

Reťazce

Poznámky k téme

Reťazec (typ `str`) je text – postupnosť znakov. Zatiaľ sme reťazce hlavne vypisovali; v tejto téme sa naučíš s nimi naozaj pracovať: vyberať jednotlivé znaky a časti textu, prechádzať text znak po znaku, meniť veľkosť písmen, deliť vetu na slová a pracovať s kódmi znakov. Úlohy s textom patria k najčastejším na písomkách aj na maturite.

Čo sa naučíš

- vybrať znak podľa indexu (aj záporného) a časť textu pomocou rezu,
- prechádzať reťazec cyklom a vytvárať nový reťazec,
- používať metódy reťazcov (`upper`, `strip`, `replace`, `count`, `find`, ...),
- rozdeliť text na slová (`split`) a spojiť ich (`join`),
- pracovať s kódmi znakov (`ord`, `chr`).

1 Reťazec ako postupnosť znakov

Každý znak reťazca má svoje poradové číslo – **index**. Indexy sa začínajú od **nuly**. Python dovoľuje aj **záporné** indexy, ktoré sa počítajú od konca: `-1` je posledný znak, `-2` predposledný atď.

index	0	1	2	3	4	5
prvok	P	y	t	h	o	n
záporný index	-6	-5	-4	-3	-2	-1

Počet znakov zistí funkcia `len`. Platné indexy sú od `0` po `len(s) - 1` (alebo od `-len(s)` po `-1`).

```
>>> s = "Python"
>>> len(s)
6
>>> s[0]
'P'
>>> s[5]
'n'
>>> s[-1]
'n'
>>> s[-2]
'o'
>>> s[len(s) - 1]
'n'
```

Ak index „vyjde“ mimo reťazca, nastane chyba:

```
>>> s = "Python"
>>> s[6]
IndexError: string index out of range
```

1.1 Reťazce sa nedajú meniť

Reťazec je **nemenný** (*immutable*): jednotlivý znak sa nedá prepísať.

```
>>> s = "mama"
>>> s[0] = "l"
TypeError: 'str' object does not support item assignment
```

Ak potrebuješ „zmenený“ text, vytvor **nový** reťazec a ulož ho (aj do tej istej premennej):

```
>>> s = "mama"
>>> s = "l" + s[1:]
>>> s
'lama'
```

2 Rezy

Rez (*slice*) vyberie časť reťazca: `s[od:do]` dá znaky od indexu **od** **po index** **do** – **ten už nie**.

Vynechané **od** znamená „od začiatku“, vynechané **do** „až do konca“.

```
>>> s = "programovanie"
>>> s[0:3]
'pro'
>>> s[:3]
'pro'
>>> s[3:7]
'gram'
>>> s[7:]
'ovanie'
>>> s[-4:]
'anie'
>>> s[1:-1]
'rogramovani'
>>> s[:]
'programovanie'
```

Tretie číslo je **krok**. Záporný krok prechádza reťazec odzadu – `s[::-1]` je najjednoduchší spôsob, ako reťazec obrátiť:

```
>>> s = "programovanie"
>>> s[::-2]
'pormvne'
>>> s[1::-2]
'rgaoai'
>>> s[::-1]
'einavomargorp'
```

Na rozdiel od indexov rezy nikdy nespôsobia chybu – ak presahujú koniec, jednoducho vrátia menej znakov (alebo prázdny reťazec):

```
>>> "Ahoj"[2:100]
'oj'
>>> "Ahoj"[10:]
''
```

Rez	Význam
<code>s[:n]</code>	prvých n znakov
<code>s[-n:]</code>	posledných n znakov
<code>s[n:]</code>	všetko okrem prvých n znakov
<code>s[1:-1]</code>	bez prvého a posledného znaku
<code>s[::2]</code>	každý druhý znak (od prvého)
<code>s[::-1]</code>	reťazec odzadu

3 Prechádzanie reťazca

Cyklus `for` dokáže prejsť reťazec **znak po znaku**:

```
slovo = "kolotoč"
for znak in slovo:
    print(znak, end=" ")
print()
```

Výstup:

```
k o l o t o č
```

Ak potrebuješ aj **pozíciu** znaku, prechádzaj indexy:

```
slovo = "kolo"
for i in range(len(slovo)):
    print(i, slovo[i])
```

Výstup:

```
0 k
1 o
2 l
3 o
```

3.1 Počítanie znakov

Klasický vzor s počítadlom – spočítame samohlásky:

```
veta = "Programovanie v Pythone je zábava"
pocet = 0
for znak in veta.lower():
    if znak in "aáäeéííoóôuúýý":
        pocet += 1
print("Samohlások:", pocet)
```

Výstup:

```
Samohlások: 13
```

Operátor `in` zisťuje, či sa znak (alebo celý podreťazec) nachádza v inom reťazci: `"y" in "Python"` je `True`, `"tho" in "Python"` tiež.

