Reťazce (znakové reťazce)

Reťazec je nemenná postupnosť znakov Unicode údajového typu str (skratka slova string). Hodnoty typu str sú uzavreté v úvodzovkách alebo apostrofoch (je jedno, ktoré ohraničenie použijeme; na oboch stranách reťazca musí byť rovnaké). Prázdny reťazec je reťazec, ktorý nemá medzi ohraničením ani jeden znak. K jednotlivým znakom v reťazci sa dá dostať pomocou výrazu *reťazec*[*index*] , pričom prvý znak v reťazci má index 0. Teda napr. "Python"[0] vráti jednoznakový reťazec P; "Python"[5] vráti n; "Python"[6] vráti chybové hlásenie „Index reťazca je mimo rozsah“. Ukážka:

```
>>> "Python"[0]
'P'
>>> "Python"[5]
'n'
>>> "Python"[6]
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    "Python"[6]
IndexError: string index out of range
```

Dĺžka reťazca je počet jeho znakov. Dĺžku reťazca, čo je prirodzené číslo, vracia funkcia len(*reťazec*) (názov funkcie zo slova length). Dĺžka prázdneho reťazca je nula. Ukážka:

```
>>> len("Python")
6
>>> len("")
0
```

Ukážka indexovania v reťazci s s hodnotou Python (s = "Python"):

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]
P	y	t	h	o	n
s[-6]	s[-5]	s[-4]	s[-3]	s[-2]	s[-1]

Možné a užitočné sú aj záporné indexy, index s hodnotou -1 vždy ukazuje na posledný znak v reťazci. Uvedomte si aj, že funkcia len() vracia celé číslo o 1 väčšia ako je index posledného znaku v reťazci, resp. s[len(s)-1] == s[-1].

V rámci konverzie (prevodu) možno iný údajový typ previesť na reťazec funkciou str(). Konverziu musíme použiť aj pri vytváraní výstupného reťazca („+“ v ukážke znamená spojiť dva reťazce). Ukážka:

```
>>> str(999)
'999'
>>> str(1.456)
'1.456'
>>> str(1,456)
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    str(1,456)
TypeError: str() argument 2 must be str, not int
>>> pi = 3.14159
>>> print(pi)
3.14159
>>> print("PI = ", pi)
PI =  3.14159
>>> print("PI = " + pi)
Traceback (most recent call last):
  File "<pyshell#38>", line 1, in <module>
    print("PI = " + pi)
TypeError: Can't convert 'float' object to str implicitly
>>> print("PI = " + str(pi))
PI = 3.14159
>>>
```

Reťazce sa porovnávajú operátormi ==,!=, <, <=, >, >= (porovnanie sa realizuje lexikograficky - podľa abecedy, pričom ak sa prvé znaky v porovnávaných reťazcoch rovnajú, automaticky sa porovnajú ďalšie znaky atď.), spájajú operátorom + a opakujú operátorom *, ako to vidieť v ukážke:

```
>>> "Python" == "python"
False
>>> "Python" < "python"
True
>>> "Python" < "Python "
True
>>> s = "Python"
>>> s += "python"
>>> s
'Pythonpython'
>>> s *= 3
>>> s
'PythonpythonPythonpythonPythonpython'
```

Reťazce možno rezať. Rezací operátor má tri varianty:

1. reťazec[*začiatok* :]
2. reťazec[*začiatok* : *koniec*]
3. reťazec[*začiatok* : *koniec* : *krok*]

Začiatok, *koniec* a *krok* sú buď celé čísla alebo premenné obsahujúce celé čísla; môžu sa aj nezadať.

