

## Téma: Použitie štandardných funkcií na prácu s reťazcami

Zopakuj si: Python - reťazce, predovšetkým:

**Funkcia len(reťazec)** - vráti počet znakov, dĺžku reťazca *reťazec*.

**Rezací operátor** reťazec [ *začiatok* : *koniec* : *krok* ] má mnoho variant, napríklad

- reťazec [ *začiatok* : ] - vracia podreťazec zo zadaného reťazca so znakmi od znaku s indexom *začiatok* až po posledný znak pôvodného reťazca
- reťazec [ : *koniec* ] - vracia podreťazec zo zadaného reťazca so znakmi od znaku s indexom 0 až pred znak na indexe *koniec*.
- reťazec [ *začiatok* : *koniec* ] - vracia podreťazec zo zadaného reťazca so znakmi od znaku s indexom *začiatok* až pred znak na indexe *koniec*.
- reťazec [ *začiatok* : *koniec* : *krok* ] - vracia podreťazec zo zadaného reťazca so znakmi od znaku s indexom *začiatok* až pred znak na indexe *koniec*; *krok* udáva, znak s akým prírastkom indexu sa má preniesť do podreťazca.

**Funkcia reťazec.count (hľadať, začiatok, koniec)** - vráti počet výskytov reťazca *hľadať* v reze reťazca *reťazec*.

**Funkcia reťazec.find (hľadať, začiatok, koniec)** - vráti index prvého výskytu zľava reťazca *hľadať* v reze reťazca *reťazec*. Ak sa hľadaný reťazec v reze reťazca nenachádza, funkcia vráti číslo -1.

**Funkcia reťazec.index (hľadať, začiatok, koniec)** - vráti index prvého výskytu zľava reťazca *hľadať* v reze reťazca *reťazec* alebo vyvolá výnimku (ak sa reťazec *hľadať* v reze nevyskytuje).

**Funkcia reťazec.isalpha()** vráti True, ak je *reťazec* neprázdny a každý znak je písmenom (aj národnej abecedy).

**Funkcia reťazec.isdigit()** vráti True, ak je *reťazec* neprázdny a každý jeho znak je číslicou (od 0 po 9).

**Funkcia reťazec.isalnum()** vráti True, ak je *reťazec* neprázdny a každý znak v reťazci je písmenom (aj národnej abecedy) alebo číslicou (od 0 po 9).

**Funkcia oddelovač.join(reťazce)** vráti reťazec spojených reťazcov vytvorený z postupnosti *reťazce*, oddelených reťazcom *oddelovač*.

**Funkcia reťazec.lower()** vráti kópiu reťazca *reťazec*, veľké písmená zmenené na malé. Našu funkciu si nazvime *mlower* a predpokladajme, že v reťazci boli použité len písmená anglickej abecedy.

**Funkcia reťazec.replace(nahrad\_čo, nahrad\_čím, n-krát)** vráti kópiu reťazca *reťazec*, v ktorej je každý reťazec *nahrad\_čo* nahradený reťazcom *nahrad\_čím* najviac *n-krát*.

**Funkcia reťazec.split(oddelené\_čím, n-krát)** vráti zoznam reťazcov oddelených najviac *n-krát* reťazcom *oddelené\_čím*; ak nie je zadané *oddelené\_čím*, *reťazec* sa rozdelí podľa medzier; *n-krát* je nepovinný parameter.

**Úloha 1:** Napíšte program, ktorý určí cenu SMS podľa počtu slov. Cena za každé slovo je 0,10 €.

```
SLOVO = 0.10 #€
sprava = input("Správa: ")
pocetSlov = len(sprava.split())
print("Počet slov:", pocetSlov)
print("Cena SMS: {:.2f} €".format(pocetSlov*SLOVO))
```

**Použitie:**

```
Správa: Prídem večer 18:15. Peter
Počet slov: 4
Cena SMS: 0.40 €
```

**Úloha 2:** Vytvorte funkciu, ktorá vráti True, ak je reťazec symetrický, inak vráti False. Symetrické sú napríklad slová radar, ABBA.

