IOM International Organization for Migration

BRITISH EMBASSY

Uvod u Python

Implementing Partner: coders ACADEMY

IOM International Organization for Migration

BRITISH EMBASSY

Uvod u Python

Upoznajmo programski jezik Python Nizozemac Guido van Rossum je 1989. uveo programski jezik Python kao prikladan za podučavanje programiranja. Za njegovu popularnost postoji nekoliko razloga:

- Besplatan je i može se preuzeti na web adresi http://www.python.org/download/
- i instalirati na operacijske sustave Windows, Linux i Mac OS.
- Jednostavna pravila pisanja (sintaksa) tako da se lako može naučiti

Implementing Partner: coders ACADEMY

- Prikladan je za rješavanje svakodnevnih problema jer omogućuje brzo pripremanje i i ispitivanje programa
- Osim za podučavanje, koristi se u profesionalne svrhe za izradu programa koji se koriste u svakodnevnom životu

Osnovni tipovi podatka

Osnovni tipovi podataka u Pythonu su:
Integer (Cijeli brojevi)
Float/Double (Decimalni brojevi)
Znakovni niz ili string
Boolean (True ili False)

Operatori

Operatori (eng. Operator) je simbol koji daje računaru instrukciju da izvrši neku operaciju. Operatori se dijele na operator pridruživanja, matematičke operatore, relacione operatore i logičke operatore.

Aritmetički operatori: Manipuliramo sa cijelim brojevima (integer) ili decimalnim brojevima (float/double) i kao rezultat imaju isti tip podatka						
+	-	*	/	%	**	//

Relacioni operatori: Manipuliarju sa svim tipovima podatka i kao rezultat vracaju samo booleanske vrednosti (True ili False)					
>	>=	<	<=	==	!=

Logički operatori: Manipuliarju sa podacima tipa boolean i kao rezultat vracaju booleansku vrednost

AND	OR	NOT

Operator pridruživanja '=': Upotrebljavamo kada hocemo da se varijabli dodjeljuje određena vrijednost. On pridružuje vrijednost izraza sa lijeve strane varijabli sa desne strane. Upotrebljava se u sledećem obliku: `variabla = "jCoders"`

Variable (Promenljive)

Varijable (eng. Variable) su su memorijske lokacije koje sadrže podatke. Sa varijablama ste se sreli još u prvom razredu osnovne škole. Samo, tada ste ih zvali nepoznate. Npr. Koliko je x, ako je x = 2 + 3? Isto važi i za varijable u Python-u, uz neke male modifikacije, jer je ovo ipak programiranje, a ne matematika.

Imena varijabli

Svaka varijabla ima svoje ime. Nisu sva imena dozvoljena i moguća. Za to postoje pravila:

- Ime varijable mora biti sastavljeno od slova, cifri i donjih crta (_) (eng. Underscore).
- Prvi znak mora biti slovo. Dozvoljeno je i da donja crta bude prva, ali se ne preporucuje.
- Rezervisane riječi C sintakse takođe ne mogu biti varijable.
- Dozvoljena je upotreba i kombinovanje velikih i malih slova.
- Dužina imena neka ima do 31 znak. Ne podržava svaki kompajler duža imena.

Sledeća tabela pokazuje primjere ispravnih i neispravnih imena varijabli.

Ime Variable	Ispravnost
Procenat	Ispravna
procenat	Ispravna
pROCenat	Ispravna

Ukupna_vrijednost	Ispravna
MarkovNovac	Ispravna
a_b_c_d_e__f___g	Ispravna
cijena	Ispravna, ali se ne preporučuje
ukupan_broj_predjenih_kilometara	Ispravna, ali se ne preporučuje
M4rk0	Ispravna
root#komanda	Neispravna, jer sadrži znak #
Sličica	Neispravna, jer sadrži znak sa kvačicom
5ak	Neispravna, jer počinje sa brojem
int	Neispravna, jer je to rezervisana riječ sintakse
char	Neispravna, jer je to rezervisana riječ sintakse

Varijable "Procenat" i "procenat" nisu iste varijable. Python je jezik koji je osjetljiv na velika i mala slova (eng. Case-Sensitive).
Preporučuje se upotreba opisnih varijabli, kao što su "ZbirBrojeva" ili "Ukupna_vrijednost", jer čine kod razumljivijim nego da se koriste "a", "b", "c", "d"...

Definisanje varijabli

U Python-u, varijable moraju se deklarisati (definisati). Deklaracija varijable informiše kompajler o imenu varijablekoje ona sadrži U toku izvršavanja aplikacije, varijabli se obično dodjeljuje određena vrijednost. To se čini sa znakom pridruživanja (=).

Deklarisanje i dodeljenje određenih vrijednosti ima sledeći oblik:

```
variabla = "jCoders"
```

Komentari

Dok programiramo mnogo je vazno da razjasnimo funkciju naseg koda. Početnici obično smatraju komentare nepotrebnima. Varaju se. Oni su od neprocjenjivog značaja za razumjevanje koda. Kod je jasan dok se piše, dan-dva poslije pisanja ili kad je jednostavan. Međutim, teško se sjetiti šta ste mislili dok ste pisali kod prije šest mjeseci. Ne kažem da treba komentarisati svaki red koda. To je suludo. Već treba komentarisati ono što je nejasno, novo, komplikovano, ono što kod treba da uradi i sl.

