METODIKA NASTAVE RAČUNARSTVA A

Python – uvodni zadaci

Stefan Mišković

Python je na Linuxu automatski instaliran. U bilo kom tekstualnom editoru se Python program može saču- vati sa ekstenzijom py, a zatim pokrenuti preko terminala (komandne linije) naredbom python naziv.py. Inače, i za Windows i za Linux se može instalirati bilo koji Python IDE, na primer *PyCharm Community*, koji se za Windows može preuzati sa [ovog](https://www.jetbrains.com/pycharm/download/#section%3Dwindows) linka, a za Linux sa [ovog](https://www.jetbrains.com/pycharm/download/#section%3Dlinux) linka. Može se koristiti i neki od online editora, poput [ovog](https://www.onlinegdb.com/online_python_compiler), [ovog](https://www.tutorialspoint.com/execute_python_online.php) ili [ovog](https://repl.it/repls/ThornyBuzzingRedundancy).

# PROMENLJIVE I OPERATORI

1. Napisati program koji na standardni izlaz ispisuje *Zdravo svete!*.

*Rešenje.* Funkcija print se koristi za ispisivanje teksta na standardni izlaz.

**print** ( " Zdravo. s v e te ! " )

1. Napisati program koji ispisuje vrednost promenljive na standardni izlaz.

*Rešenje.* Promenljive u Pythonu nisu tipizirane, već se iz konkteksta zaključuje o kom tipu je reč. Tako možemo imati neku celobrojnu ili realnu promenljivu, čiji tip nije eksplicitno naveden. U rešenju zadatka su dodati i jednolinijski komentari, koji počinju znakom #.

*# ova promenljiva j e celobrojna ( i n t t i p u C u)*

−

ceo Broj = 7

**print** ( ceo Broj )

*# ova promenljiva j e realna ( f l o a t t i p u C u)*

−

r e a l a n Br o j = 10 . 54

**print** ( r e a l a n Br o j )

1. Ispisati celobrojnu i realnu promenljivu u sklopu stringa na standardni izlaz.

*Rešenje.* Za ispisivanje brojevne promenljive u okviru teksta na standardni izlaz, potrebno je kastovati njen tip u string preko funkcije str. Nadovezivanje se vrši pomoću operatora +.

a = b = 6 *# mogu se d o d e l i t i v rednosti i na ovaj nacin*

c = 12 . 3

**print** ( " Celobrojna . promenljiva .a.=." + **str** ( a ) ) **print** ( " Celobrojna . promenljiva .b.=." + **str** ( b )) **print** ( " Realna . promenljiva . c .=." + **str** ( c ))

1. Definisati promenljive celobrojnog, realnog tipa i tipa string. Ispisati tipove tih promenljivih na stan- dardni izlaz. Zatim izvršiti kastovanje, gde god je to moguće.

*Rešenje.* Iako Python nije sktriktno tipiziran jezik, funkcijom type se može odrediti tip promenljive. Celo- brojna vrednost se može kastovati u realnu vrednost i u string. Realna se može kastovati u string, a pri kastovanju u celobrojnu vrednost, briše se razlomljeni deo broja. String se može kastovati u ceo ili realan broj, ukoliko je njegov zapis podesan.

ceo Broj = 5

r e a l a n Br o j = 10 . 9 s t r i n g 1 = " 15 "

s t r i n g 2 = " 15 . 5 "

s t r i n g 3 = " zdravo "

**print** ( **type** ( ceo Broj )) **print** ( **type** ( r e a l a n Br o j ) ) **print** ( **type** ( s t r i n g 1 )) **print** ( **type** ( s t r i n g 2 )) **print** ( **type** ( s t r i n g 3 ))