Klasické „nepythonovské“ riešenie:

```
def jeSymetrickyy(s):
    for i in range(len(s)//2):
        if s[i] != s[len(s)-i-1]:
            return False
    return True
```

Využijeme skutočnosť, že rezací operátor s krokom -1 vracia reťazec so znakmi v opačnom poradí, ako sú v pôvodnom reťazci (posledný znak bude prvý, predposledný druhý atď.). Ale pozor na indexy, ktoré budú použité pri kroku -1, keď sa ide len do polovice reťazca!

```
def jeSymetricky(s):
    return s == s[::-1]

''' Porovnanie len príslušných polovic reťazca
    if len(s)%2 == 0:
        return s[:len(s)//2] == s[:len(s)//2-1:-1]
    else:
        return s[:len(s)//2] == s[:len(s)//2:-1]
'''
```

**Použitie:**

```
s = "radar"
print("Je symetrický?", jeSymetricky(s))
výpis: Je symetrický? True
print("Je symetrický?", jeSymetricky(s))
výpis: Je symetrický? True
```

**Úloha 3:** Palindróm je slovo, veta, číslo (všeobecne akákoľvek postupnosť symbolov), ktorá má tú vlastnosť, že ju možno čítať v ľubovoľnom smere (sprava doľava alebo zľava doprava) a má vždy rovnaký význam. Pri posudzovaní, či ide o rovnaký význam sa obvykle neberú do úvahy medzery medzi slovami a diakritika (ak je použitá). V našom prípade diakritiku nepoužijeme, ale reťazec môže obsahovať medzery a na konci interpunkčný znak, teda dovolený je napríklad reťazec: Jelenovi pivo nelejl! alebo Kobyla ma maly bok.

Zrejme po úprave reťazca môžeme použiť funkciu jeSymetricky. Úprava spočíva v premene všetkých písmen napríklad na malé písmená, v odstránení medzier a iných ako alfanumerických znakov z reťazca.

```
def jePalindrom(s):
    s = s.lower()
    pom = ""
    for znak in s:
        if znak.isalnum():
            pom += znak
    return pom == pom[::-1]
```

**Použitie:**

```
s = "Jelenovi pivo nelejl!"
print('"' + s + '"' + " je palindróm?", jePalindrom(s))
výpis: "Jelenovi pivo nelejl!" je palindróm? True
```

**Úloha 4a:** Vytvorte program na zašifrovanie a odšifrovanie správy (reťazca) posunutím každého znaku v správe o zadané celé číslo (posun).

```
def zasifruj(sprava, posun = 1):
    novy = ""
    for znak in sprava:
        novy += chr(ord(znak) + posun)
    return novy

def odsifruj(sprava, posun = 1):
    novy = ""
    for znak in sprava:
        novy += chr(ord(znak) - posun)
    return novy
```

**Použitie:**

```
text = input("Čo zašifrovať? ")
posun = int(input("O koľko pozícií? "))
zasifrovane = zasifruj(text, posun)
print("Zašifrované: ", zasifrovane)
```

```
print("Odšifrované: ", odsifruj(zasifrovane, posun))
výpis:
Čo zašifrovať? Abeceda zjedla Jana 1.septembra 2015.
O koľko pozícií? 3
Zašifrované:  Dehfhgd#}mhgod#Mdqd#4lvhswhpseud#53481
Odšifrované:  Abeceda zjedla Jana 1.septembra 2015.
```

**Úloha 4b:** Vytvorte program na dynamické zašifrovanie správy (reťazca) posunutím každého znaku v reťazci o zadané celé číslo (posun) + jeho index v texte. Napr. Abeceda pre posun = 1 sa zakóduje na Bdhgijh.

```
def zasifruj(sprava, posun = 1):
    novy = ""
    for i in range(len(s)):
        novy += chr(ord(s[i])+posun+i)
    return novy

# iné riešenie:
novy = ""
for znak in sprava:
    novy += chr(ord(znak) + posun)
    posun += 1
return novy
```

Program doplňte o funkciu dešifruj.