Često komentari mogu pomoći pri otklanjanju grešaka. Ubacimo dio koda, za koji smatramo da je problematičan, i vidimo da li aplikacija radi bez tog dijela, kakve greške tada javlja i sl.

Da bi to postigli, samo trebamo pred komentara staviti # (bez navodnika). Primjer:

```
#Jedan red komentar
```

Kad hocemo da napisemo komentar koji je duzi nego jedan red onda trebamo svoj komentar staviti između tri navodnicima **"""". Primjer:**

```
"""
komentar
koji
ima
vise
redova
"""
```

Print funkcija

Print je funkcija za ispis podataka na ekranu monitora. O tome šta će da ispiše, ovisi o tome šta će dobiti za argumente.

U primjeru iz jednog od prošlih poglavlja vidi se da je za argumente dobila tekst. Da bi funkcija mogla da ispiše tekst, on mora biti između znakova navodnika ("). Primjeri:

```
print("Hello World")
```

Kako formatirati tekst u Python

Sada se možda pitate, kako će da ispiše navodnike, pošto navodnicima označavamo početak i kraj teksta. Ima i za to riješenje. Ako se u tekstu nalaze navodnici potrebno je prije istog staviti "naopako dijeljenje" (\) (eng. Back Slash).

Znak	Ispisuje
\n	Novi red (eng. New Line)
\t	Tab (eng. Tab) Horizontalni razmak
\’	Apostrof
\”	Navodnici (eng. Double quote)
\\	Backslash

Sa print funkcijom se mogu ispisivati i vrijednosti varijabli. Za taj efekat potrebno je da na mjesta gdje će biti varijable stavimo %d, %s ili %f, i da poslije završnih navodnika stavimo “%” simbol koje će se varijable ispisati sa zarezima prije. Primjer:

```
print("Ja imam %d euro" %(novac))
```

Rezultat:

```
>
Ja imam 100 euro
>
```

Input

Input je funkcija koja pridružuje varijabli podatke unjete tastaturom. Sintaksa joj je slična print, koja ispisuje varijablu. Primjer upotreblavanja:

```
godine = input("Koliko godina imas ?")
```

Kada se izvrsi ovaj kod, mi cemo imati mogucnost da se odgovorimo u konsoli

```
Koliko godina imas ? 35
```

Kad odgovorimo 35, onda se ova vrednost odredjuje variablom sa imenom godine

Konvertiranje tipova podatka

Funkcije str(), int() i float()

Funkcija za konvertiranje	Rezultat
x='33' y=int(x)	Tekst '33' se konvertuje u cijeli broj 33
x=100 y=str(x)	Cijeli broj 100 se konvertuje u tekst '100'
x='33.77' y=float(x)	Tekst '33.77' se konvertuje u decimalni broj 33.77

Prethodni primeri pozivaju funkcije str(), int() i float() i prosleđuju im vrednosti drugih vrsta podataka za dobijanje niza, celog broja ili oblika pokretne tačke tih vrednosti

Pokušajte da konvertujete neke vrednosti u interaktivnoj konzoli pomoću ovih funkcija i pogledajte šta će se desiti.

Primer	Rezultat
str(0)	'0'
str(-3.14)	'-3.14'
int('42')	42
int('-99')	-99
int(1.25)	1
int(1.99)	1
float('3.14')	3.14
float(10)	10.0

Funkcija int() je takođe korisna ako imate broj kao vrednost niza koji želite da upotrebite u nekim drugim izračunavanjima. Na primer, funkcija input() uvek vraća tekst, čak i ako korisnik unese broj. Unesite broj = input() u konzolu i unesite 101 kada program zatraži tekst. Posle toga, ako probate da zbirite variablu broj sa nekom drugim brojem, pojavice se greska, jer ne mozemo da zbirimo tekst sa brojem

Primer:

```
broj = input("Unesi neki broj")
print(broj+2)
```

Rezultat:

```
 Unesi neki broj 100
Traceback (most recent call
last):
  File "python", line 4, in
<module>
TypeError: must be str, not int
```

Pravilan primer:

```
broj = int(input("Unesi neki broj"))
print(broj+2)
```

Rezultat

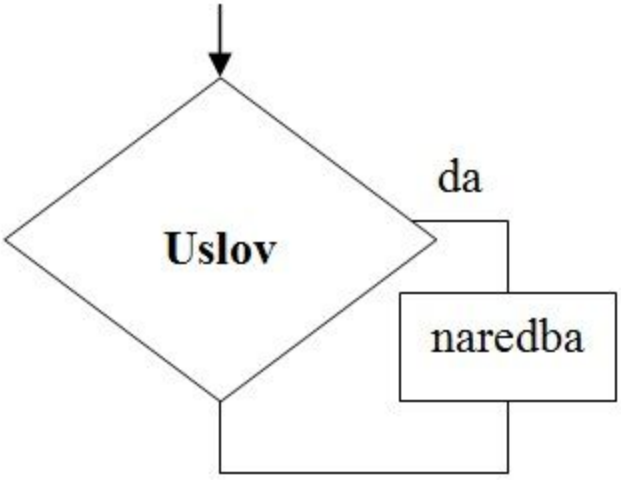
```
Unesi neki broj 5
7
```

U programu ste upotrebili funkcije int()

Kontrola toka aplikacije (if/elif/else)

if naredba

If naredba je naredba kontrole toka aplikacije. U dosadašnjim primjerima sve naredbe aplikacije su morale biti izvršene. If naredba se sada javlja kao nekakav semafor, koji reguliše koje će se naredbe izvršiti.



