Python karakterláncok

Ebben az oktatóanyagban megtanulja létrehozni, formázni, módosítani és törölni a karakterláncokat a Pythonban. Ezenkívül megismerkedik a különféle string műveletekkel és funkciókkal.

Mi a karakterlánc a Pythonban?

A karakterlánc egy karaktersorozat.

A karakter egyszerűen szimbólum. Például az angol nyelv 26 karakterből áll.

A számítógépek nem karakterekkel, hanem számokkal (bináris) foglalkoznak. Annak ellenére, hogy karaktereket láthat a képernyőn, belsőleg 0-k és 1-ek kombinációjaként tárolják és kezelik.

Ezt a karakter számgá alakítását kódolásnak nevezik, a fordított folyamat pedig a dekódolás. Az ASCII és az Unicode a népszerű kódolások közül néhány.

A Pythonban a karakterlánc Unicode karakterek sorozata. Az Unicode-ot azért vezették be, hogy minden karaktert minden nyelven tartalmazzon, és egységessé tegye a kódolást. Megismerheti az Unicode-ot a Python Unicode-ból .

Hogyan hozzunk létre egy karakterláncot a Pythonban?

A karakterláncok úgy hozhatók létre, hogy karaktereket zárnak be egyetlen idézetbe vagy dupla idézőjelbe. Még hármas idézetek is használhatók a Pythonban, de általában többsoros húrok és docstringek ábrázolására használhatók.

```
# defining strings in Python
# all of the following are equivalent
my_string = 'Hello'
print(my_string)

my_string = "Hello"
print(my_string)

my_string = '''Hello'''
print(my_string)

# triple quotes string can extend multiple lines
my_string = """Hello, welcome to
           the world of Python"""
```

```
print(my_string)
```

A program futtatásakor a kimenet a következő lesz:

```
Helló
Helló
Helló
Helló, üdvözlöm a
           a Python világa
```

Hogyan férhetünk hozzá a karakterláncokhoz?

Az egyes karakterekhez indexeléssel, a karakterek tartományához pedig szeleteléssel férhetünk hozzá. Az index 0-tól kezdődik. Ha megpróbál elérni egy karaktert, amely az index tartományán kívül esik, akkor egy `IndexError`. Az indexnek egész számnak kell lennie. Nem használhatunk úszókat vagy más típusokat, ez eredményez `TypeError`.
A Python lehetővé teszi negatív indexelését szekvenciáihoz.
Az index `-1`az utolsó elemre, `-2`a második utolsó elemre és így tovább. A szeletelő operátor `:`(kettőspont) használatával elérhetünk egy sztringben lévő elemek egy részét .

```
#Accessing string characters in Python
str = 'programiz'
print('str = ', str)

#first character
print('str[0] = ', str[0])

#last character
print('str[-1] = ', str[-1])

#slicing 2nd to 5th character
print('str[1:5] = ', str[1:5])

#slicing 6th to 2nd last character
print('str[5:-2] = ', str[5:-2])
```

A fenti program futtatásakor a következő kimenetet kapjuk:

```
str = programiz
str [0] = p
```

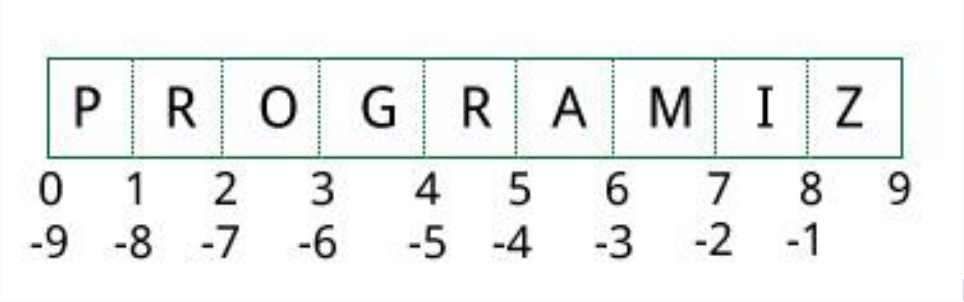
```
str [-1] = z
str [1: 5] = rogr
str [5: -2] = am
```

Ha megpróbálunk elérni egy tartományon kívüli indexet, vagy egész számtól eltérő számokat használunk, hibákat kapunk.

```
# index must be in range
>>> my_string[15]
...
IndexError: string index out of range

# index must be an integer
>>> my_string[1.5]
...
TypeError: string indices must be integers
```

A szeletelés legjobban úgy vizualizálható, ha az indexet az alábbiak szerint az elemek között tartjuk.
Ha egy tartományhoz szeretnénk hozzáférni, szükségünk van arra az indexre, amely szeleteli a karaktersorozat részét.