3.2 Vytváranie nového reťazca

Nový reťazec sa často skladá znak po znaku: začneme prázdny reťazcom `""` a v cykle pridávame:

```
text = "Ahoj, ako sa máš?"
bez_samohlasok = ""
for znak in text:
    if znak.lower() not in "aáäeéííoóôuúýý":
        bez_samohlasok += znak
print(bez_samohlasok)
```

Výstup:

```
hj, k s mš?
```

```
slovo = "python"
obratene = ""
for znak in slovo:
    obratene = znak + obratene    # nový znak pred doterajší text
print(obratene)
```

Výstup:

```
nohtyp
```

Obrátenie sa dá zapísať aj rezom `slovo[::-1]`, ale princíp skladania textu v cykle budeš potrebovať v mnohých iných úlohách.

4 Metódy reťazcov

Metóda je funkcia, ktorá patrí k hodnote. Volá sa cez bodku: `text.upper()`. Metódy reťazcov **nemenia** pôvodný reťazec – vracajú **nový**:

```
>>> meno = "eva"
>>> meno.upper()
'EVA'
>>> meno
'eva'
>>> meno = meno.capitalize()
>>> meno
'Eva'
```

Ak chceš výsledok zachovať, musíš ho uložiť (`meno = meno.capitalize()`).

4.1 Prehľad najpoužívanějších metód

Metóda	Čo vráti	Príklad	Výsledok
<code>upper()</code>	veľké písmená	<code>"Ahoj".upper()</code>	<code>'AH0J'</code>
<code>lower()</code>	malé písmená	<code>"Ahoj".lower()</code>	<code>'ahoj'</code>
<code>capitalize()</code>	prvé písmeno veľké, ostatné malé	<code>"pYTHON".capitalize()</code>	<code>'Python'</code>
<code>title()</code>	každé slovo s veľkým písmenom	<code>"jan novak".title()</code>	<code>'Jan Novak'</code>
<code>strip()</code>	bez medzier (a koncov riadkov) na okrajoch	<code>" ok \n".strip()</code>	<code>'ok'</code>
<code>replace(a, b)</code>	každé <code>a</code> nahradené <code>b</code>	<code>"a-b-c".replace("-", "+")</code>	<code>'a+b+c'</code>
<code>count(x)</code>	počet výskytov <code>x</code>	<code>"banana".count("a")</code>	<code>3</code>
<code>find(x)</code>	index prvého výskytu, inak <code>-1</code>	<code>"banana".find("n")</code>	<code>2</code>
<code>index(x)</code>	ako <code>find</code> , ale bez výskytu chyba	<code>"banana".index("n")</code>	<code>2</code>
<code>startswith(x)</code>	začína na <code>x</code> ?	<code>"Python".startswith("Py")</code>	<code>True</code>
<code>endswith(x)</code>	končí na <code>x</code> ?	<code>"obr.png".endswith(".png")</code>	<code>True</code>

```
>>> "banana".find("x")
-1
>>> "banana".count("an")
2
>>> " viac medzier ".strip()
'viac medzier'
>>> "Jablko".replace("J", "j").upper()
'JABLK0'
```

Metódy sa dajú reťaziť – posledný príklad najprv nahradí písmeno a výsledok prevedie na veľké písmená.

4.2 Testovacie metódy

Metódy začínajúce na `is` vracajú `True` alebo `False`:

Metóda	True, ak...
<code>isdigit()</code>	všetky znaky sú čísllice (a reťazec nie je prázdny)
<code>isalpha()</code>	všetky znaky sú písmená
<code>isalnum()</code>	všetky znaky sú písmená alebo čísllice
<code>isupper()</code> , <code>islower()</code>	písmená sú veľké / malé
<code>isspace()</code>	všetky znaky sú medzery (biele znaky)

```
>>> "2026".isdigit()
True
>>> "20.5".isdigit()
False
>>> "Žilina".isalpha()
True
>>> "heslo123".isalnum()
True
>>> "A".isupper()
True
```

`isdigit` sa hodí na kontrolu vstupu pred prevodom na číslo – ale pozor, záporné a desatinné čísla (" -5", "20.5") neprejdú. Spoľahlivejší spôsob je v téme Výnimky.

