIENDS
```

```
string.zfill(len)
Yöntem, belirtilen uzunluğa ulaşana kadar dizenin başına sıfırlar (0) ekler.zfill()

Len parametresinin değeri dizenin uzunluğundan küçükse, dolgu yapılmaz.
```

```
a = "hello"
b = "welcome to the jungle"
c = "10.000"

print(a.zfill(10))
print(b.zfill(10))
print(c.zfill(10))
```

```
00000hello
welcome to the jungle
000010.000
```