Z ukážky nižšie je zrejmý význam jednotlivých variant. Prvý variant vracia podreťazec zo zadaného reťazca so znakmi od znaku s indexom *začiatok* až po posledný znak pôvodného reťazca. Druhý variant vracia podreťazec so znakmi od znaku s indexom *začiatok* až **pred** znak na indexe *koniec*. *Krok* v treťom variante udáva, znak s akým indexom sa má preniesť do podreťazca, napríklad číslo 2 znamená preniesť znaky s indexmi 0, 2, 4, 6,..., t.j. preniesť znaky len s tými indexmi, ktoré sú násobkami kroku. Záporný krok znamená, že budú vyberané znaky smerom k začiatku reťazca, preto `s[: : -1]` vráti obrátený reťazec.

```
>>> s = "Python je programovací jazyk"
>>> s[7:]
'je programovací jazyk'
>>> s[-5:len(s)]
'jazyk'
>>> s[-5:-1]
'jazy'
>>> s[-5:]
'jazyk'
>>> s[0:len(s):2]
'Pto epormvc ay'
>>> s[::2]
'Pto epormvc ay'
>>> s[5::-1]
'nohtyP'
>>> s[::-1]
'kyzaj ícavomargorp ej nohtyP'
```

Rezacie operátory majú ešte ďalšie možnosti, použitie napríklad pri spájaní reťazcov

```
>>> s[:9] + s[len(s)-7:]
'Python je jazyk.'
```

a možno ich použiť aj na niektoré iné údajové typy.

Najpoužívanejšie funkcie pre reťazce:

Funkcia	Použitie
len(*s*)	vráti dĺžku reťazca s, počet jeho znakov; pre prázdny reťazec vráti nulu
s.count (r, zač, kon)	vráti počet výskytov reťazca r v reze reťazca s

	>>> s = "Mama ma Emu"
	>>> s.count("ma")
	2
	>>> s.count("má")
	0
	>>> s.count("ma",2,6)
	1

s.find (r, zač, kon)	vráti najľavejší výskyt reťazca r v reze reťazca s alebo -1
	>>> s.find("ma")
	2
	>>> s.find("Ma")
	0
	>>> s.find("má")
	-1

s.format (parametre)	vráti reťazec naformátovaný podľa parametrov - kapitola Formátovanie reťazcov

s.index (r, zač, kon)	vráti index prvého výskytu zľava reťazca r v reze reťazca s alebo vyvolá výnimku;
	>>> s.index("ma")
	2
	>>> s.index("E")
	8
	>>> s.index("má")
	Traceback (most recent call last):
	File "<pyshell#18>", line 1, in <module>
	s.index("má")
	ValueError: substring not found
	>>> s.index("ma", 4)
	5
	s.rindex(r, zač, kon) vráti index prvého výskytu r sprava alebo vyvolá výnimku
	>>> s.rindex("ma")
	5
	>>> s.rindex("m",4)
	9
	>>> s.rindex("m", 0, 4)
	2

s.isalpha ()	vráti True, ak je reťazec s neprázdny a každý znak je písmenom
s.isdigit ()	vráti True, ak je reťazec s neprázdny a každý jeho znak je číslicou z kódu ASCII (0 až 9)
s.isalnum ()	vráti True, ak je reťazec neprázdny a každý znak v s je písmenom alebo číslicou

r.join (p)	vráti reťazec spojených reťazcov oddelených reťazcom r vytvorený z postupnosti p reťazcov
	>>> "".join(["Pondelok", "Utorok", "Streda", "Štvrtok", "Piatok"])
	'PondelokUtorokStredaŠtvrtokPiatok'
	>>> ", ".join(["Pondelok", "Utorok", "Streda", "Štvrtok", "Piatok"])
	'Pondelok, Utorok, Streda, Štvrtok, Piatok'

s.lower ()	vráti kópiu reťazca s, veľké písmená zmenené na malé
s.replace (r, w, n)	vráti kópiu reťazca s, v ktorej je každý reťazec r (alebo maximálne n-krát) nahradený reťazcom w >>> "jelenovi pivo nelej".replace(" ","") 'jelenovipivonelej'