**print** ( **f loat** ( ceo Broj ))

**print** ( **str** ( ceo Broj ))

*# ovde ce b i t i odsecen raz l oml j en i deo broja*

**print** ( **int** ( r e a l a n Br o j ) ) **print** ( **str** ( r e a l a n Br o j ) ) **print** ( **int** ( s t r i n g 1 )) **print** ( **f loat** ( s t r i n g 1 )) **print** ( **f loat** ( s t r i n g 2 ))

*# s l ed ece naredbe b i p r i j a v l j i v a l e gresku , pa su pod komentarom # prin t ( i n t ( s t r i n g 2 ))*

*# prin t ( i n t ( s t r i n g 3 ))*

*# prin t ( f l o a t ( s t r i n g 3 ))*

1. Definisati brojevne promenljive i nad njima izvršiti računske operacije sabiranja, oduzimanja, množenja, deljenja, stepenovanja i modula.

*Rešenje.* Osnovne računske operacije su u Pythonu definisane na sledeći način:

* + sabiranje: +,
	+ oduzimanje: −,
	+ množenje: ∗,
	+ deljenje: */*,
	+ celobrojno deljenje: *//*,
	+ stepenovanje: ∗∗,
	+ modul: %.

a = 9

b = 9 . 2

c = 2

**print** ( a + b ) **print** ( a c ) **print** ( a b c )

∗ ∗

−

*# za dva cela broja u pythonu 2 ovo j e celobrojno d e l j en j e , # a u pythonu 3 se t r e t i r a kao d e l j en j e real n i h brojeva* **print** ( a / c )

**print** ( b / c ) **print** ( a % c ) **print** ( b % c )

*# za celobrojno d e l j en j e se najcesce k o r i s t i // umesto /*

**print** ( a // c )

**print** ( b // c )

1. Iskombinovati operatore dodele sa aritmetičkim operatorima, radi kompaktnijeg zapisa. Ispisati dobijene vrednosti na standardni izlaz.

*Rešenje.* Umesto naredbe broj = broj + 2, može se koristiti broj += 2. Slično, može se pisati i -=, \*=,

/=, //=, = i \*\*=.

b r o j = 10 **print** ( b r o j ) b r o j += 6

b r o j = 2 **print** ( b r o j ) b r o j = 2

−

∗

b r o j %= 10 **print** ( b r o j ) b r o j = 2 **print** ( b r o j )

∗∗

1. Koristeći operatore poređenja, uporediti vrednosti dva broja. Istinitosne vrednosti ispisati na standardni izlaz.

*Rešenje.* Operatori poređenja u Pythonu su ==, !=, <, >, <= i >=. Vrednosti izraza koji koriste ove operatore su True ili False.

b roj 2 = 2

b roj 3 = 3

je d n a k i = broj 2 == broj 3

r a z l i c i t i = broj 2 != broj 3

**print** ( je d n a k i ) **print** ( r a z l i c i t i ) **print** ( b roj 2 > b roj 3 )

*# i z r az se moze n ap i s a t i i u ovom ob l i k u*

**print** ( 1 < broj 2 < broj 3 < 4 )

**print** ( 4 >= 10 >= 3 )

**print** ( b roj 2 <= b roj 3 )

# STRINGOVI

1. Iskombinovati vrednosti dva stringa koristeći operator konkatenacije.

*Rešenje.* Operator konkatenacije stringova je +. zdravo = " zdravo "

s v e te = ". s v e te "

zdravo Svete = zdravo + s v e te

**print** ( zdravo Svete )

**print** ( " zdravo ." + " svima " )

1. Ispisati string odgovarajući broj puta množeći ga sa prirodnim brojem.

*Rešenje.* U Pythonu se stringovi mogu množiti brojevima, a rezultat predstavlja nadovezani string odgo- varajući broj puta. Na primer, za string *zdravo*, množenjem brojem 3, dobija se *zdravozdravozdravo*.