If naredba se upotrebljava u sledećem obliku:

```
if(USLOV):
  NAREDBA
```

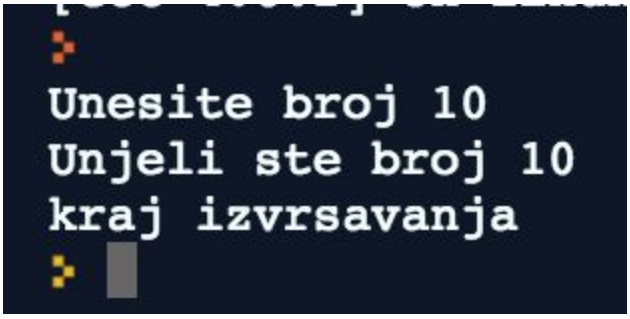
Vazno je da znamo da naredba treba da ima razmak, i na kraju svakog if-a treba da imamo dve tocke “:”.

što znači: "Ako je uslov zadovoljen (tačan), izrvšiće se naredba (tačno).". Primjer:

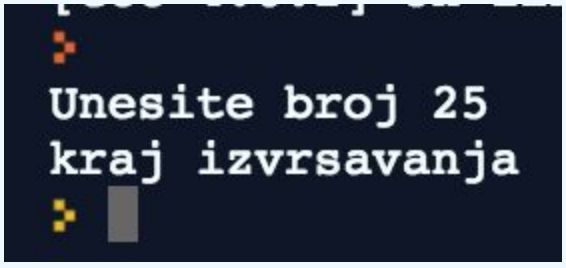
```
broj = int(input("Unesite broj"))
if(broj==10):
  print("Unjeli ste broj 10")

print("kraj izvrsavanja")
```

Ako odgovorimo 10, onda je uslov popunjen i rezultat ce da bude:



Ako odgovrimo nesto drugo od 10, na primjer 20, onda uslov nije popunjen, i rezultat ce da bude:

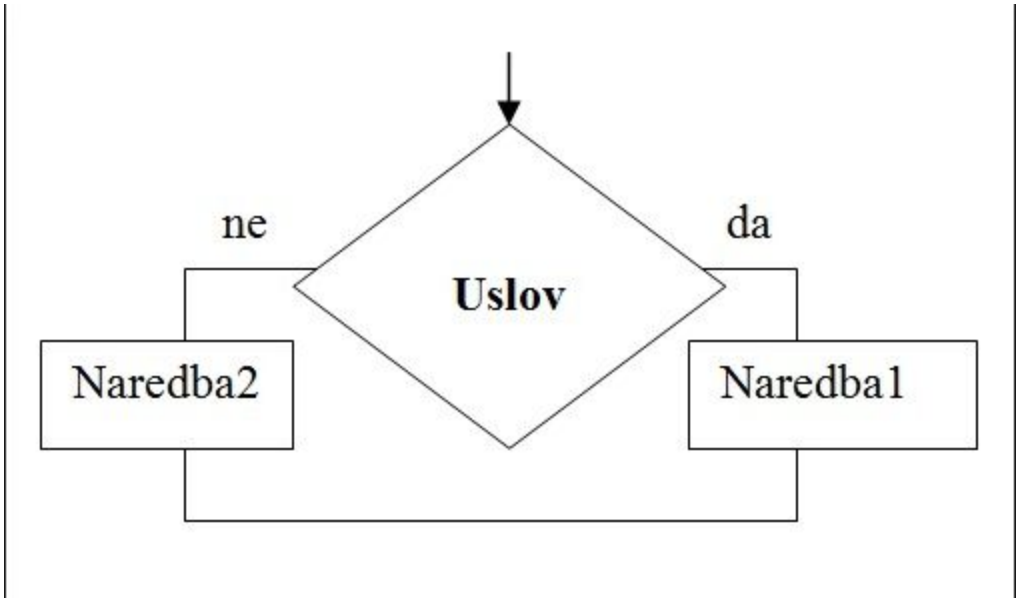


Kada smo kompjalirali i pokrenuli aplikaciju, aplikacija je zatražila da unesemo broj. Unjeli smo broj 10, koji je pridružen broj-u. Postavlja se pitanje, da li je broj jednako 10. Odgovor je tačno. Uslov je zadovoljen i izvršiće se naredba ispisa. U drugom slučaju, odgovor je bio netačan pa se nije izvršila naredba. Sledeća naredba ispisa ("Kraj izvrsavanja.") se izvršava u oba slučaja jer ta naredba nije obuhvaćena if naredbom. Ukoliko želimo da se izvrši više naredbi pri zadovoljenom uslovu, onda trebamo napraviti blok naredbi.

```
broj = int(input("Unesite broj"))
if(broj==10):
  print("Unjeli ste broj 10")
  print("Vas broj je 10")
```

else naredba

Da bi izbjegli if naredbu za slučaj kada uslov nije zadovoljen, koristimo else naredbu. Ona se, poput if, može koristiti van bloka, ali se to ne preporučuje.