s.split (r, n)	vráti zoznam reťazcov oddelených najviac n-krát reťazcom r; ak nie je zadané r, rozdelí sa podľa medzier
	>>> s = "13 120.75"
	>>> s.split()
	['13', '120.75']
	>>> v = s.split()
	>>> sucet = int(v[0]) + float(v[1])
	>>> sucet
	133.75
	pre delenie sprava možno použiť s.rsplit(r, n)

s.splitlines (b)	vráti zoznam riadkov, ktorý je výsledkom rozdelenia viacriadkového reťazca s podľa

	oddeľovačov riadkov (\n); ak b nemá uvedenú hodnotu True, tak sa oddeľovače riadkov odstránia
	`>>> s = """Horela hora, horela`
	`V tej hore hrala kapela`
	`Kapela hrala zvesela, jejé`
	`Zatiaľ čo hora horela"""`
	`>>> s.splitlines()`
	`['Horela hora, horela', 'V tej hore hrala kapela', 'Kapela hrala zvesela, jejé', 'Zatiaľ čo hora horela']`
	`>>> s.splitlines(True)`
	`['Horela hora, horela\n', 'V tej hore hrala kapela\n', 'Kapela hrala zvesela, jejé\n', 'Zatiaľ čo hora horela']`

s.strip (z)
s.lstrip(z)
s.rstrip(z)
	vráti kópiu reťazca s bez začiatočných a koncových bielych znakov (alebo bez znakov uvedených v reťazci z, musia byť však na začiatku alebo konci reťazca s)
	`>>> s = "  -12.75 6.58 "`
	`>>> s.strip()`
	`'-12.75 6.58'`
	s.lstrip(z) odreže biele znaky zo začiatku, s.rstrip(z) z konca reťazca z

s.upper ()	vráti kópiu reťazca s, malé písmená zmenené na veľké

Viacriadkové reťazce

Ak potrebujeme vytvoriť jeden reťazec, ktorého znaky sú vo viacerých, za sebou idúcich riadkoch, musíme na začiatku a konci reťazca použiť tri apostrofy alebo tri úvodzovky. Vytvorí sa reťazec, ktorý obsahuje na miestach prechodov na nový riadok oddeľovače riadkov (\n).

```
>>> viacriadkovy_retazec = """Horela hora, horela
V tej hore hrala kapela
Kapela hrala zvesela, jejé
Zatiaľ čo hora horela"""
>>> viacriadkovy_retazec
'Horela hora, horela\nV tej hore hrala kapela\nKapela hrala zvesela, jejé\n
Zatiaľ čo hora horela'
>>> print(viacriadkovy_retazec)
Horela hora, horela
V tej hore hrala kapela
Kapela hrala zvesela, jejé
Zatiaľ čo hora horela
```

Používanie neštandardných (na klávesnici sa nevyskytujúcich) znakov Unicode

Každý znak uvedený v tabuľke Unicode má jednoznačne priradené poradové číslo. Ak chceme použiť neštandardný znak, potrebujeme poznať jeho poradové číslo v kódovacej tabuľke Unicode, najlepšie v šestnástkovej sústave. K poradovému číslu zvoleného znaku sa môžeme dostať napríklad pomocou aplikácie Microsoft Word: Vložiť - Symbol - Ďalšie symboly... - záložka Symboly - Písmo: (normálny text) - nájdeme hľadaný znak a z poľa Kód znaku opíšeme štvorciferné hexadecimálne číslo z: Unicode (šestnástkovo). Napríklad pre znak „na druhú“ (^{2}) to je 00B2, pre znak „na tretiu“ (^{3}) to je 00B3, pre grécke písmeno Ω to je 03A9, pre Ω pomenované ohm 2126, pre dolný index $_1$ 2081 atď. Na zobrazenie znaku máme dve možnosti. Môžeme využiť „priamy“ zápis znaku, ktorý obsahuje '\uHHHH' alebo "\uHHHH", t.j. apostrofy alebo úvodzovky (pokiaľ kód znaku nie je priamo zapísaný v reťazci!), opačnú lomku - vkladá sa v slovenskej klávesnici pravé Alt+Q a HHHH reprezentuje štvorciferné hexadecimálne číslo z Unicodu pre zvolený znak, napríklad zápis '\u03a9' vypíše grécke písmeno Ω.