zdravo = " zdravo " b r o j = 3

**print** ( zdravo ∗ b r o j )

1. Ispisati vrednosti nekih karaktera stringa, koristeći indeksiranje.

*Rešenje.* Karakteru stringa se može pristupiti preko []. Indeksiranje sleva počinje sa 0, a indeksiranje zdesna sa 1. Tako, na primer, za treći karakter sleva indeks je 2, a za pretposlednji karakter indeks je 2. Može se ispisati više karaktera istovremeno, i to na sledeći način:

− −

* + [i:j] – svi karakteri od indeksa i do indeksa j - 1,
	+ [i:] – svi karakteri od indeksa i,
	+ [:j] – svi karakteri do indeksa j - 1,
	+ [:] – svi karakteri stringa.

r e c = " Republika . S r b i ja " prvo Slovo = r e c [ 0 ]

**print** ( prvo Slovo )

*# i s p i s u j e p r e t p os l ed n j e s l ovo*

**print** ( r e c [ 2 ] )

−

*# i s p i s u j e sva s l ova sem prvog i poslednjeg*

**print** ( r e c [ 1 : 1 ] ) prvaRec = r e c [ : 9 ] drugaRec = r e c [ 1 0 : ] **print** ( prvaRec ) **print** ( drugaRec )

−

*# dva nacina i s p i s a celog s t r i n g a*

**print** ( r e c )

**print** ( r e c [ : ] )

1. Proveriti da li se određeni podstring nalazi u nekom stringu.

*Rešenje.* Za proveru da li se podstring nalazi u nekom stringu, koristi se operator in. p od s tr in g 1 = " S r b i ja "

p od s tr in g 2 = " S r b i j e "

r e c = " Republika . S r b i ja " **print** ( p od s tr in g 1 **in** r e c ) **print** ( p od s tr in g 2 **in** r e c )

1. Ispisati dužinu stringa. Razmotriti i slučaj kada je string napisan u više redova.

*Rešenje.* Funkcija koja vraća dužinu stringa je len. Ukoliko string počinje i završava se trostrukim navod- nicima, može se napisati u više redova.

manji String = " ovo. j e . manji . s t r i n g " v e c i S t r i n g = *""" ovo j e v eci s t r i n g . on se moze n ap i s a t i u v i s e redova .*

*pocinje i zavrsava se t rostrukim navodnicima"""*

**print** ( " Manji. s t r i n g : ." + manji String )

**print** ( " Veci . s t r i n g : ." + v e c i S t r i n g )

**print** ( " Duzina. manjeg. s t r i n g a : ." + **str** ( **len** ( manji String ) ) )

**print** ( **len** ( v e c i S t r i n g ) )

1. Dati string, koristeći odgovarajuće funkcije, ispisati svim velikim, odnosno svim malim slovima.

*Rešenje.* Funkcija koja uvećava sva slova stringa je upper, a koja umanjuje lower.

drzava = " Republika . S r b i ja "

**print** ( drzava . lower ( ) ) drzava Velikim Slovima = drzava . upper () **print** ( drzava Velikim Slovima )

1. Razmotriti slučajeve kada se jednostruki ili dvostruki navodnici mogu naći unutar stringa.

*Rešenje.* Jednostruki navodnici se mogu koristiti u stringu koji počinje i završava se dvostrukim, i obratno. (String se ravnopravno može koristiti i ako je naveden i jednostrukim i dvostrukim navodnicima.) Inače, karaktere je potrebno naznačiti sa \’, odnosno \".

s t r i n g 1 = ’ Program. i s p i s u j e : ." Zdravo. s v e te ! " ’ s t r i n g 2 = "Program. i s p i s u j e : . ’ Zdravo. s v e te ! ’ "

s t r i n g 3 = ’ Program. i s p i s u j e : . \ ’ Zdravo. s v e te ! \ ’ ’ s t r i n g 4 = "Program. i s p i s u j e : .\" Zdravo. s v e te ! \ " "

s t r i n g 5 = "Oba. navodnika . ’ . i .\".u. jednom. s tr i n g u "