**Úloha 4c:** Vytvorte program na zašifrovanie správy (reťazca) posunutím každého znaku v správe o +1 doprava, avšak každé písmeno bude zašifrované na písmeno, t.j. Z na A resp. z na a . Ostatné znaky sa nezmenia.

```
def sifrujCyklicky(sprava):
    POSUNUT = "ABCDEFGHGIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxy" # chýbajú Z a z!
    novy = ""
    for znak in sprava:
        if POSUNUT.count(znak):
            novy += chr(ord(znak)+1)
        elif znak == 'Z':
            novy += 'A'
        elif znak == 'z':
            novy += 'a'
        else:
            novy += znak # ak má posunúť každý znak: novy += chr(ord(znak)+1)
    return novy
```

**Použitie:**

```
text = input("Čo zašifrovať? ")
zasifrovane = sifrujCyklicky(text)
print("Zašifrované: ", zasifrovane)
```

**výpis:**

```
Čo zašifrovať? Abeceda zjedla Zola 1.septembra 2016.
Zašifrované:  Bcdffeb akfemb Apmb 1.tfqufnscsb 2016.
```

Program doplňte o funkciu odsifrujCyklicky a o vlastnosť, aby aj číslice šifrovalo a dešifrovalo len na číslice, t.j. 9 na 0 a opačne.

**Úloha 5a:** Vytvorte funkciu, ktorá zistí, či zadaný reťazec obsahuje aspoň jedno písmeno.

```
def obsahujeAsponJednoPismeno(s):
    for znak in s:
        if znak.isalpha():
            return True
    return False
```

Použitie:

```
ret = ""
print("Obsahuje aspoň jedno písmeno?: ", obsahujeAsponJenoPismeno(ret))
vypíše False
```

Úlohu 5a upravte tak, aby funkcia zisťovala, či zadaný reťazec obsahuje veľké (malé) písmeno anglickej abecedy prípadne akejkolvek abecedy.

**Úloha 5b:** Vytvorte funkciu, ktorá vráti index výskytu prvého písmena anglickej abecedy v reťazci alebo hodnotu -1.

```
def indexVyskytuPrvehoPismenaAA(s):
    for i in range(len(s)):
        if s[i].isalpha() and "A"<=s[i]<="z":
            return i
    return -1
```

Použitie:

```
ret = "11.1. Peter a Beáta neprídu."
print("Index výskytu prvého písmena angl.abecedy:", indexVyskytuPrvehoPismenaAA(ret))
vypíše 6
```

**Úloha 5c:** Vytvorte funkciu, ktorá vráti

```
def vratAbecednePrvePismenoAAzRetazca(s):
    ivpp = indexVyskytuPrvehoPismenaAA(s)
    if ivpp > -1:
        prve = s[ivpp]
        for i in range(ivpp+1, len(s)):
            if s[i].isalpha() and "A"<=s[i]<="z":
                if s[i] < prve:
                    prve = s[i]
        return prve
    else:
        return "neexistuje"
```

```
s = "11.1. Peter a Beáta neprídu."
print("Abecedne prvé písmeno v texte je:", vratAbecednePrvePismenoAAzRetazca(s))
```

**Úloha 5d:** Vytvorte funkciu, ktorá vráti

```
def vratAbecednePrveaPoslednePismenoAAzRetazca(s):
    ivpp = indexVyskytuPrvehoPismenaAA(s)
    if ivpp > -1:
        prve = posledne = s[ivpp]
        for i in range(ivpp+1, len(s)):
            if s[i].isalpha() and "A"<=s[i]<="z":
                if s[i] < prve:
                    prve = s[i]
                if s[i] > posledne:
                    posledne = s[i]
        return (prve, posledne)
    else:
        return ("neexistuje", "neexistuje")
```

```
ret = "11.1. Peter a Beáta neprídu."
vysledok = vratAbecednePrveaPoslednePismenoAAzRetazca(ret)
print("Abecedne prvé písmeno v reťazci:", vysledok[0])
print("Abecedne posledné písmeno v reť:", vysledok[1])
```

