VEŽBA 5

Pregled

Python je savremeni skript jezik koji danas ima široku primenu u mnogim oblastima. Velika prednost je što programi napisani u ovom programskom jeziku ne zahtevaju prevođenje, nego se mogu izvršavati neposredno iz oblika programskog teksta – izvornog koda (engl. *source code*).

Korišćenje sistema ulaza/izlaza prema terminalu

izlaz prema terminalu – ispis

Za ispis na terminalu koristi se naredba `print` (u Pythonu 2 ovo *nije* funkcija, u Pythonu 3 *jeste*). Formatirani ispis se obavlja za sve tipove podataka bez njihovog posebnog obeležavanja. Jednom naredbom se može ispisati i veći broj podataka i oni se tada odvajaju *zarezom*. Slobodan zarez na kraju reda onemogućava prelazak na novi red nakon ispisa.

```
print 'Broj je:', x
print 'Unesite broj: ',  # ne prelazi u novi red
```

ulaz sa terminala – unos

Ulaz sa terminala obavlja se ugrađenom funkcijom `input` (ovo je funkcija u svim verzijama Pythona). Ova funkcija interpretira unetu vrednost pa je moguće uneti direktno i sve strukture podataka poput npr. lista ako se koristi odgovarajuća sintaksa. Sa druge strane, neispravni brojevi ili strukture podataka sa neispravnom sintaksom izazivaju grešku, a stringovi se moraju unositi sa navodnicima. Zbog toga je za unos stringova praktičnije koristiti funkciju `raw_input`.

```
br = input()   # pogodno za brojeve i strukt. podataka
s=raw_input()  # pogodno za unos stringova jer ne trazi navodnike
```

Rad sa stringovima

String je jedan od osnovnih tipova podataka u jeziku Python i služi za skladištenje nizova karaktera, odnosno teksta. Jednom kreiran string se više ne može menjati i zbog toga se svrstava u *nepromenljive* tipove podataka (*immutable*).

kreiranje stringova

Navođenjem teksta između jednostrukih ili dvostrukih navodnika definiše se string. Ova dva tipa navodnika potpuno su ravnopravna.

```
s='Oba tipa navodnika su "ravnopravna"'
p="Drugi tip moze se koristiti u stringu: pyton's challenge"
```

pristup elementima i delovima stringa

Operator *[start:stop:step]* primenjen na string daje novi string sastavljen od elemenata stringa na koji je operator primenjen. Svi elementi stringa su indeksirani rednim brojem, a prvi element ima indeks *0*. Negativni indeksi broje karaktere od kraja stringa i indeks *-1* se odnosi na poslednji karakter sitringa. U gorenavedenom formatu *start* se odnosi na prvu poziciju koja će biti uključena u rezultat, *stop* na prvu poziciju koja neće biti uključena u rezultat, a *step* na korak kojim će biti uzimani elementi iz navedenog opsega. Ako se neki od parametara ne navede koristiće se podrazumevane vrednosti a to su prvi karakter za početak, poslednji za kraj, a *1* za korak.

```
s='0123456789'
s[3:6]    # '345'
s[:6:2]   # '024'
s[::3]    # '0369'
```

nadovezivanje stringova

Rezultat ovog „sabiranja“ stringova je novi string dobijen nadovezivanjem stringova koji u ovoj operaciji učestvuju.

```
a='prvi'+' drugi' # novi string: 'prvi drugi'
m='a'+'b'+'c' # novi string: 'abc'
m+=a # novi string: 'abcprvi drugi'
```

Konverzija drugih tipova podataka u string

Funkcija *str()* konvertuje bilo koji tip podatka poznat Python-u u string, ukoliko je takva konverzija moguća. Ukoliko nije moguća, prijavljuje se greška.

```
x=str(100)     # vraća string '100'
```

Rad sa listama

Liste su *promenljive* (*mutable*) strukture podataka tako da je postojeću listu moguće produžavati, dodavati nove elemente na proizvoljna mesta, kao i uklanjati proizvoljne elemente iz nje. U nastavku sledi jedan izbor mogućih operacija sa listama. Osim njih postoje i mnoge druge.

zadavanje liste

Lista se zadaje navođenjem podataka unutar uglastih zagrada, a podaci se odvajaju zarezom. Prazna lista se obeležava kao *[]*.

```
ls = [3, 6, 12]
pl = []   # prazna lista
```

pristup elementima liste

Pristup elementima liste moguć je preko indeksa (pozicije u listi) u uglastim zagradama slično kao i kod stringova. Za razliku od stringova, kod listi je na ovaj način moguće i menjati pojedinačne elemente.

```
x = [1, 2, 3, 4]
x[3]      # rezultat je 4
```

```
x[-2]     # rezultat je 3
x[1] = 10 # rezultat je promenjena lista: [1, 10, 3, 4]
```

kreiranje nove liste na osnovu indeksa elemenata

Notacija [start:stop:step] je i ovde moguća kao i kod stringova i funkcioniše na isti način. Rezultat operacije je nova lista.

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
a[3:6]    # rezultat je [4, 5] jer indeks 6 ne ulazi u izbor
a[1:8:2]  # rezultat je [2, 4, 6, 8]
a[1:-2:2] # rezultat je [2, 4, 6]
```

spajanje dve liste (ili više njih)

Operator + primenjen na liste stvara novu listu koja je spoj tih listi.