**print** ( s t r i n g 1 ) **print** ( s t r i n g 2 ) **print** ( s t r i n g 3 ) **print** ( s t r i n g 4 ) **print** ( s t r i n g 5 )

1. Ispisati brojevni tip podatka ili string, koristeći formatiranje u okviru stringa.

*Rešenje.* Ceo broj se formatira pomoću d, realan pomoću f, a string pomoću s. ceo Broj = 6

r e a l a n Br o j = 12 . 3 r e c = " zdravo "

**print** ( "Ceo. b r o j . j e .%d" % ceo Broj )  **print** ( " Realan . b r o j . j e .%f " % r e a l a n Br o j ) **print** ( " S tr i n g . j e .%s " % r e c )

*# i s p i s i se mogu i skombinovati*

**print** ( " Br o je v i . su .%d. i .%f , .a. r e c .%s " % ( ceo Broj , realan Broj , r e c ) )

# STRUKTURE PODATAKA

1. Definisati listu, a zatim ispisati neke od njenih elemenata, koristeći indeksiranje. Ispisati dužinu liste.

*Rešenje.* Lista predstavlja kolekciju podataka, koji mogu biti proizvoljnog tipa. Za razliku od nizova u C-u, u Pythonu se lista može sastojati i od različitih vrsta podataka. Sintaksa za definisanje liste je lista = [element1, element2, element3]. Indeksiranje se vrši na sličan način kao kod stringa. Dužina liste se određuje pomoću funkcije len.

*# u l i s t i mogu b i t i t i p o v i r a z l i c i t o g t i p a*

l i s t a = [ " jedan " , " dva" , 3 , 4 . 5 ]

**print** ( l i s t a ) **print** ( **len** ( l i s t a ) ) **print** ( l i s t a [ 0 ] ) **print** ( l i s t a [ − 1 ] )

1. Izvršiti nadovezivanje dve liste, a zatim i dodavanje, izmenu i brisanje nekih njenih elemenata.

*Rešenje.* Konkatenacija dve liste se vrši pomoću operatora +. Dodavanje elementa listi se vrši pomoću funkcije append. Izmena ili brisanje elemenata se može izvršiti pristupanjem odgovarajućim indeksima elemenata koji treba da budu izmenjeni ili izbrisani.

l i s t a = [ " jedan " , " dva" , " t r i " , " c e t i r i " ]

**print** ( l i s t a )

*# dodavanje novih elemenata*

l i s t a . append ( " pet " )

l i s t a += [ " s e s t " , "sedam" ]

**print** ( l i s t a )

*# menjanje v rednosti i l i b r i s an j e jednog elementa*

l i s t a [ 1 ] = " s to "

l i s t a [ 1 ] = " h i l ja d u "

−

**print** ( l i s t a ) **del** l i s t a [ 0 ] **print** ( l i s t a )

*# menjanje v rednosti i l i b r i s an j e v i s e elemenata*

l i s t a [ 2 : 5 ] = [ " m i l i on " ]

**print** ( l i s t a )

l i s t a [ 2 : 4 ] = [ ]

**print** ( l i s t a )

l i s t a [ 0 : ] = [ ]

**print** ( l i s t a )

1. Definisati torku, a zatim je ispisati na standardni izlaz, uključujući njenu dužinu.

*Rešenje.* Za razliku od liste, torka se definiše preko (). Indeksiranje i određivanje njene dužine se vrši slično kao kod liste. Elementi torke se ne mogu brisati niti menjati, a ne mogu se ni nakon definisanja dodavati novi.

torka = ( 4 . 5 , 4 , "sedam" , 2 )

**print** ( torka )

**print** ( " Duzina. j e : .%d . " % **len** ( torka ))

**print** ( torka [ 0 ] )

**print** ( torka [ − 1 ] )

1. Definisati rečnik, a zatim i vršiti izmenu, brisanje i dodavanje novih elemenata. Ispisati sve vrednosti ključeva, kao i vrednosti rečnika.