**Úloha 6:** Vytvorte funkciu, ktorá vráti štatistiku o reťazci, t.j. počet písmen, číslíc a iných znakov (prípadne zvlášť malých a veľkých písmen anglickej abecedy, prípadne počet riadkov).

```
text = "Horela hora, horela\nv tej hore hrala 2xkapela,..."
print("Text:", text)
print("Počet znakov:", len(text))

pocetMalychPismenAA = 0
pocetVelkychPismenAA = 0
pocetCislic = 0
pocetRiadkov = 1
for znak in text:
    if znak.isalpha():
        if 'a'<=znak<='z': pocetMalychPismenAA += 1
        else: pocetVelkychPismenAA += 1
    elif znak.isdigit(): pocetCislic += 1
    elif znak == '\n': pocetRiadkov += 1

print("Počet malých písmen angl. abecedy:", pocetMalychPismenAA)
print("Počet veľkých písmen angl.abecedy:", pocetVelkychPismenAA)
print("Počet číslíc:", pocetCislic)
pocetInychZnakov = len(text)-pocetMalychPismenAA-pocetVelkychPismenAA-pocetCislic-
pocetRiadkov+1
print("Počet iných znakov (okrem \n):", pocetInychZnakov)
print("Počet riadkov:", pocetRiadkov)
```

**Použitie:**

```
Text: Horela hora, horela
v tej hore hrala 2xkapela,...
Počet znakov: 49
Počet malých písmen: 35
Počet veľkých písmen: 1
Počet číslíc: 1
Počet iných znakov (okrem \n): 11
Počet riadkov: 2
```

**Úloha 7a:** Vytvorte funkciu, ktorá s využitím funkcie find vypíše všetky indexy výskytu hľadaného reťazca v reze iného reťazca. Ak sa hľadaný reťazec v reze nevyskytuje, vypíše o tom oznam.

```
def vypisVsetkyIndexyFind(s, hladat, od, po):
    indexy = ""
    while od < po:
        i = s.find(hladat, od, po)
        if i > -1:
            indexy += str(i) + " "
            od = i+1
        else: break
    if indexy == "":
        print("'+hladat+'" sa v reze reťazca nevyskytlo.")
    else:
        print("'+hladat+'" sa v reze vyskytlo na indexoch:", indexy)
```

**Použitie príkazov:**

```
s = "horela hora, v tej hore hrala kapela"
vypisVsetkyIndexyFind(s, "hor", 0, len(s))
vypíše:
'hor' sa v reze vyskytlo na indexoch: 0 7 19
```

**Úloha 7b:** Vytvorte funkciu, ktorá s využitím štandardných funkcií pre prácu s reťazcami vráti všetky indexy výskytu hľadaného reťazca v zadanom reťazci (napríklad ako n-ticu). Ak sa hľadaný reťazec v reťazci nevyskytuje, vráti -1.

```
def vratVsetkyIndexyCountFind(s, hladat):
    pocetVyskytov = s.count(hladat)
```

```

if pocetVyskytov > 0:
    indexy = ()
    zac = 0
    for i in range(0,pocetVyskytov):
        inx = s[zac:].find(hladat)
        indexy += (inx+zac,)
        zac = inx+zac+1
    return indexy
else:
    return -1
# hľadaný reťazec sa vyskytuje aspoň raz
# vytvorenie prázdnej n-tice
# vyhľadávanie začne od začiatku reťazca s
# treba nájsť postupne všetky výskyty
# nájsť v reze od indexu zac
# pridať do n-tice správny index od začiatku s
# nastaviť nový začiatok pre ďalší rez
# počet výskytov bol nula

```

**Volanie funkcie vratVsetkyIndexyCountFind:**

```

print(vratVsetkyIndexyCountFind("horela hora, v tej hore hrala kapela","hor"))
vráti: (0, 7, 19)

```

Funkciu vratVsetkyIndexyCountFind upravte tak, aby hľadala len v reze zadaného reťazca, t.j. v s[od:po].

**Úloha 8:** Vytvorte šifrovací program, ktorý po napísaní správy a zadaní tzv. kľúča zašifruje každý znak správy posunutím v tabuľke Unicode o poradové číslo príslušného znaku kľúča.