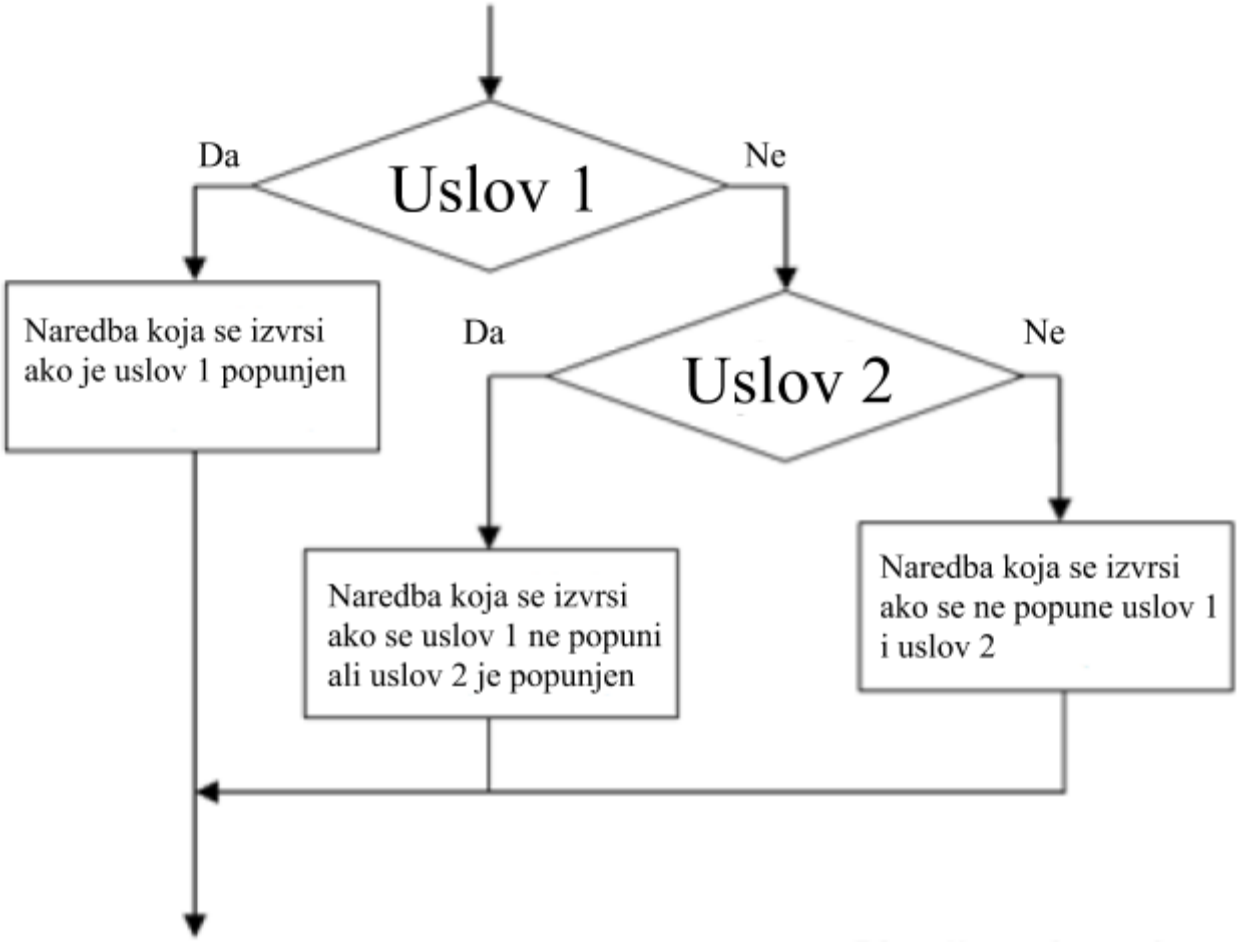
Primer:

```
broj = int(input("Unesite broj"))
if(broj>=0):
  print("Broj je pozitivan ili nula")
else:
  print("Broj je negativan")
```

Kada se uslov popuni

Kada se uslov ne popuni:

Elif naredba



Kada napisemo aplikacije koje imaju vise uslova onda treba da upotrebljavamo elif. Elif nam omogucava da predstavimo jos uslova u nasem programu. Primjer

Elif izraz nam omogucava da predstavimo vise uslova. Slicno kao else, elif izraz je opsinalan. Medjutim, mi mezomo da imami arbitarni broj elif-a u jednom if izrazu. Primjer:

U ovom programu, mi hocemo da proverime jel nas broj pozitivan, negativan ili nula, i zavisi od broja da nam ispisuje tekst koji prikazuje kakav broj je.

```
broj = 3.4

if broj > 0:
    print("Pozitivan broj")
elif broj == 0:
    print("Nula")
else:
    print("Negativan broj")
```