Karakterlánc szeletelés a Pythonban

Hogyan módosítsunk vagy töröljünk egy karakterláncot?

A húrok megváltoztathatatlanok. Ez azt jelenti, hogy a karakterlánc elemeit nem lehet megváltoztatni, miután hozzárendelték őket. Egyszerűen átrendezhetjük a különböző húrokat ugyanarra a névre.

```
>>> my_string = 'programiz'
>>> my_string[5] = 'a'
...
TypeError: 'str' object does not support item assignment
>>> my_string = 'Python'
>>> my_string
```

```
'Python'
```

Nem törölhetünk és nem távolíthatunk el karaktereket egy karakterláncból. De a karakterlánc teljes törlése a `del`kulcsszó használatával lehetséges .

```
>>> del my_string[1]
...
TypeError: 'str' object doesn't support item deletion
>>> del my_string
>>> my_string
...
NameError: name 'my_string' is not defined
```

Python karakterlánc műveletek

Számos művelet hajtható végre karakterláncokkal, ami a Python egyik leggyakrabban használt adattípusává teszi.

További információ a Pythonban elérhető adattípusokról: Python adattípusok

Két vagy több húr összefűzése

Két vagy több húr egyetlen eggyé egyesítését összefűzésnek nevezzük.

A **+** operátor ezt Pythonban teszi. Egyszerűen két húrliterál együttes megírása összefűzi őket.

A ***** operátor segítségével a karakterlánc adott számú alkalommal megismételhető.

```
# Python String Operations
str1 = 'Hello'
str2 ='World!'

# using +
print('str1 + str2 = ', str1 + str2)

# using *
print('str1 * 3 =', str1 * 3)
```

A fenti program futtatásakor a következő kimenetet kapjuk:

```
str1 + str2 = HelloWorld!
str1 * 3 = HelloHelloHello
```

Két karakterlánc literál együttes megírása szintén összefűzi őket, mint a **+** operátor.

Ha különböző vonalakban szeretnénk összefűzni a húrokat, akkor zárójeleket használhatunk.

```
>>> # two string literals together
>>> 'Hello ''World!'
'Hello World!'

>>> # using parentheses
>>> s = ('Hello '
...      'World')
>>> s
'Hello World'
```

Egy húron keresztül iterál

Iterálhatunk egy stringen keresztül egy for ciklus segítségével . Itt egy példa az 'l' számának megszámolására egy karakterláncban.

```
# Iterating through a string
count = 0
for letter in 'Hello World':
    if(letter == 'l'):
        count += 1
print(count,'letters found')
```

A fenti program futtatásakor a következő kimenetet kapjuk:

```
3 levél található
```

Vonós tagsági teszt

A kulcsszó segítségével tesztelhetjük, hogy létezik-e egy sztring egy sztringben, vagy sem `in`.

```
>>> 'a' in 'program'
True
>>> 'at' not in 'battle'
False
```

Beépített függvények a Python használatához

Különböző beépített funkciók, amelyek szekvenciával működnek, húrokkal is működnek.

Néhány általánosan használt `enumerate()`és `len()`. A `enumerate()`függvény egy felsorolt objektumot ad vissza. Ez párban tartalmazza a karakterlánc összes elemének indexét és értékét. Ez hasznos lehet az iterációhoz.

Ehhez hasonlóan `len()`adja vissza a karakterlánc hosszát (karakterek számát).

```
str = 'cold'

# enumerate()
list_enumerate = list(enumerate(str))
print('list(enumerate(str) = ', list_enumerate)

#character count
print('len(str) = ', len(str))
```

A fenti program futtatásakor a következő kimenetet kapjuk:

```
lista (felsorolja (str) = [(0, 'c'), (1, 'o'), (2, 'l'), (3, 'd')]
len (str) = 4
```

Python karakterlánc-formázás

Menekülési sorrend

Ha olyan szöveget akarunk kinyomtatni, mint `Azt mondta: - Mi van ott?`, nem használhatunk sem egyetlen, sem dupla idézőjelet. Ez azt eredményezi, hogy `SyntaxError`a szöveg önmagában egyszeres és dupla idézőjeleket is tartalmaz.

```
>>> print("He said, "What's there?"")
...
SyntaxError: invalid syntax
>>> print('He said, "What's there?"')
...
SyntaxError: invalid syntax
```

A probléma kikerülésének egyik módja a hármas idézetek használata. Alternatív megoldásként használhatunk menekülési szekvenciákat.
A menekülési sorrend egy visszavágással kezdődik, és másként értelmezik. Ha egyetlen idézetet használunk egy karakterlánc ábrázolására, akkor a karakterláncon belüli összes idézetet el kell hagyni. Hasonló a helyzet a dupla idézőjelekkel is. Így teheti meg a fenti szöveg ábrázolását.

```
# using triple quotes
print('''He said, "What's there?"''')

# escaping single quotes
print('He said, "What\'s there?"')

# escaping double quotes
print("He said, \"What's there?\"")
```

A fenti program futtatásakor a következő kimenetet kapjuk:

```
Azt mondta: - Mi van ott?
Azt mondta: - Mi van ott?
Azt mondta: - Mi van ott?
```

Itt található a Python által támogatott összes menekülési szekvencia felsorolása.

