# 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*).

Ovakav način rada moguć je zahvaljujući Python virtualnoj mašini koja se pokreće pozivom komande `pyhton`. Virtuelna mašina ima w načina rada:

- REPL mod – zadavanjem komande `python` u terminalu bez dodatnih argumenata pokreće se interaktivni način rada, tzv. REPL mod (Read-Eval-Print-Loop). Svaka uneta komanda se odmah izvršava, a rezultati se ispisuju na terminalu. Ovaj način rada je pogodan za istraživanje i testiranje. Sistem ispisuje znak *>>>* kada je spreman za prijem nove komande.
- Programski mod – argument komande python se tumači kao ime fajla koji sadrži python program. Taj fajl se čita, proverava se njegova sintaksna i strukturna ispravnost, a ako se pritom ne nađe nijedna grešaka, program se izvršava. Greške nastale u toku izvršavanja programa (run-time errors) će obustaviti izvršavanje programa uz ispisivanje odgovarajuće poruke o grešci. Pokretanje programa test.py postiže se na sledeći način

  ```
  python test.py
  ```

## Format programa pisanog u jeziku Python

Za razliku od jezika C, Python ne zahteva raspoređivanje kompletnog izvršnog dela programa u podprograme, odnosno funkcije. Kao što će se videti već u sledećem odeljku, funkcije se često koriste u jeziku Python, ali to nije obavezno. Sve komande navedene u fajlu koji sadrži program, biće izvršene redom (osim ako su delovi funkcija, ali o tome će biti reči kasnije).

Komentar se (kao i u shell skriptovima) započinje karakterom `#`. Sve što se nađe iza ovog karaktera do kraja reda se zanemaruje i može da služi kao komentar.

## Osnovni tipovi podataka

Osnovni tipovi su `int`, `float`, `bool` i `str`. Ovaj poslednji će biti detaljnije objašnjen u odeljku koji sledi (Rad sa stringovima).

| | |
|---|---|
| `int` | celi brojevi – za razliku od jezika C, Python ne postavlja ograničenje na veličinu broja, čime se dobija praktično neograničen opseg |
| `float` | brojevi u pokretnom zarezu – svaki broj koji ima razlomljeni deo, automatski će biti svrstan u ovu kategoriju |
| `bool` | logički tip – tačno ili netačno – odgovaraju im ključne reči `True` i `False` |
| `str` | string – pojedinačni karakteri ili nizovi karaktera (tekst) |

# Promenljive

Programi se pišu da bi se automatizovano obrađivali podaci, a u toku obrade je neophodno obezbediti pristup tim podacima. Osnovni tipovi podataka pomenuti su u prethodnom poglavlju, a pristup tim podacima obezbeđen je putem promenljivih.

Ključna operacija sa promenljivama je *dodela*. Dodela se obavlja operatorom =, pri čemu se sa leve strane navodi promenljiva (njeno ime), a sa desne vrednost koja se dodeljuje. Vrednost može da bude konstanta ili proizvoljan izraz koji uključuje aritmetičke operacije, pozive funkcija itd.

```
x = 150
a = 300 * 3 + x
x = a
```

**Kreiranje promenljivih**

Promenljiva se kreira onog trenutka kada se u programu izvrši prva operacija dodele. Pre tog trenutka promenljiva ne postoji i pokušaj pristupanja rezultuje prekidom izvršavanja programa (run-time error).

**Dinamički sistem tipova**

Za razliku od jezika kod kojih se svaka promenljiva za ceo svoj životni vek povezuje sa određenim tipom podatka (*strongly-typed languages*), kod Python-a se svakoj promenljivoj u bilo kom trenutku operacijom dodele može dodeliti podatak bilo kog tipa – ovakav sistem se naziva *dinamičkim* (*dynamically-typed languages*).

```
q = 10    # q ukazuje na podatak int
w = 3.8   # w ukazuje na podatak float
q = w     # sada q ukazuje na podatak float
```

Praktično, promenljiva je samo registrovano ime koje tokom svog životnog veka može da se poveže sa proizvoljnim podatkom. Sam podatak određuje tip, a ne promenljiva. Možemo o tome razmišljati kao „promenljiva ukazuje na podatak“, a sam podatak je nezavisan objekat.

# Aritmetičke i druge operacije

Izvođenje operacija se izvodi navođenjem konstanti ili promenljivih i odgovarajućih operatora, kao što je prikazano u prethodnom primeru. Pisanje izraza na ovaj način je prilično intuitivno i gotovo identično operacijama u jeziku C, pa ovde neće biti do detalja razmatrano.