Petlje

OSNOVNI POJMOVI

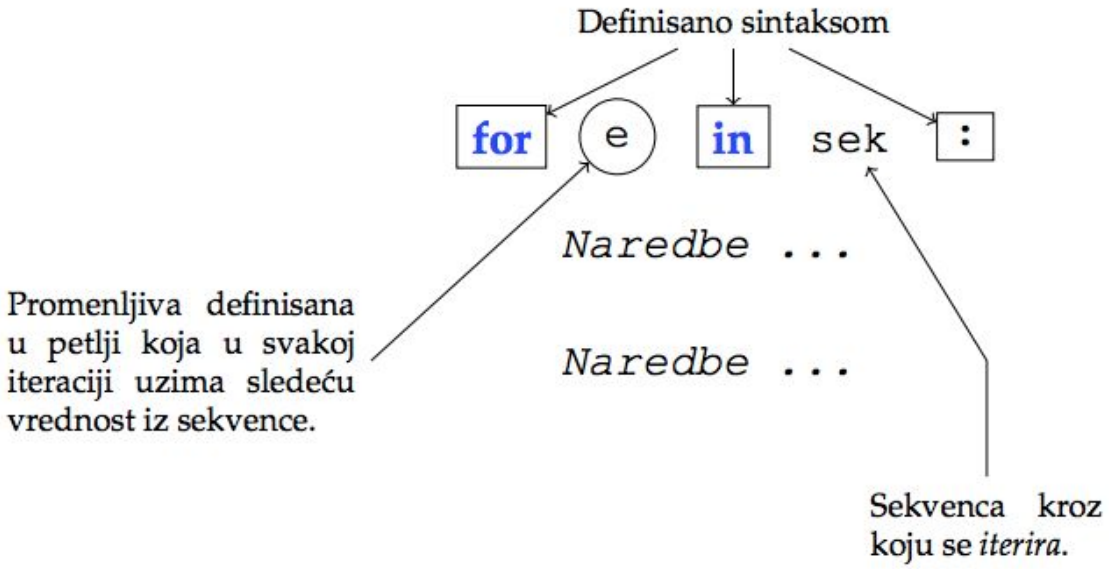
Petlje su programske konstrukcije koje omogu´cavaju da se deo koda izvršava ve´ci broj puta. Razlikujemo dve vrste petlji:

for-petlja: Nabrojiva ili for petlja jeste petlja u kojoj se kod izvršava po jednom za svaki element neke sekvence (recimo liste ili N-torke).

while-petlja Nenabrojiva ili while petlja jeste petlja u kojoj se kod izvršava sve dok je neki logi˘cki uslov ispunjen.

- Iteracija predstavlja jedan ciklus petlje (odnosno skup operacija koje se izvrše u jednom ciklusu).
- **for** petlja se izvršava unapred poznat broj iteracija.
- **while** petlja se izvršava broj puta koji je unapred (u principu) nepoznat - iteracije slede jedna za drugom sve dok je uslov ispunjen.

for PETLJA



Sintaksa:

```
for i in range (pocetna_vrednost, krajna_vrednost):
  #Naredba
```

JEDNOSTAVNI PRIMERI UPOTREBE PETLJI

Poput if/else naredbe, for naredba je naredba kontrole toka aplikacije. To je naredba kojoj se postiže višestruko izvršavanje neke naredbe ili bloka. Često se naziva for petlja. U upotrebi ima najčešće sledeći oblik:

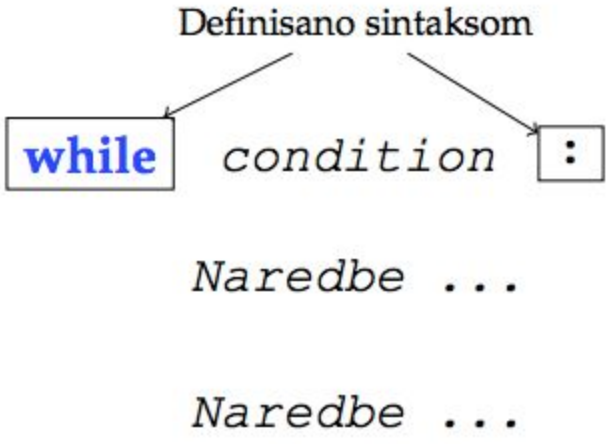
Primer:

```
for i in range (0, 10):
  print("%d"%(i))
```

Rezultat:

Variabli ***i*** se pridruži početna vrijednost, 0. i (0) zadovoljava uslov zato sto je u intervalu (0,10), izvršava se naredba u bloku i ***i*** se uvećava za jedan (1). i (1) ponovo zadovoljava uslov, izvršava se naredba u bloku i ***i*** se uvećava za jedan (2). Ciklus se ponavlja sa stalnim uvećanjem, sve dok ***i*** ne postane 9. x (9) zadovoljava uslov, izvršava se naredba u bloku i ***i*** se uvećava za jedan (10). i (10) ne zadovoljava uslov, izlazi se iz ciklusa (naredba se ne izvršava), jer *funkcija range ne uzima na primer zadnji broj* (znaći na ovom slućaju ne uzima 10, nego ide do 9)

while PETLJA



While je naredba kontrole toka i petlja poput for petlje. Suštinski između ove dvije petlje nema razlike, sem što se savjetuje for za upotrebu. Upotreba:

```
while (uslov):
  #Naredba broj 1
  #Naredba broj 2
```

Primer:

```
broj = 0
while (broj<=10):
  print(broj)
  broj=broj+1
```

Rezultat:

While petlja može sve isto što i for petlja.

