

Beleške sa časa – 7.nedelja

Terminologija

- brojač = brojačka promenljiva = promenljiva koja nešto broji
- inicijalizacija = dodela inicijalne (početne) vrednosti (promenljive, objekta)
- inkrementacija, inkrementiranje = uvećanje vrednosti (promenljive, brojača)
- iteracija = korak (petlje)
- ulaz programa = svi unosi korisnika prilikom izvršavanja programa

O petljama

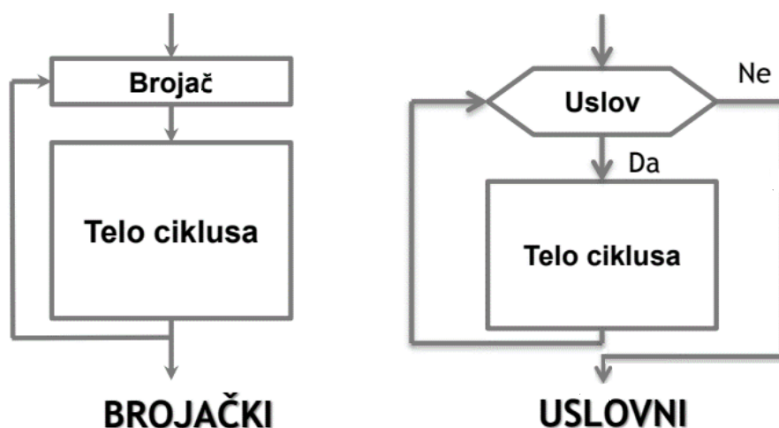
- petlje/ciklusi predstavljaju konstrukciju programskih jezika pomoću kojih se omogućava ponavljanje tj. izvršavanje više puta jedne ili više naredbi (blok naredbi)
- blok naredbi koji se izvršavati više puta nazivaju se *telo petlje/ciklusa*
- jedan korak petlje tj. jedan ciklus naziva se *iteracija petlje/ciklusa*
- jedan korak petlje podrazumeva izvršavanje svih naredbi unutar bloka naredbi koji čini telo petlje
- postoje dve vrste petlji/ciklusa:
 - brojačke petlje
 - petlje sa uslovom
- u Python-u postoje dve naredbe koje se koriste za realizaciju petlji u programima – **for** i **while** naredbe (nazivaju se još *naredbama ponavljanja*)
- po prirodi svoje konstrukcije, za brojačke petlje se uobičajeno koristi **for** naredba, dok se za petlje sa uslovom koristi **while** naredba, iako se pomoću i jedne i druge naredbe mogu realizovati obe vrste petlji

Brojačke petlje

- ovakve petlje podrazumevaju postojanje jedne kontrolne promenljive, tzv. *brojača (brojačke promenljive)*, koja obavlja odbrojanje koraka petlje od neke početne do neke krajnje vrednosti
- kod brojačke petlje broj koraka petlje je unapred poznat i određen je opsegom ili skupom vrednosti koje uzima brojačka promenljiva
- koliko se različitih vrednosti dodeli brojačkoj promenljivoj, u obliku nekog opsega vrednosti ili skupa vrednosti, toliko puta se izvrši i samo telo petlje

Petlje sa uslovom

- ovakve petlje podrazumevaju postojanje *uslova za ulazak u petlju* (logički uslov, tj. logički izraz) koji kontroliše kada će se ponavljanje koraka petlje zaustaviti, tj. do kada će trajati izvršavanje petlje
- petlja se izvršava tj. naredbe unutar petlje (telo petlje) se ponavljaju, sve dok je logički uslov koji definiše petlju ispunjen (rezultat logičkog izraza je **True**)
- prilikom započinjanja svake iteracije petlje proverava se ispunjenost uslova petlje i sve dok je uslov petlje ispunjen ide se na sledeću iteraciju petlje, a prvi put kada se dobije da uslov nije ispunjen prekida se izvršavanje petlje
- kod petlji sa uslovom, broj koraka petlje koji će se izvršiti nije unapred poznat
- kako uslov petlje može biti neispunjen na samom početku petlje (pre prvog koraka petlje), moguće je da se naredbe unutar petlje (telo petlje) ni jednom ne izvrše