Treba skrenuti pažnju da operacije ++ i - - poznate (i omiljene) iz jezika C u Python-u nisu na raspolaganju. Sa druge strane, operacije op= jeste, pa je moguće sledeće:

```
x += 1    # <=> x = x+1
a *= 2    # <=> a = a*2
```

# Korišćenje sistema ulaza/izlaza prema terminalu

Najosnovnija interakcija sa korisnikom obavlja se preko terminala – po pravilu istog onog sa kojeg je program pokrenut.

**izlaz prema terminalu – ispis**

Za ispis na terminalu koristi se ugrađena funkcija `print`. U pitanju je tzv. *ugrađena funkcija* koju sistem prepoznaje bez dodatne pripreme. 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*, odnosno svaki podatak je poseban argument funkcije i može ih biti proizvoljno mnogo.

```
print('Dobrodosli u Python!')
print('Broj je:', x);
```

Navođenje znaka tačka-zarez na kraju operacije/naredbe je dozvoljeno, ali ne i neophodno.

Svi argumenti se nakon ispisa odvajaju *separatorom*, a nakon poslednjeg ispisanog argumenta ispisuje se *sekvenca za kraj*. Podrazumevani separator je prazno mesto, a podrazumevana sekvenca za kraj pralazak na novi red. Ove dve stvari se mogu izmeniti zadavanjem imenovanih argumenata `sep` i `end` na sledeći način:

```
# sledeca operacija ce na terminalu ispisati: A*B*C#
print('A', 'B', 'C', sep='*', end='#')
# sled. op. ce ispiati 'Unesi broj:' BEZ prelaska na novi red
print('Unesite broj: ', end='')
```

Imenovani argumenti poput `sep` i `end` pri pozivu funkcije se obavezno navode na kraju, nakon svih ostalih argumenata.

**ulaz sa terminala – unos**

Ulaz sa terminala obavlja se ugrađenom funkcijom `input`. Ova funkcija sa terminala očitava sve karaktere onako kako su uneti (unos se završava pritiskom na ENTER) i vraća podatak koji je tipa `string`.

Veoma često je potrebno uneti brojeve na ovaj način. Pošto funkcija input može da vrati jedino stringove, ovako dobijeni stringovi se mogu konvertovati u brojeve tipa int i float pozivom istoimenih funkcija sa argumentom koji je string. Treba napomenuti da ako se uneti string ne može interpretirati kao broj, program će se zaustaviti uz ispis poruke o grešci pri konverziji.

```
s = input()          # pogodno za unos stringova
i = int(s)           # pokusaj konverzije stringa u int
f = float(input())   # unos float broja – kompaktniji zapis
```

# 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*). Manipulacija stringovima je moguća, ali se pri tome ne menja originalni string, nego se kreira novi koji sadrži sve napravljene izmene.

**kreiranje stringova**

Navođenjem teksta između jednostrukih ili dvostrukih navodnika definiše se string. Ova dva tipa navodnika potpuno su ravnopravna. Razlog za ovo je stvaranje mogućnosti da se onaj drugi tip

navodnika može sadržati u kreiranom stringu (kako pokazuje primer koji sledi):

```
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*, i *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 (konkatenacijom) 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.

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

# 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 – *if* struktura**

*if* je upravljačka struktura koja kao i kod drugih imperativnih programskih 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. Rezultat se

*smatra 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 (prazne strukture podataka npr.), ali one ovde neće biti razmatrane.

`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 ispunjen 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 naredbom `break`, ili da se vrati na vrh (mesto provere uslova) naredbom `continue`. Pri izlasku iz petlje naredbom `break`, `else` blok se preskače. 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 REPL (neposrednom, interaktivnom) modu. Dodeliti vrednosti promenljivama `a` i `b`, a promenljivu `c` izračunavati kao `c = (a+b)/2`.

   a) dodeljivati celobrojne vrednosti promenljivama `a` i `b`. Kakvog su tipa rezultati?

   b) dodeljivati vrednosti sa razlomljenim delom promenljivama `a` i `b`. Kakvog su tipa rezultati?

   *Napomena*: tip promenljive (podatka) može se proveriti pozivom funkcije `type(arg)`. Primetiti da u REPL modu nije neophodno koristiti funkciju `print` za ispis, svaka operacija koja ima rezultat, odmah će ga ispisati na terminalu.

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 za redom ispiše navedeni string.

3. Napisati program 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. Koristiti upravljačke strukture *if* i *while*!

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.

   *Napomena*: Koristiti funkciju input i opcije za rad sa stringovima.