*Rešenje.* Rečnici se u Pythonu definišu u obliku recnik = {kljuc1:vrednost1, kljuc2:vrednost2, kljuc2:vrednost2}. Ključevi se mogu ispisati funkcijom keys, a vrednosti funkcijom values. Rečnici predstavljaju uopštenje listi, u smislu da njihovi indeksi (ključevi) ne moraju biti celi brojevi, već bilo koji tip podatka.

imenik = {" Petar " : 2 1 3 , " Jovan" : 7 9 8 , "Ana" : 427 }

**print** ( imenik . keys ( ) ) **print** ( imenik . v a l u e s ( ) ) **print** ( imenik )

imenik [ " Petar " ] = 214 **print** ( imenik [ " Petar " ] ) imenik [ " M i l i c a " ] = 908 **del** imenik [ " Jovan" ] **print** ( imenik )

1. Proveriti da li se neki element nalazi u listi.

*Rešenje.* Slično kao provere da li se podstring nalazi u nekom stringu, preko operatora in se proverava da li se neki element nalazi u datom nizu.

l i s t a = [ " ponedeljak " , " utorak " , " sreda " ]

**print** ( " c e tv r ta k " **in** l i s t a )

pripada = " sreda " **in** l i s t a

**print** ( pripada )

imenik = {" Petar " : 2 1 3 , " Jovan" : 7 9 8 , "Ana" : 427 }

**print** ( "Ana" **in** imenik . keys ( ) )

# NAREDBE GRANANJA

1. Napisati program koji proverava da li je dužina stringa veća od date vrednosti.

*Rešenje.* U Pythonu naredba if ima sledeći oblik:

**i f** uslov :

naredbe

Naredbe će se izvršavati, ukoliko je uslov ispunjen. Sve naredbe treba uvući i navesti jednu ispod druge. Kod Pythona treba strogo voditi računa o uvlačenju naredbi. Ovaj princip se koristi i kod naredbi grananja i kod ciklusa, kao i kod definisanja funkcija ili klasa. U C-u se, na primer, ne mora strogo voditi računa o uvlačenju, jer se naredbe odvajaju zagradama {}.

vrednost = 12

s t r i n g 1 = " Zdravo. s v e te " s t r i n g 2 = " Zdravo. s v e te ! "

**i f len** ( s t r i n g 1 ) > vrednost :

**print** ( " Prvi . s t r i n g . j e . duzi .od.%d" % vrednost )

# else :

**print** ( " Prvi . s t r i n g . n i j e . duzi .od.%d" % vrednost )

drugi Duzi = **len** ( s t r i n g 2 ) > vrednost

**i f** drugi Duzi :

**print** ( " Drugi . s t r i n g . j e . duzi .od.%d" % vrednost )

# else :

**print** ( " Drugi . s t r i n g . n i j e . duzi .od.%d" % vrednost )

1. Napisati program koji određuje da li je dati broj pozitivan, negativan ili jednak nuli.

*Rešenje.* Naredba grananja if se može napisati i u kombinaciji sa else na sledeći način:

**i f** uslov :

naredbe 1

# else :

naredbe 2

Ukoliko je uslov ispunjen, izvršiće se naredbe iz prve grupe, a ako nije, izvršiće se naredbe iz druge grupe. Može se razmatrati i više uslova istovremeno na sledeći način:

**i f** uslov 1 :

naredbe 1

**e l i f** uslov 2 :

naredbe 2

**e l i f** uslov 3 :

naredbe 3

# else :

naredbe 4

Može postojati proizvoljan broj elif naredbi. U ovom primeru, naredbe iz prve grupe će se izvršiti ukoliko je ispunjen prvi uslov. Inače, ukoliko je ispunjen drugi uslov, izvršiće se naredbe iz druge grupe. Inače, ukoliko je ispunjen treći uslov, izvršiće se naredbe iz treće grupe. Inače, izvršiće se naredbe iz četvrte grupe.