Brojačke promenljive

- za imenovanje brojačkih promenljivih najčešće se koriste slova **i**, **j**, **k**, ... itd, a može biti i neka reč koja opisuje šta za šta je ta brojačka promenljiva vezana (šta ona tačno broji)
- brojačke promenljive se najčešće koriste za brojanje koraka petlji, ali nije im to jedina uloga
- brojačke promenljive mogu da odrbojavaju unapred ili unazad
- kada brojačka promenljiva treba da odbroji **n** koraka petlje, prilikom brojanja unapred brojač se najčešće inicijalizuje na **0*** i ide sve do vrednosti **n-1**, dok se prilikom brojanja unazad brojač inicijalizuje vrednošću **n-1** i ide sve do vrednosti **0**
- podrazumevano vrednost brojača se svaki put uvećava za **1**
- umesto da se uvećava za 1, čest je slučaj da se brojačka promenljiva uvećava i za neku drugu vrednost, čime se dobija nabrojanje elemenata nekog aritmetičkog niza (npr. uvećavanjem za 2 dobija se niz 0, 2, 4, 6, ... ili uvećavanjem za 5 dobija se niz 0, 5, 10, 15, ... i sl.)

while petlja (naredba)

- struktura while naredbe:

```
while (uslov) :  
    blok-naredbi
```

- uslov petlje može biti naveden u okviru zagrada a i ne mora
- kao uslov petlje može biti naveden bilo koji logički izraz (izraz čiji je rezultat logička konstanta **True** ili **False**) ili bilo koji numerički izraz čiji se rezultat implicitno kastuje u **True/False** (sve ne-nula vrednosti se kastuju u **True**, dok se vrednost **0** kastuje u **False**)
- sve naredbe unutar bloka naredbi koji čini telo petlje moraju biti uvučene sa jednim tabulatorom
- **while** naredba se najčešće koristi za realizaciju petlji sa uslovom, međutim može se koristiti i kao brojačka petlja kada se kao uslov za ulazak u petlju postavi logički izraz koji proverava da li je brojač dostigao određenu graničnu vrednost

Naredbe break i continue

- naredbe **break** i **continue** menjaju tok izvršavanja petlji
- naredbom **continue** prekida se izvršavanje tekuće iteracije petlje i nastavlja se na narednu iteraciju petlje, tj. na ispitivanje uslova za sledeću iteraciju petlje, dok se naredbom **break** prekida izvršavanje cele petlje i nastavlja se na prvu naredbu programa koja sledi nakon petlje
- ove naredbe se koriste isključivo unutar petlje, u suprotnom dovode do greške prilikom izvršavanja programa
- obe naredbe se mogu koristiti u kombinaciji i sa **while** i sa **for** petljom (naredbom)

Beskonačne petlje

- beskonačna petlja = petlja sa uvek ispunjenim uslovom (rezultat odgovarajućeg logičkog izraza je **True**)
- eksplicitno definisana beskonačna **while** petlja:

```
while True:
```

```
    blok-naredbi
```

- beskonačna petlja se izvršava neograničen broj puta (beskonačno puta), osim ako unutar bloka naredbi koji čini telo petlje ima neke naredbe koja prekida izvršavanje petlje (naredba **break** ili **return**)
- česta upotreba beskonačne petlje je u kombinaciji sa naredbom **break** kada je potrebno da se uslov petlje ne proverava na početku iteracije, već negde u okviru tela petlje, tj. u toku izvršavanja iteracije petlje
- beskonačne petlje koje se izvršavaju neograničen broj puta nemaju smisao i obično su posledica postojanja greške u kôdu - pogrešno formulisanog uslova petlje i/ili pogrešne kontrole toka izvršavanja iteracija petlje tako da nikada ne dođe do toga da uslov petlje postane neispunjen
- kada se pokrene program koji sadrži beskonačnu petlju koja se izvršava neograničen broj puta, program se neko vreme neće zaustaviti sve dok ne probije memorijska ograničenja računara
- za nasilno prekidanje izvršavanja programa koristimo **Ctrl + Z** komandu terminala

*u programiranju numeracija svega (karaktera stringa, elemenata liste, koraka petlje itd.) počinje od **0**