Menekülési sorrend	Leírás
\új sor	A visszavonást és az új sort figyelmen kívül hagyta
\\	Visszavágás
\'	Egyetlen idézet
\"	Dupla idézet
\a	ASCII Bell
\b	ASCII Backspace
\f	ASCII Formfeed
\n	ASCII Linefeed
\r	ASCII kocsi vissza
\t	ASCII vízszintes fül

\ v	ASCII függőleges fül
\ ooo	Oktálértékű karakter ooo
\ xHH	HH hexadecimális értékű karakter

Íme néhány példa

```
>>> print("C:\\Python32\\Lib")
C:\Python32\Lib

>>> print("This is printed\nin two lines")
This is printed
in two lines

>>> print("This is \x48\x45\x58 representation")
This is HEX representation
```

Nyers karakterlánc a menekülési sorrend figyelmen kívül hagyásához

Néha figyelmen kívül hagyhatjuk a sztring belsejében található menekülési szekvenciákat. Ehhez elhelyezhetjük `r`vagy `R`a húr elé. Ez azt jelenti, hogy ez egy nyers karakterlánc, és a benne lévő minden menekülési szekvenciát figyelmen kívül hagyják.

```
>>> print("This is \x61 \ngood example")
This is a
good example
>>> print(r"This is \x61 \ngood example")
This is \x61 \ngood example
```

A format () metódus a karakterláncok formázásához

A `format()`string objektummal elérhető módszer nagyon sokoldalú és hatékony a karakterláncok formázásában. A formátum-karakterláncok göndör zárójeleket tartalmaznak `{}`helyőrzőként vagy helyettesítő mezőkként, amelyek cserélődnek. Pozíciós vagy kulcsszó argumentumokat használhatunk a sorrend megadásához.

```
# Python string format() method

# default(implicit) order
default_order = "{}, {} and {}".format('John','Bill','Sean')
print('\n--- Default Order ---')
print(default_order)

# order using positional argument
positional_order = "{1}, {0} and {2}".format('John','Bill','Sean')
print('\n--- Positional Order ---')
print(positional_order)

# order using keyword argument
keyword_order = "{s}, {b} and {j}".format(j='John',b='Bill',s='Sean')
print('\n--- Keyword Order ---')
print(keyword_order)
```

A fenti program futtatásakor a következő kimenetet kapjuk:

```
--- Alapértelmezett sorrend ---
John, Bill és Sean

--- Helyzeti sorrend ---
Bill, John és Sean

--- Kulcsszó sorrend ---
Sean, Bill és John
```

A `format()` módszer opcionális formátum specifikációval rendelkezhet. A mező nevétől kettősponttal választják el őket. Például balra igazolhatunk `<`, jobbra igazolhatunk `>` vagy középre `^` helyezhetünk egy karakterláncot az adott térben.
Az egész számokat bináris, hexadecimális stb. Formátumban is formázhatjuk, és az úszók kerekíthetők vagy exponens formátumban jeleníthetők meg. Rengeteg formázás használható. Látogasson el ide a `format()` módszerrel elérhető összes karakterlánc-formázáshoz .

```
>>> # formatting integers
>>> "Binary representation of {0} is {0:b}".format(12)
'Binary representation of 12 is 1100'

>>> # formatting floats
>>> "Exponent representation: {0:e}".format(1566.345)
'Exponent representation: 1.566345e+03'

>>> # round off
>>> "One third is: {0:.3f}".format(1/3)
'One third is: 0.333'
```

```
>>> # string alignment
>>> "|{:<10}|{:^10}|{:>10}|".format('butter','bread','ham')
'|butter    |  bread   |       ham|'
```

Régi stílusú formázás

Akár olyan karakterláncokat is formázhatunk, mint a `sprintf()`C programozási nyelvben használt régi stílus. Ennek `%`megvalósításához az operátort használjuk .

```
>>> x = 12.3456789
>>> print('The value of x is %3.2f' %x)
The value of x is 12.35
>>> print('The value of x is %3.4f' %x)
The value of x is 12.3457
```