5 split a join

Metóda `split()` rozdelí reťazec na **zoznam** častí. Bez argumentu delí podľa medzier (aj viacerých za sebou); s argumentom podľa zadaného oddeľovača:

```
>>> "Učíme sa programovať v Pythone".split()
['Učíme', 'sa', 'programovať', 'v', 'Pythone']
>>> "jeden dva tri".split()
['jeden', 'dva', 'tri']
>>> "12;45;7".split(";")
['12', '45', '7']
>>> "2026-10-01".split("-")
['2026', '10', '01']
```

Výsledok je zoznam (téma Zoznamy): `len` povie počet slov, `for` ich prejde, `[0]` vyberie prvé.

Opak urobí metóda `join` – spojí zoznam reťazcov a medzi časti vloží reťazec, na ktorom ju voláme:

```
>>> slova = ["jablko", "hruška", "slivka"]
>>> ", ".join(slova)
'jablko, hruška, slivka'
>>> "-".join(slova)
'jablko-hruška-slivka'
>>> "".join(["a", "b", "c"])
'abc'
```

```
veta = "programovanie v pythone je super"
slova = veta.split()
print("Počet slov:", len(slova))
print("Prvé slovo:", slova[0])
print("Odzadu:", " ".join(slova[::-1]))
```

Výstup:

```
Počet slov: 5
Prvé slovo: programovanie
Odzadu: super je pythone v programovanie
```

Pozor: `join` spája iba **reťazce**. Zoznam čísel treba najprv previesť: `" ".join(str(x) for x in cisla)`.

6 Znaky a ich kódy

Každý znak má v počítači číselný **kód** (podľa normy Unicode; pre anglickú abecedu, číslice a interpunkciu sa zhoduje so staršou tabuľkou ASCII). Funkcia `ord` vráti kód znaku, funkcia `chr` vráti znak k zadanému kódu:

```
>>> ord("A")
65
>>> ord("a")
97
>>> ord("0")
48
>>> chr(66)
'B'
>>> chr(ord("a") + 1)
'b'
>>> ord("č")
269
```

Znaky	Kódy
"0" až "9"	48 až 57
"A" až "Z"	65 až 90
"a" až "z"	97 až 122
medzera	32

Písmená bez diakritiky idú za sebou, takže sa s nimi dá „počítat“: písmeno o tri ďalej, poradie písmena v abecede (`ord(z) - ord("a")`), prevod číslice na číslo (`ord("7") - ord("0")` je 7).

6.1 Porovnávanie reťazcov

Reťazce sa porovnávajú znak po znaku podľa kódov. Preto `"B" < "a"` ($66 < 97$) a `"apple" < "banana"`. Písmená s diakritikou majú vyššie kódy – `"č"` je „za“ písmenom `"z"`. Ak chceš porovnávať bez ohľadu na veľkosť písmen, porovnávaj `lower()` verzie.

7 Riešené úlohy

7.1 Palindróm

Zadanie: Palindróm je text, ktorý sa číta rovnako spredu aj odzadu (napríklad „kajak“ alebo „Jelenovi pivo nelej“). Zisti, či je zadaný text palindróm; medzery a veľkosť písmen neber do úvahy.

Riešenie: Najprv text „vyčistíme“: prevedieme na malé písmená a odstránime medzery. Potom ho porovnáme s jeho obrátenou verziou.

```
text = input("Text: ")
cisty = text.lower().replace(" ", "")
if cisty == cisty[::-1]:
    print("Je to palindróm.")
else:
    print("Nie je to palindróm.")
```

Výstup:

```
Text: Jelenovi pivo nelej
Je to palindróm.
```

7.2 Cézarova šifra

Zadanie: Zašifruj text Cézarovou šifrou s posunom 3: každé písmeno anglickej abecedy nahraď písmenom o 3 miesta ďalej (a → d, ..., x → a). Ostatné znaky nechaj tak.

Riešenie: Poradie písmena v abecede je `ord(znak) - ord("a")` (0 až 25). Pripočítame posun a zvyšok po delení 26 zabezpečí, že sa za z pokračuje opäť od a. Veľké písmená spracujeme rovnako, len od "A".

```
def sifruj(text, posun):
    vysledok = ""
    for znak in text:
        if "a" <= znak <= "z":
            vysledok += chr((ord(znak) - ord("a") + posun) % 26 + ord("a"))
        elif "A" <= znak <= "Z":
            vysledok += chr((ord(znak) - ord("A") + posun) % 26 + ord("A"))
        else:
            vysledok += znak
    return vysledok

tajne = sifruj("Veni, vidi, vici! xyz", 3)
print(tajne)
print(sifruj(tajne, -3))
```

Výstup:

```
Yhql, ylg!, ylf! abc
Veni, vidi, vici! xyz
```

Dešifrovanie je šifrovanie so záporným posunom. Pri zápornom čísle funguje `%` v Pythone tak, že výsledok je stále od 0 po 25.