```
nl = [8, 15, 20, 9]
sl = ls + nl   # sl: nova lista [3, 6, 12, 8, 15, 20, 9]
nl += ls; # nl: [8, 15, 20, 9, 3, 6, 12]
```

uklanjanje elementa liste

Naredba *del* služi za uklanjanje promenljivih. Kada se primeni na element liste, taj element se briše iz liste.

```
del ls[1]
```

utvrđivanje dužine liste

Ugrađena funkcija len vraća dužinu svhi složenih struktura podataka. U slučaju liste to je broje elemenata liste.

```
dl = len(ls)
```

dodavanje na kraj liste

Ova funkcija je *metoda* objekta tipa liste i poziva se pomoću operacije *tačke*. Argument je element koji treba dodati na kraj liste.

```
ls.append(100)
```

uklanjanje iz (sa kraja) liste

Metoda *pop* vraća vrednost elementa sa pozicije navedene kao argument metode, a istovremeno i briše navedeni element.

```
ls.pop(2)
ls.pop(-1)     # uklanja poslednji element liste
```

dodavanje u listu

Metoda *insert* dodaje element u listu na proizvoljnom mestu. Kao prvi argument se navodi pozicija u listi, a kao drugi podatak koji se u listu smešta.

```
ls.insert(1, 220)
```

Upravljačke strukture *if* i *while* jezika Python

Provera uslova i petlje suštinski su deo svakog programskog jezika. Tako je i kod Pythona. U odnosu na druge jezike Python poseduje neke osobenosti.

Provera uslova sa if

If kao i kod drugih imperativnih jezika služi za odlučivanja u toku programa. U Pythonu ima sledeću formu:

```
if uslov_1:
  blok_1
elif uslov_2:
  blok_2
else:
  blok_3
```

Uslovi navedeni uz *if* i *elif* su bilo koji izraz koji se izračunava u neku vrednost. Smatra se tačnim u svim slučajevima osim kada je ta vrednost: *False*, *0*, *0.0*, *''*, *[]*. Poslednja dva elementa su prazan string i prazna lista. Postoje i druge mogućnosti, ali one ovde neće biti pomenute.

Blok 1 se izvršava ako je uslov 1 zadovoljen, blok 2 ako uslov 1 nije, ali uslov 2 jeste, a blok 3 ako nijedan od prethodnih uslova nije zadovoljen. *elif* i *else* segmenti su opcioni, dok *elif* segmenata može biti i više od jednog ako za tim postoji potreba.

Python je osoben po tome što se blokovi raspoznaju i istovremeno i definišu nivoom uvučenosti od desne ivice – indentacijom. Posebna oznaka za početak i kraj bloka ne postoji. Na ovaj način programer je nateran da ispravno formatira svoj program.

Petlje tipa while

I while petlja je uobičajena konstrukcija u programskim jezicima. U pythonu ima sledeći oblik:

```
while uslov:
  blok_1
else:
  blok_2
```

Uslov se ponaša identično uslovu u *if* konstrukciji. Dok je ispunjet telo petlje označeno sa *blok_1* se izvršava. Prvi put kada uslov nije tačan izvršava se *blok_2*, i to samo jednom.

Prekidanje i vraćanje na početak petlji

Svaka petlja u Python-u može da se prekine naredbon *break*, ili da se vrati na mesto provere uslova naredbom *continue*. Ovim blok *else* iz prethodnog odeljka dobija nešto više smisla jer izlazak iz petlje pomoću *break* otvara mogućnost prekidanja *while* petlje, a da je sam uslov petlje pri tome ispunjen. *break* i *continue* se skoro uvek koriste zajedno sa uslovom tipa *if*.

Zadaci

1. Pokrenuti Python sa terminala i isprobati osnovnu funkcionalnost u neposrednom modu.. Dodelti vrednosti tipa `int`, `float` i `string` promenljivama po slobodnom izboru i proveriti njihov sadržaj. Od sadržaja tih promenljivih napraviti listu i dodeliti je novoj promenljivoj. Proveriti razliku u ispisu za sve promenljive kad se koristi print naredba i kada se ispis dobija čistim navođenjem imena promenljive.
2. Napisati program i snimiti ga pod imenom `hello.py`. On treba da ispiše string *„Hello World!“*. Nakon što uspešno proradi, izmeniti program tako da u petlji 10 puta redom ispiše navedeni string.
3. Napisati skript koji omogućava sledeće: korisnik preko tastature unosi broj `x`, a odmah potom računar izračunava vrednost `3 * x + 5` i ispisuje je u sledećem redu na ekranu. Nakon toga očekuje se unos sledećeg broja. Postupak se ponavlja sve dok korisnik ne unese vrednost 0.

   Primer:

   ```
   Unesite broj X : 5
   (3 * X) + 5  je : 20
   Unesite broj X : 7
   (3 * X) + 5  je : 26
   Unesite broj X : 0
   Kraj!
   ```

4. Napisati program u Python-u koji omogućava korisniku da unese proizvoljan string preko terminala, a onda će taj string biti ispisan na terminalu:
   a) u izvornom obliku
   b) karakter po karakter
   c) karakter po karakter u obrnutom redosledu
   d) od njega će biti kreiran string u obrnutom redosledu karaktera i ispisan u celini.
5. Napisati program u Python-u koji omogućava korisniku da unese proizvoljni niz brojeva koji se skladište u python listi. Sortirati listu u neopadajući redosled metodom uklanjanja najmanjeg elementa iz liste i ubacivanjem tog elementa na kraj nove (u početku prazne) liste. Ispisati elemente liste u sortiranom obliku element po element.