Gyakori Python karakterlánc-módszerek

A string objektummal számos módszer áll rendelkezésre. A `format()`fent említett módszer egyike ezeknek. Néhány általánosan használt módszerek `lower()`, `upper()`, `join()`, `split()`, `find()`, `replace()`stb Itt van egy teljes listát az összes beépített módszerek munka húrok Python .

```
>>> "PrOgRaMiZ".lower()
'programiz'
>>> "PrOgRaMiZ".upper()
'PROGRAMIZ'
>>> "This will split all words into a list".split()
['This', 'will', 'split', 'all', 'words', 'into', 'a', 'list']
>>> ' '.join(['This', 'will', 'join', 'all', 'words', 'into', 'a', 'string'])
'This will join all words into a string'
>>> 'Happy New Year'.find('ew')
7
>>> 'Happy New Year'.replace('Happy','Brilliant')
'Brilliant New Year'
```

függvény	leírás
capitalize()	**Az első karaktert nagybetűssé alakítja**
casefold()	**A karakterláncot kisbetűssé alakítja**
center()	Középre adott karakterláncot ad vissza
count()	**Visszaadja a megadott érték előfordulásainak számát egy karakterláncban**
encode()	A karakterlánc kódolt változatát adja vissza
endswith()	Igaz értéket ad vissza, ha a karakterlánc a megadott értékkel végződik
expandtabs()	Beállítja a karakterlánc tabulátor méretét
find()	Megkeresi a karakterláncot egy megadott értékre, és visszaadja a megtalálás helyét
format()	A megadott értékeket formázza egy karakterláncban
format_map()	A megadott értékeket formázza egy karakterláncban
index()	**Megkeresi a karakterláncot egy megadott értékre, és visszaadja a megtalálás helyét**
isalnum()	True értéket ad vissza, ha a karakterlánc összes karaktere alfanumerikus
isalpha()	True értéket ad vissza, ha a karakterlánc összes karaktere az ábécé része
isdecimal()	True értéket ad vissza, ha a karakterlánc összes karaktere tizedes
isdigit()	True értéket ad vissza, ha a karakterláncban szereplő összes karakter számjegyű
isidentifier()	True értéket ad vissza, ha a karakterlánc azonosító
islower()	True értéket ad vissza, ha a karakterlánc összes karaktere kisbetűs
isnumeric()	True értéket ad vissza, ha a karakterlánc összes karaktere numerikus
isprintable()	True értéket ad vissza, ha a karakterlánc összes karaktere kinyomtatható
isspace()	True értéket ad vissza, ha a karakterláncban szereplő összes karakter szóköz

istitle()	True értéket ad vissza, ha a karakterlánc egy cím szabályait követi
isupper()	True értéket ad vissza, ha a karakterlánc összes karaktere nagybetű
join()	**Összekapcsolja az iterálható elemeket a karakterlánc végéig**
ljust()	A karakterlánc balra igazított változatát adja vissza
lower()	**A karakterláncot kisbetűvé alakítja**
lstrip()	Visszaadja a karakterlánc bal oldali változatát
maketrans()	Visszaadja a fordításokban használandó fordítási táblázatot
partition()	Visszaad egy duplát, ahol a karakterlánc három részre oszlik
replace()	Karakterláncot ad vissza, ahol a megadott érték helyébe egy megadott érték lép
rfind()	Megkeresi a karakterláncot egy megadott érték után, és a találat utolsó helyét adja vissza
rindex()	Megkeresi a karakterláncot egy megadott érték után, és a találat utolsó helyét adja vissza
rjust()	A karakterlánc jobbra igazolt változatát adja vissza
rpartition()	Visszaad egy duplát, ahol a karakterlánc három részre oszlik
rsplit()	Hasítja a karakterláncot a megadott elválasztóhoz, és egy listát ad vissza
rstrip()	A karakterlánc jobb oldali változatát adja vissza
split()	Hasítja a karakterláncot a megadott elválasztóhoz, és egy listát ad vissza
splitlines()	Hasítja a karakterláncot sortörésekkor, és listát ad vissza
startswith()	Igaz értéket ad vissza, ha a karakterlánc a megadott értékkel indul
strip()	A karakterlánc kivágott változatát adja vissza
swapcase()	**Nagybetűket cserél, kisbetűket nagybetűvé és fordítva**
title()	**Minden szó első karakterét nagybetűvé alakítja**
translate()	**Fordított karakterláncot ad vissza**

upper()	**A karakterláncot nagybetűvé alakítja**
zfill()	Az elején egy megadott számú 0 értékkel tölti ki a karakterláncot