7.3 Kontrola hesla

Zadanie: Heslo je silné, ak má aspoň 8 znakov a obsahuje aspoň jedno veľké písmeno, jedno malé písmeno a jednu číslicu. Napíš funkciu, ktorá to overí.

Riešenie: Prejdeme znaky hesla a pre každú požiadavku si nastavíme príznak (`True`, keď nájdeme vhodný znak).

```
def silne_heslo(heslo):
    velke = male = cislica = False
    for znak in heslo:
        if znak.isupper():
            velke = True
        elif znak.islower():
            male = True
        elif znak.isdigit():
            cislica = True
    return len(heslo) >= 8 and velke and male and cislica

for h in ["heslo", "Heslo123", "HESL01234", "dlhehesloBezCisla"]:
    print(h, silne_heslo(h))
```

Výstup:

```
heslo False
Heslo123 True
HESL01234 False
dlhehesloBezCisla False
```

7.4 Znaký a slová vo vete

Zadanie: Načítaj vetu a vypíš počet znakov bez medzier, počet slov a najdlhšie slovo.

```
veta = input("Veta: ")
slova = veta.split()
print("Znakov bez medzier:", len(veta.replace(" ", "")))
print("Slov:", len(slova))
najdlhsie = ""
for slovo in slova:
    if len(slovo) > len(najdlhsie):
        najdlhsie = slovo
print("Najdlhšie slovo:", najdlhsie)
```

Výstup:

```
Veta: Programovanie v Pythone je naozaj zábavné
Znakov bez medzier: 36
Slov: 6
Najdlhšie slovo: Programovanie
```

7.5 Iniciály

Zadanie: Z celého mena (napríklad „ema katarína nováková“) vypíš iniciály s bodkami veľkými písmenami: E. K. N.

```
meno = "ema katarína nováková"
inicialy = []
for cast in meno.split():
    inicialy.append(cast[0].upper() + ".")
print(" ".join(inicialy))
```

Výstup:

```
E. K. N.
```

8 Časté chyby

- **Index mimo rozsahu:** posledný znak je `s[len(s) - 1]` alebo `s[-1]`, nie `s[len(s)]`.
- **Zabudnuté uloženie výsledku metódy:** `text.upper()` samo o sebe `text` nezmení – treba `text = text.upper()`.
- **Pokus o zmenu znaku:** `s[0] = "X"` je chyba; vytvor nový reťazec.
- **Veľké a malé písmená:** `"Áno" == "áno"` je `False` – porovnávaj `lower()` verzie.
- `split(" ")` **verzus** `split()`: pri viacerých medzerách za sebou vytvorí `split(" ")` aj prázdne reťazce, `split()` nie.

```
>>> "a b".split(" ")
['a', '', 'b']
>>> "a b".split()
['a', 'b']
```

Zhrnutie

- Indexy od 0, záporné od konca (`s[-1]` je posledný znak); `len(s)` je počet znakov.
- Rez `s[od:do:krok]` – `do` už nepatrí; `s[::-1]` obráti reťazec; rezy nespôsobia chybu.
- Reťazce sú nemenné; metódy vracajú nový reťazec, ktorý treba uložiť.
- `for znak in s` prechádza znaky; `for i in range(len(s))` prechádza indexy.
- `in` testuje výskyt znaku alebo podreťazca.
- `split()` rozdelí text na zoznam slov, `oddelovac.join(zoznam)` ich spojí.
- `ord(znak)` → kód, `chr(kod)` → znak; písmená a–z majú kódy 97–122.

Úlohy na precvičenie

1. Načítaj slovo a vypíš ho zvisle (každý znak na samostatnom riadku) a potom odzadu.
2. Načítaj vetu a vypíš, koľkokrát sa v nej vyskytuje každá samohláska a, e, i, o, u.
3. Načítaj meno a priezvisko (oddelené medzerou) a vypíš ich v tvare `PRIEZVISKO, Meno`.
4. Napíš funkciu, ktorá v texte nahradí každú číslicu znakom `#`.
5. Načítaj text a vypíš ho „striedavo“: `pYtHoN` (znaky na párnych indexoch malé, na nepárnych veľké).
6. Dešifruj text `"Dkrm, dnr vd pdv?"`, ktorý bol zašifrovaný Cézarovou šifrou s posunom 3.

Úlohy k tejto téme nájdeš na learnwithmente.com v časti Informatika → Python.