Programska potpora komunikacijskim sustavima

Dr. sc. Adrian Satja Kurdija

Programski jezik Python - 3. predavanje

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Sadržaj predavanja

- Funkcije
- Stringovi
- For petlja, range
- Komprehenzija
- Rad s datotekama
- Stdin/stdout
- Doseg varijabli

Funkcije

Funkcije

- Pomoćni dio programa koji: prima nula ili više objekata (npr. niz brojeva), obavlja neki zadatak i (neobavezno) vraća rezultat dijelu programa koji je pozvao funkciju
- Primjer: funkcija koja pretvara broj u niz znamenaka

```
def f(x):
    znamenke = []
    while x > 0:
        znamenke.append(x % 10)
        x //= 10
    return znamenke
...
n = int(input())
lista = f(n)
```

Stringovi

Stringovi

- Znakovni nizovi - navode se unutar jednostrukih ili dvostrukih navodnika

```
ime = 'Joe'
```

- Dohvaćanje elementa (ime[k]), slicing, len isto je kao za listu
- Brojenje/traženje znaka ili podstringa:

```
s.count(podrijec), s.index(podrijec)
```

- *Immutable tip* - Ne možemo mijenjati pojedini znak; ako to želimo stvaramo novi string:

```
s = s[0:5] + 'A' + s[6:]  # 5. znak postaje A
```

Stringovi

- Rastavljanje u listu stringova prema separatoru:

```
s.split()
s.split(znak)
s.split(',')
```

- Spajanje više stringova u jedan:

<separator>.join(niz_stringova)

```
>>> ' ? '.join(['prva', 'druga', 'treca'])
'prva ? druga ? treca'
```

Stringovi

- Stringove odvojene razmakom (ili nekim drugim separatorom) možemo učitati u listu stringova:

```
a = input().split()
```

- Ako je riječ o brojevima:

```
a = list(map(int, input().split()))
```

- Tri broja u istom retku:

```
x, y, z = map(float, input().split())
```

Zadatak

Napisati funkciju koja prima rečenicu te vraća njenu **kraticu** tako da redom za svaku riječ iz rečenice uzmemo prvo slovo, osim ako riječ ima jedno ili dva slova. Kraticu pišemo velikim slovima. (npr. Public relations → PR)

- str.split()
- str.upper(), str.lower()

Zadatak

- Napisati funkciju koja prima rečenicu te vraća postotak riječi u rečenici koje nisu veznici ('i', 'pa', 'te', 'ni', 'niti', 'a', 'ali', 'nego', 'no', 'ili')
- Napisati funkciju koja prima rečenicu te vraća broj slova (duljinu riječi) za sve riječi u rečenici koje nisu veznici
  - Izlaz je mapa gdje su ključevi riječi, a vrijednosti duljina riječi

Vježba

- Napisati funkciju koji iz dane riječi izbacuje sve samoglasnike.

- Napisati funkciju koja prima string te vraća novi string tako da su sva mala slova zamijenjena velikim slovima, a sva velika slova zamijenjena malim slovima.

- Napisati funkciju koji će danu riječ ispisati tako da ubaci razmak između svakih dvaju susjednih slova riječi
(npr. *Python → P y t h o n*).

- Napisati funkciju koji učitava dva stringa te odgovara na pitanje može li se drugi string dobiti premetanjem znakova prvog stringa.

Zadatak

- Pomoću naredbe `help(str)` i `help(str.metoda)` proučiti ostale metode string objekata
- Napisati funkciju koja prima rečenicu i vraća istu rečenicu, ali
  - Bez višestrukih praznina
  - Ispravljenog prvog početnog slova (ako je dano malim slovom)
  - Dodana točka na kraj (ako ne postoji)
  - npr. **“ovo je neki   tekst”** → **“Ovo je neki tekst.”**

Range, komprehenzija

For petlja i range

- Funkcija range generira *n*-torku cijelih brojeva
  - Dohvaćanje raspona brojeva
  - Po njemu je moguće iterirati for petljom

```
range(pocetak, kraj, korak)
```

- Na primjer, sljedeći odsječak koda generira brojeve 1, 3, 5, …, 19 i ispisuje ih:

```
for i in range(1, 20, 2):
    print(i)
```

- Moguće je navesti samo *kraj* (za niz 0, 1, 2, …, kraj - 1):

```
for i in range(10):
    print(i)
```

- Primjer - obilazak liste:

```
for i in range(len(lista)):
    print(i, lista[i])
```

Komprehenzija

- Želimo stvoriti listu na osnovi nekog postojećeg niza
- "Dulji" način: for petljom prođemo po postojećem nizu i dodajemo elemente u novu listu (*append*)

```
for i in range(10):
    novaLista.append('Number ' + str(i))
```

- Kraći način: *list comprehension*

```
nova = ['Number ' + str(i) for i in range(10)]
```

- Općenito: `[<izraz> for <varijabla> in <niz>]`
- Primjeri:
  - unos više redaka podataka: `redci = [input() for i in range(n)]`
  - pretvorba stringa u listu slova:

```
h_letters = [letter for letter in 'human']
print(h_letters)
```

Datoteke, stdin/stdout

Rad s datotekama

- Čitanje/pisanje u datoteku

f = open(path, 'r|w') - otvara datoteku za čitanje/pisanje
f.read() - vraća cijeli sadržaj kao string
f.readline() - vraća idući redak kao string
f.readlines() - vraća retke kao listu stringova
f.write() - zapisuje string u datoteku
f.writelines() - zapisuje listu stringova kao retke
f.close() - zatvara datoteku

Rad s datotekama

- Iteracija po datoteci redak po redak

```
with open(path) as f:
    for line in f:
        print(line)
```

Stdin/stdout

- Prilikom pokretanja programa iz komandne linije (Terminal, Command Prompt, PowerShell) možemo datoteku s ulaznim podatcima preusmjeriti na standardni ulaz (stdin) operatorom redirekcije ulaza (znak <):

```
$ python obrada.py < podatci.txt
```

- Tada program učitava podatke kao “s tipkovnice” (*input()* i slično)
- Za ispis na standardni izlaz (stdout) koristimo print(), ali i njega možemo preusmjeriti u datoteku (znak >):

```
$ python obrada.py < podatci.txt > izlaz.txt
```

Vježba

- Napišimo program koji iz tekstualne datoteke učitava retke s podatcima oblika:
  - *ime prezime, OIB*
- Program treba ispisati:
  - broj osoba (redaka),
  - osobe sortirane po prezimenu silazno,
  - najčešće ime.
- Program napišimo kao funkciju koja prima ime tekstualne datoteke iz koje čita podatke.
- Prilagodimo funkciju da ako dobije prazno ime datoteke, čita sa standardnog ulaza.

Doseg varijabli - primjer

```
def scope_test():
    def do_local():
        x = "local x"

    def do_nonlocal():
        nonlocal x
        x = "nonlocal x"

    def do_global():
        global x
        x = "global x"

    x = "test x"
    do_local()
    print("After local assignment:", x)     # ispis: test x
    do_nonlocal()
    print("After nonlocal assignment:", x)  # ispis: nonlocal x
    do_global()
    print("After global assignment:", x)    # ispis: nonlocal x

scope_test()
print("In global scope:", x)              # ispis: global x
```