Ukážka:

```
>>> '\u03a9'
'Ω'
>>> r1 = float(input("Zadaj odpor R\u2081 v \u2126: "))
Zadaj odpor R₁ v Ω: 10
>>> print("R\u2081 = {:.2f} \u2126".format(r1))
R₁ = 10.00 Ω
```

```
>>> print( "Obsah = {:.2f} m\u00b2".format(5.4))
Obsah = 5.40 m²
>>> print( "Objem = {:.2f} m\u00b3".format(9.8765))
Objem = 9.88 m³
```

Môžeme tiež využiť funkciu chr(poradové_číslo), ktorá vráti znak - reťazec, zodpovedajúci poradovému číslu z kódovacej tabuľky. Poradové číslo znaku môžeme zapísať v desiatkovej alebo šestnástkovej sústave. Preferuje sa zápis v tvare 0xHHHH kde každé H reprezentuje jednu cifru šestnástkovej číselnej sústavy a znak bude zobrazený po zápise chr(0xHHHH). Python podporuje aj zápis u'\uHHHH' alebo u"\uHHHH", kde u pred poradovým číslom znaku je rovnocenné chr(). „Opačnou“ funkciou je ord(znak), ktorá vráti poradové číslo znaku v desiatkovej číselnej sústave.

Ukážka:

```
>>> ord('Ω')                    Poznámka: Medzi apostrofy sme vložili grécke písmeno Ω z MS Wordu.
937
>>> chr(937)
'Ω'
>>> chr(0x3A9)                  Poznámka: Pomocou kalkulačky sme zistili, že 937₁₀ je 3A9₁₆.
'Ω'
>>> chr(0x03a9)                 Poznámka: Len sme pozmenili zápis (doplnili 0 a zmenili veľkosť písmena A).
'Ω'
```

Pre kľúčové slová „znak ohm v unicode“ nám Google ponúkol tabuľku (http://www.fileformat.info/info/unicode/char/2126/index.htm):

UTF-16 (hex)	0x2126 (2126)
UTF-16 (decimal)	8 486
UTF-32 (hex)	0x00002126 (2126)
UTF-32 (decimal)	8 486
C/C++/Java source code	"\u2126"
Python source code	**u"\u2126"**

Pokračovanie ukážky:

```
>>> r1 = float(input("Zadaj odpor R" + chr(0x2081) + " v " + chr(0x2126) + ": "))
Zadaj odpor R₁ v Ω: 10
>>> print("Odpor R{} = {:.2f} {}".format(chr(0x2081), r1, chr(8486)))
Odpor R₁ = 10.00 Ω
>>> print("Odpor R{} = {:.2f} {}".format(chr(0x2081), r1, u"\u2126"))
Odpor R₁ = 10.00 Ω
>>> print( "Obsah = {:.2f} m{}".format( 5.4, chr(0x00B2)))
Obsah = 5.40 m²
>>> print( "Objem = {:.2f} m{}".format(9.8765, u'\u00b3'))
Objem = 9.88 m³
```

Možností zápisu na klávesnici sa nevyskytujúcich znakov je v Pythone viacej. Najľahšie sa asi dostaneme k hexadecimálnemu zápisu poradového čísla hľadaného znaku. Pokiaľ nám to zadanie umožňuje, je najjednoduchšie použiť zápis \uHHHH. Ak ale potrebujeme napríklad vypísať znaky z určitého intervalu poradových čísel, bez zápis chr(poradové_číslo) si asi neporadíme.