Kao kod if/else naredbe, i kod while je se uslov može postaviti ovako:

Kod gniježdenja petlji, bilo to da je for-for, while-while, for-while ili while-for, često se griješi. Dešava se da se koristi ista varijabla i u spoljnjoj i unutrašnjoj petlji. Fatalna greška!

Funkcije

Kada se neki deo koda u istom ili sličnom obliku ponavlja iznova i iznova, pogodno ga je definisati u obliku funkcije.

Modularizacija koda (podela na manje segmente jasno definisane namene) poželjna je (i neophodna) iz više razloga:

- Funkcija se definiše na jednom mesto, a poziva proizvoljan broj puta.
- Funkcija se može testirati nezavisno od koda u kome če se pozivati.
- Sve izmene treba vršiti na jednom i samo jednom mestu.

DEFINISANJE FUNKCIJA

Definisanje funkcije:

```
def ime_funkcije (arg1, arg2, ..., argn) :
    Naredbe…
```

Primer Definisati funkciju koja za dati uneti broj (nazovimo ga x) vraca broj x + 6, ako je x paran broj, a x/2 ako je x neparan broj. Definicija funkcije def primer (x): if x % 2 == 0: # Ispitujemo da li je x paran broj. return x+6 # Ako jeste, vracamo x+6. else: return x/2 # Ako nije, vracamo x/2. Poziv funkcije y = primer(6) print(y)

DEFINISANJE FUNKCIJA ALTERNATIVNO, CITKIJE REŠENJE ˇ ◭ Jedan od razloga zbog koga je zgodno i potrebno definisati funkcije jeste radi pove´canja ˇcitljivosti koda. Python je jezik koji je ˇcuvan po ˇcitljivosti svog programskog koda (sa tom je namenom i projektovan). ◭ Konstrukt if x % 2 == 0 se nakon dovoljno dugo vremena može uˇciniti neˇcitkim (naroˇcito ukoliko ne proveravate parnost brojeva na ovaj naˇcin tako ˇcesto). ◭ Umesto toga, korisnije je definisani posebnu funkciju koja samo proverava parnost, a potom tu funkciju koristiti u uslovu. Definicija funkcije – Drugi naˇcin def is_even(x): return x % 2 == 0 # Ako je x parno, ostatak je nula. def

primer (x): if is_even(x): # Ispitujemo da li je x paran broj. return x+6 # Ako jeste, vracamo x+6. else: return x/2 # Ako nije, vracamo x/2.

USLOVNA DODELA VREDNOSTI ILI: JOŠ JEDAN NACIN DA SE URADI PRETHODNI ZADATAK ˇ ... ▲ Uslovna dodela vrednosti omogu´cava da se na ˇcitkiji, jednostavniji naˇcin promenljivoj dodeli vrednost u zavisnosti od ispunjenosti nekog uslova. Sintaksa promenljiva = vrednost 1 if uslov else vrednost 2 Definicija funkcije – Tre´ci naˇcin def is_even(x): return x % 2 == 0 # Ako je x parno, ostatak je nula. def primer (x): return x+6 if is_even(x) else x/2

Kolekcije

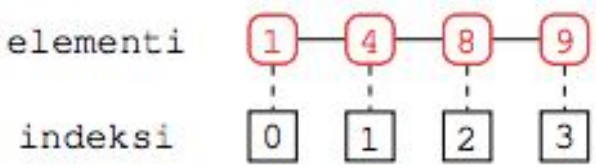
Kao što im i samo ime kaže, kolekcije su objekti koji služe za grupisanje i organizaciju drugih objekata.

Kolekcije se medu sobom razlikuju prema strukturi (odnosno prema uredenju objekata koje sadrže), te prema performansama.

- Pozivom len(k) dobija se broj elemenata kolekcije k.
- Pripadnost elementa e kolekciji k proverava se konstrukcijama e in k, odnosno e not in k. list Lista predstavlja niz objekata koji slede jedan iza drugog.
- Svaki objekat se na jedinstven način može identifikovati pomoću indeksa.
- Indeks predstavlja rednu broj objekta u listi, pri čemu se početni objekat indeksira nulom, a potonji objekti jedan za drugim brojevima 1, 2, ... Poslednji objekat ima indeks L-1, gde je L dužina liste (broj objekata koji su u njoj sadržani).
- tuple N-torka je strukturno identična listi, razlika je u performansama i načinu upotrebe. dictionary Rečnik predstavlja skup parova ključ/vrednost, pri čemu je ključ identifikator koji na jedinstven način identifikuje vrednost. Rečnik je, u suštini, preslikavanje koje svakom ključu pridružuje vrednost na jedinstven način. set Neureden skup objekata koje se ne mogu ponavljati.

Liste

Veoma često (gotovo uvek) elementima liste je zgodno pristupati po indeksu. To je naročito slučaj ukoliko jednovremeno treba pristupati većem broju listi, ili ukoliko je potrebno prostupati prethodnom i sledečem elementu datog elementa. Tada se koristi ugradena funkcija range. range(N) vrača sekvencu čiji su elementi 0, 1, ..., N-1. Postoji i varijanta naredbe kojom se mogu zadati i početni indeks, kao i korak (slično kao kod sečenja liste).