b roj 1 = 3

−

b roj 2 = 19

**i f** b roj 1 >= 0 :

**print** ( " Prvi . b r o j . j e . nenegativan " )

# else :

**print** ( " Prvi . b r o j . j e . negativan " )

**i f** b roj 2 == 0 :

**print** ( " Drugi . b r o j . j e . nula " )

**e l i f** b roj 2 > 0 :

**print** ( " Drugi . b r o j . j e . p o z i t i v a n " )

# else :

**print** ( " Drugi . b r o j . j e . negativan " )

1. Napisati program kojim se u naredbi grananja proveravaju uslovi koji u sebi sadrže logičke funkcije and, or ili not.

*Rešenje.* Logičke funkcije and, or i not su definisane u Pythonu i služe za konjukciju, disjunkciju, odnosno negaciju logičkih izraza. Vrednosti logičkih izraza koje ih koriste su definisane na sledeći način:

* + izraz1 and izraz2: vrednost izraza je True ako su vrednosti oba izraza True, a inače je False,
	+ izraz1 or izraz2: vrednost izraza je True ako je vrednost bar jednog izraza True, a inače je False,
	+ not izraz: vrednost izraza je suprotna vrednosti izraza koji je zadat kao argiment.

Logička funkcija not ima najveći prioritet, zatim funkcija and, a najmanji prioritet ima funkcija or. ime = " Petar "

godine = 20

**i f** ime == " Petar " **and** godine == 2 1 :

**print** ( " Petar .ima. 21 . godinu " )

# else :

**print** ( " Petar .nema. 21 . godinu " )

**i f** ime == " Jovan" **or** godine == 2 0 :

**print** ( "Ime. j e . Jovan. i l i . su . godine . 20 " )

# else :

**print** ( "Ime. n i j e . Jovan , .a. godine . nisu . 20 " )

*# redosled operatora j e not , and , or # r e z u l t a t uslova j e False*

**i f** ime == " Jovan" **or** ime == " Petar " **and not** godine == 2 0 :

**print** ( " P o s l e d n j i . uslov . j e . i s p u n je n " )

# else :

**print** ( " P o s l e d n j i . uslov . n i j e . i s p u n je n " )

# CIKLUSI

1. Koristeći for ciklus, ispisati elemente niza, odnosno karaktere stringa.

*Rešenje.* U Pythonu je for ciklus oblika:

**for** b r o ja c **in** i n t e r v a l : naredbe

Interval može biti naveden korišćenjem funkcije range, pri čemu range(i) označava da su u intervalu svi brojevi od 0 do *i* 1, a range(i, j) da su u intervalu svi brojevi između *i* i *j* 1. Elemetima niza i stringa se može pristupati u for ciklusu preko indeksa (tada se koristi funkcija range) ili direktno u obliku for element in niz, odnosno for element in string.

− −

*# i s p i s u j e b rojeve od 0 do 4*

**for** i **in range** ( 5 ) :

**print** ( i )

*# i s p i s u j e b rojeve od 3 do 10*

**for** i **in range** ( 3 , 1 1 ) :

**print** ( i )

n i z = [ 4 . 5 , 2 . 3 , 1 1 . 5 , 1 6 . 8 ]

*# elementi niza preko indeksa*

**for** i **in range** ( **len** ( n i z ) ) :

**print** ( n i z [ i ] )

*# elementi niza bez indeksa*

**for** element **in** n i z :

**print** ( element )

r e c = " zdravo "

*# elementi s t r i n g preko indeksa*

**for** i **in range** ( **len** ( r e c ) ) :

**print** ( r e c [ i ] )

*# elementi s t r i n g a bez indeksa*

**for** element **in** r e c :

**print** ( element )

1. Preko while ciklusa, ispisati kvadrate svih jednocifrenih prirodnih brojeva.

*Rešenje.* U Pythonu je while ciklus oblika:

**while** uslov :

naredbe

Naredbe će se izvršavati sve dok je ispunjen uslov koji je naveden. b r o j = 1

**while** b r o j < 1 0 : **print** ( b r o j 2 ) b r o j += 1

∗∗

1. Unutar for ciklusa koji prolazi kroz sve brojeve od 1 do 100, ispisati sve parne brojeve. Ukoliko je broj ispisanih parnih brojeva veći od 40, zaustaviti izvršavanje.