Napríklad ak je kľúčom alibaba, posunie prvý znak správy o  $\text{ord}("a") = 97$ , druhý znak správy o  $\text{ord}("l") = 108$ , tretí znak správy o  $\text{ord}("i") = 105$  atď. Po vyčerpaní všetkých znakov kľúča sa šifruje znakmi od začiatku kľúča, teda ako by bol kľúč alibabaaalibabaaalibaba...

Po zašifrovaní správy napíšte funkciu na odšifrovanie zašifrovanej správy.

Pripomíname, že funkcia  $\text{ord}(\text{znak})$  vráti poradové číslo znaku v tabuľke Unicode a funkcia  $\text{chr}(\text{celé\_číslo}_{10})$  vráti znak Unicode (ak sa dá zobrazit).

```

kluc = "alibaba"

def zasifruj(text, kluc):
    sifra = ""
    modulo = len(kluc)
    i = 0
    for znak in text:
        sifra += chr(ord(znak) + ord(kluc[i%modulo]))
        i += 1
    return sifra

def odsifruj(sprava, kluc):
    text = ""
    modulo = len(kluc)
    i = 0
    for znak in sprava:
        text += chr(ord(znak) - ord(kluc[i%modulo]))
        i += 1
    return text

# =====

sprava = input("Zašifrovať: ")
zasifrovana_sprava= zasifruj(sprava, kluc)
print("Zašifrovaná správa:", zasifrovana_sprava)

odsifrovana_sprava= odsifruj(zasifrovana_sprava, kluc)
print("Odšifrovaná správa:", odsifrovana_sprava)

```

**Ukážka:**

```

Zašifrovať: Kontakt dnes 21:30 miesto C. # 28 znakov
Zašifrovaná správa: -Û×ÖÄÍÔÐ×ÇÔ;İÊÆBÝŇŸ # 19 znakov
Odšifrovaná správa: Kontakt dnes 21:30 miesto C.

```

Poradové čísla znakov zašifrovanej správy:  $172(=75+97)^1$  219 215 214 194 205 213 129 208 215 199 212 130 147 146 166 156 146 129 207 202 198 223 221 209 129 165 143. Prečo zobrazený počet znakov zašifrovanej správy nesúhlasí s počtom znakov šifrovanej správy?

Riešenie zašifrovania správy študentom Dominikom Čiernym:

```
import math

sprava = input("Zadaj správu: ")
kluc = "alibaba"

x = len(sprava) / len(kluc)
klucx = kluc * math.ceil(x)          # "násobenie" reťazca kluc podľa dĺžky správy
                                     # funkcia ceil(x) zaokrúhli reálne číslo x nahor na celé číslo

print("Kľúč: {}".format(klucx))

zasifrovane = ""

for i in range(len(sprava)):
    znak = sprava[i]
    klucovyZnak = klucx[i]
    zasifrovanyZnak = ord(znak) + ord(klucovyZnak)
    zasifrovane += chr(zasifrovanyZnak)

print("Pôvodná správa: {}".format(sprava))
print("Zašifr. správa: {}".format(zasifrovane))
```

Poznámka:

V programe zvýraznené príkazy možno nahradiť aj príkazmi:

```
x = len(sprava) // len(kluc) + 1
klucx = kluc * x
```

**Úloha 9\*:** Vytvorte funkcie, ktoré overia, či zadané rodné číslo je správne (deliteľné 11 bezo zvyšku), určia pohlavie (muž, žena) a dátum narodenia. Rodné číslo má tvar RRRMMDDxxxx, kde RR je posledné dvojčíslicie roku narodenia, MM je u mužov mesiac narodenia (pri jednocifernom mesiaci zľava doplnený nulou), u žien je k ich mesiacu narodenia pripočítané číslo 50; DD je deň narodenia (pri jednocifernom dni zľava doplnený nulou); xxxx sú štyri čísla vybrané tak, aby celé rodné číslo bolo deliteľné bezo zvyšku 11. Napríklad r.č. = 0062022532 je správne r.č. ženy narodenej 2.12.2000; číslo = 0152291436 je síce deliteľné bezo zvyšku 11, ale obsahuje neexistujúci deň 29.2.2001!