Primeri

```
x = [1, 4, 8, 9] # Lista elemenata
y = [] # Prazna lista
z = [2.0, 'Zdravo'] # Lista koja sadrzi raznorodne elemente
q = ['Zdravo', [1.3, 2], []] # Lista koja sadrzi druge liste
```

- Lista predstavlja niz objekata (proizvoljnog tipa, objekti u istoj listi mogu biti raznorodni, a neki ili svi i sami mogu biti liste).
- Svakom elementu liste na jednoznačan način se pridružuje nenegativa

INDEKSIRANJE LISTI PRISTUPANJE ELEMENTIMA LISTE

- Indeksiranje jeste pristupanje elementima liste po rednom broju, počev od 0.
- Indeksiranje se vrši pomoću uglastih zagrada: ime_liste[indeks].

```
lista = ['a','b','c','d'] # Definisanje liste
print(lista[0]) # Na ekranu se ispisuje 'a'
lista[1] = '2' # Element 'b' se zamenjuje sa '2'
```

Ukoliko se zahteva element koji ne postoji u listi (recimo da lista ima ukupno 5 elemenata, sa zahteva se 10-ti), javlja se greška.

- Dužina liste, odnosno broj elemenata u listi, dobija se pozivom funkcije len.
- Lista se može indeksirati i negativnim brojevima. Tada je -1 poslednji, -2 predposlednji, itd. broj u listi.

Primer:

```
N = len(lista) # Promenljiva N sadrzi broj elemenata liste
lista[N] # Greska! Poslednji element liste indeksira se sa N-1
```

Sečenje listi(slicing)

Pristupanje podlistama

Sečenje jeste postupak formiranja podlisti.

- Sečenje se vrši pomoću sintakse slične indeksiranju. Indeksiranje se može vršiti u osnovnoj varijanti (specificiranjem početnog i krajnjeg indeksa) i u proširenoj varijanti (specificiranjem početnog i krajnjeg indeksa i koraka)

```
ime_liste[start_indeks:stop_indeks] > ime_liste[start_indeks:stop_indeks:korak]
```

- Pri indeksiranju, element čiji je indeks jednak stop_indeks-u se ne uzima u obzir. (Sem ukoliko start_indeks i stop_indeks nisu jednaki).
- Ukoliko se prvi indeks izostavi, smatra se seče od prvog elementa.
- Ukoliko se poslednji indeks izostavi, smatra se da se seče do zadnjeg elementa

```
#Primeri sečenja liste
lista = ['a','b','c','d'] # Definisanje liste
x = lista[0:1] #x <- ['a']
y = lista[1:3] #y <- ['b', 'c']
z = lista[1:] #z <- ['b', 'c', 'd']
```

Sečenje listi (slicing) - Složeniji primeri

```
#Primeri sa negativnim indeksima
lista = ['a','b','c','d'] # Definisanje liste
x= lista[1:-1] # x <- ['b', 'c']
y = lista[1:-2] # x <- ['b']
```

```
#Primeri sa korakom
lista = ['a','b','c','d'] # Definisanje liste
x = lista[1:4:2] # x<- ['b', 'd']
y = lista[::2] # x<- ['a', 'c'] (svaki drugi element)
z = lista[::-1] # x<- ['d', 'c', 'b', 'a'] (obrnuti poredak)
```

Osnovne Operacije nad listama neke funkcije clanice klase

append `lst.append(e)` dodaje element `e` na kraj liste `lst`.

extend `lst.extend(l)` dodaje sve elemente liste `l` na kraj liste `lst`.

insert `lst.insert(index, object)` umeće `objekat` pre indeksa `index`. Nakon umetanja `lst[index]` će biti jednako `object`.

index `lst.index(value)` vraća indeks na kome se vrednost `value` pojavljuje prvi put. Dodatnim argumentima mogu se specificirati početni i krajnji indeks pretrage. U slučaju da traženi element ne postoji, javlja se greška.

count `lst.count(value)` vraća broj ponavljanje vrednosti `value` u listi.

remove `lst.remove(value)` uklanja prvo pojavljivanje vrednosti `value` u listi. Javlja grešku ukoliko data vrednost ne postoji.

Bitna operacija nad listama koja nije implementirana kao funkcija ˇclanica jeste uklanjanje elementa po indeksu. Ova operacija se vrši pomo´cu ugra ¯dene funkcije del. Tako recimo, del(lista[i]) uklanja element sa indeksom i iz liste.

Zadaci

Ovdje se nalaze zadaci za vježbanje sa rješenjima.

Broj Aktiviteta	Opis	Link
1	Napisati aplikaciju koja ispisuje “Hello World” na ekranu	
1	Napisati aplikaciju u kojoj imamo definirane variable za svaki tip podatka u Python-u i svaku vrednost svake variable da ispisujemu na ekranu	
1	Napisati aplikaciju koja ispisuje u ekranu zbir, umanjenje, mnozenje, proizvod dva cijeli broja	
1	Napisati aplikaciju koja ispisuje vaše ime na ekranu.	
1	Napisati aplikaciju koja ispisuje vaše ime i prezime u dva reda. Napisati aplikaciju koja ispisuje vaše ime i prezime odvojene tabom.	
2	Napisati aplikaciju koja učitava ime korisnika (kroz input funkciju) i na kraju ispisuje na ekranu ime na ovaj oblik: .: jCoders Academy :.	
2	Napisati aplikaciju koja učitava godinu rodjenja i racuna koliko godine ima korisnik. Na kraju na ekranu treba de se ispisuje: Zdravo IME, vi imate X godine	
2	Napravite konverziju evra u dolara, gde se evri uzimamo kao input iz korisnika i na kraju ispisujemu u ekrani: Ti imas X dollara	
2	Napisati program koji izracunava povrsinu kruga I kao Input uzima radius kruga : Formula povrshine kruga : S=pi*radius 2	
	Napisati aplikaciju za određivanje parnosti broja.	
3	Napisati aplikaciju koja između dva unesenena broja	