*Rešenje.* Izvršavanje for ciklusa može biti prekinuto korišćenjem naredbe break. Naredbom continue se direktno prelazi na sledeću iteraciju.

broj Parnih = 0

**for** i **in range** ( 1 , 1 0 1 ) :

**i f** i % 2 == 1 :

# continue

**i f** i % 2 == 0 : **print** ( i ) broj Parnih += 1

**i f** broj Parnih > 4 0 :

# break

1. Koristeći while ciklus, ispisati niz Fibonačijevih brojeva. Ukoliko je trenutni ispisani Fibonačijev broj veći od unapred zadatog broja, prekinuti izvršavanje ciklusa.

*Rešenje.* Naredbe break i continue se mogu koristiti i u kombinaciji sa while ciklusom. U ovom primeru se u svakoj iteraciji definiše novi Fibonačijev broj kao zbir prethodna dva, a zatim se redom, na osnovu njegove vrednosti, ažuriraju vrednosti prethodnog i trenutnog broja.

g r a n i c a = 10000 prethodni = 1 **print** ( prethodni ) tr e n u tn i = 1 **while** True :

**print** ( tr e n u tn i )

**i f** tr e n u tn i > g r a n i c a :

# break

novi = tr e n u tn i + prethodni prethodni = tr e n u tn i

tr e n u tn i = novi

# FUNKCIJE

1. Definisati funkciju koja na standardni izlaz ispisuje *Zdravo svete!*, a zatim je pozvati.

*Rešenje.* Funkcije se definišu korišćenjem ključne reči def, a zatim navođenjem imena funkcije i zagrada (), unutar kojih se eventualno mogu navesti argumenti. Naredbe unutar funkcije se zatim moraju uvući i navesti jedna ispod druge. U ovom primeru je navedena funkcija koja nema povratnu vrednost (odgovara tipu void u C-u) i koja nema argumenata. Funkcija se zatim može pozvati u glavnom delu programa.

**def** i s p i s ( ) :

**print** ( " Zdravo. s v e te ! " )

*# pozivanje f u n k c i j e i s p i s*

i s p i s ( )

1. Definisati funkciju koja sabira dva broja i pozvati je, pri čemu zbir treba ispisati na standardni izlaz.

*Rešenje.* Funkcija zbir koja je ovde implementirana ima dva argumenta i povratnu vrednost. Povratna vrednost funkcije se dodeljuje preko ključne reči return (na sličan način kao i u C-u).

**def** z b i r ( x , y ) : z = x + y **return** z

a = 10

b = 5

c = z b i r ( a , b )

**print** ( " Zbir . j e ." + **str** ( c ) )

1. Definisati funkciju čiji su argumenti string i broj. Ukoliko pri pozivu funkcije broj nije naveden, po- drazumevana vrednost je 3. Funkcija treba da vraća string nadovezan onoliko puta kolika je vrednost broja.

*Rešenje.* Ukoliko neki argument ima podrazumevanu vrednost, ona se može navesti pri navođenju argu- menata funkcije. Ako se takav argument eksplicitno ne navede, pri izračunavanju će se koristiti njegova podrazumevana vrednost. Argumenti koji imaju podrazumevanu vrednost, moraju biti navedeni kao posled- nji argumenti funkcije.

**def** nadovezi ( rec , b r o j = 3 ) :

**return** r e c ∗ b r o j

r e c = " zdravo " b r o j = 4

**print** ( nadovezi ( rec , b r o j ) )

**print** ( nadovezi ( r e c ) )