Poznámka: Uvedené riešenie je komplexnejšie a teda aj zložitejšie. Je postačujúce, ak si úlohu zjednodušíte a budete overovať len deliteľnosť zadaného čísla 11 a z neho určíte pohlavie a dátum narodenia, pričom, ak je dvojčíslicie roku narodenia väčšie ako 15, predpokladajte rok narodenia 19RR inak 20RR.

```
import datetime                                     # import knižnice pre určenie aktuálneho - systémového dátumu
from calendar import monthrange                   # import funkcie monthrange (vysvetlené nižšie)

def jeRCspravneBezKontrolyDatumu(rc):
    # ak r.č. nemá 10 znakov alebo neobsahuje len číslice - chyba
    if len(rc) != 10 or not rc.isdigit(): return False
    return int(rc)%11 == 0

def vratDenMesRok(rc):
    dnes = datetime.datetime.now() # vráti aktuálny dátum a čas v tvare rrrr-mm-dd hh:mm:ss.xxxxxx
    dnesRRMMDD = str(dnes).replace("-", "") [2:8]

    den = rc[4:6]
    RRRMMDDrc = den # postupné vytvorenie RRRMMDD z r.č. pre porovnanie s akt.dátumom
    if den[0] == "0": den = den[1]
```

<sup>1</sup> Pozri napríklad Microsoft Word – Vložiť – Symbol – Ďalšie symboly... – Písmo: (normálny text) – kliknúť K – Kód znaku: 75 z: ASCII (desiatkovo).

```

mes = rc[2:4]
if mes[0] == "5": mes = "0" + mes[1]
if mes[0] == "6": mes = "1" + mes[1]
RRMMDDrc = mes + RRMMDDrc
if mes[0] == "0": mes = mes[1]

r;ok = rc[:2]
RRMMDDrc = rok + RRMMDDrc
if RRMMDDrc <= dnesRRMMDD:
    rok = "20"+rok
else:
    rok = "19"+rok

den = int(den)
mes = int(mes)
rok = int(rok)

return (den, mes, rok)          # funkcia vráti n-ticu

```

'''

Funkcia **monthrange(rok, mesiac)** z knižnice calendar vracia dvojicu: na indexe 0: číslo dňa v týždni, na ktorý pripadne prvého v danom mesiaci (pondelok - 0, ..., nedela - 6); na indexe 1: maximálny počet dní v danom mesiaci v danom roku. Autorom tohto elegantného riešenia - použitia funkcie monthrange je študent Dominik Čierny.

'''

```

def jeSpravnyPocetDnivMesiaci(rc):
    den, mesiac, rok = vratDenMesRok(rc)
    maxDnivMesiaci = monthrange(rok, mesiac)[1]
    dni = int(rc[4:6])
    return dni <= maxDnivMesiaci

def jeRCspravne(rc):
    if len(rc) != 10 or not rc.isdigit(): return False
    return (int(rc)%11 == 0 and jeSpravnyPocetDnivMesiaci(rc))

```

```

def vratPohlavie(rc):
    if jeRCspravne(rc):
        if rc[2]=="0" or rc[2]=="1":
            return "muž"
        elif rc[2]=="5" or rc[2]=="6":
            return "žena"

```

# ===== HLAVNÝ PROGRAM RČ =====

'''

Príklady rodných čísel na testovanie:

```

0011292545 OK, muž, 29.11.2000
160228/1670 OK, muž, 28.2.1916
0153273373 OK, žena, 27.3.2001
0062022531 nie je deliteľné 11
0011312455 neexistujúci dátum 31.11., deliteľné 11
0152291436 neexistujúci dátum 29.2.2001, deliteľné 11
'''

```

**Použitie:**

```

rc = "200229/1269"
rc = rc.replace('/', '')          # odstráni prípadnú lomku v zadanom rodnom čísle
print("RČ:", rc)
if jeRCspravne(rc):
    print("Pohlavie:", vratPohlavie(rc))
    den, mesiac, rok = vratDenMesRok(rc)
    print("Dátum narodenia: {den}.{mesiac}.{rok}".format(**locals()))
else:
    print("Chyba v rodnom čísle!")

```



**vypíše**

RČ: 2002291269

Pohlavie: muž

Dátum narodenia: 29.2.1920