	određuje veći.	
3	Programirati jedan kviz na ovaj nacin • Dobiti odgovor od korisnika koristeci funksiu Input • Kontrollisati da li je odgovor tacan I u tom slucaju upisati u ekranu “Tacan Odgovor” • U slucaju da odgovor nije tać an , pritisnuti poruku “Odgovor nije tać an , pokusaj ponovo” Stvoriti jednu variablu sa imenom: poeni Kad god je odgovor tacan povecati poen za jedan i na kraju programa odstampati koliko je poena dobiveno .	
3	Napisajte program koji u zavisnosti od poena stvoreni inputom, izracunava I prikazuje ocenu po sledecim pravilima . Ocena 1 - 0-29 Ocena 2 -30-49 Ocena 3 - 50- 69 Ocena4 - 70- 89 Ocena 5 - 90- 100	
4	Napisati aplikaciju koja ispisuje sve brojeve od 1 do n.	
4	Napisati aplikaciju koja ispisuje parne brojeve od 1 do n.	
4	Napisati aplikaciju koja ispisuje sumu brojeva od 1 do n.	
4	Napisati aplikaciju koja ispisuje sumu parnih brojeva od 1 do n.	
4	Napisati aplikaciju koja izracunava faktore unesenog broja.	
4	Napisajte program koji pokazuje koliko puta se ponovi jednu odredjeno slovo koje se uzima kao Input u okviru jedne reci	

	• Uzeti rec kao input i sacuvatije u variable “Rec” • Uzeti slovo kao Input I sacuvati u variabli “slovo” • Povecati jednu variablu koliko puta pronadjes slovo u reci • Odstampati koliko puta je nadjeno slovo unutra recenice	
5	Napisati aplikaciju koja ispisuje sve brojeve od 1 do n.	
5	Napisati aplikaciju koja ispisuje parne brojeve od 1 do n.	
5	Napisati aplikaciju koja ispisuje sumu brojeva od 1 do n.	
5	Napisati aplikaciju za ispis proizvoda brojeva od 1 do n.	
5	Napisati aplikaciju koja neprestano ispisuje vaše ime.	
5	HANGMAN Upotrebljavajuci listu formirati igru HANGMAN 1. Stvoriti listu **word** sa slovima od reci da bih Ih pogodili 2. Stvoriti jednu listu **guessed_word** sa onoliko elemenata koliko imamo na listi word 3. Uzeti kao Input jedno slovo 4. Proveriti dali postoji to slovo na listi **word** 5. Ukoliko postoji na listi **word** , dodati slovo na listi **guessed_word** na poziciji gde je nadjeno nalisti **word** 6. Nastaviti trazenje po listama dok na listi **guessed_word** ima jos (petlja while) Dodatni izazov : Dati korisniku 11 mogucnosti da pogodi	
6	Heads Or Tails Resiti naslucajan nacin Heads ili Tails	

	Hint : Rendom.choice(lista) Uzeti od korisnika Heads ili Tails U koliko je izabrao Heads, I na slucajan nacin je izabreo Heads , povecati score za 1 Usuprotnom smanjiti score za 1	
6	Stvoriti kviz 1. Stvoriti jednu listu sa pitanjima 2. Stvoriti jednu listu sa odgovorima 3. Stvoriti uslov koji kontrolira dali je odgovor tacan 4. Ukoliko je odgovor tacan , povecati variablu score za 1 5. Ukoliko nije smanjiti variablu score za 1 Dodatni izazov : Stvoriti Highscore	
7	Napisati aplikaciju koja koristi funkciju za izračunavanje zbira.	

7	Napisati aplikaciju koja koristi funkciju za izračunavanje faktorijela.	def faktorieli(numri):
		  rezultati = 1
		  while(numri>0):
		  rezultati=rezultati*numri
		  numri=numri-1
		  return rezultati
		
		print(faktorieli(8)+7)

7	Napisati program koji koristi funkciju **calculate(x,y,o)** koji u zavisnosti od parametra o ce vratiti zbir x+y ili proizvod x*y	
7	Razviti igru rock paper scissors. Za kontrollu pobednika upotrebiti funkciju **checkWinner(pl1,pl2)** koji dstampava rezultat igre . Upotrebiti listu na kojoj se cuvaju vrednosti (rock,paper , scissors), zatim jedno resenje da bode od strane kompjutera a drugo od strane igraca I zatim poslati funkcije za kontrollu..	

7	Napisati program na kome ce se stvoriti Jedna lista koja saderzi neke numericke vrednosti . Stvoriti tri funkcije koje prihvataju kao parameter jednu listu , prvi prnolazi minimalnu vrednost na listi , a drugi nalazi maksmalnu vrednost , dok treci nalazi srednju vrednost na listi	