

Beleške sa časa – 11.nedelja

Terminologija

- struktura podataka = kolekcija (podataka)
- parametar = argument (funkcije)
- torka = uređeni niz dva ili više objekata

Tip podataka list

- tip **list** predstavlja *uređenu* kolekciju objekata istog ili različitog tipa
- uređene kolekcije podrazumevaju postojanje redosleda elemenata, tj. da je svakom elementu pridružen indeks koji određuje njegovu poziciju u kolekciji
- liste se koriste za čuvanje (skladištenje) podataka (objekata, elemenata) na strukturiran način, kao i svi kolekcijski tipovi generalno
- **kolekcije se još nazivaju strukturama podataka, s obzirom da pored pukog grupisanja podataka obezbeđuju neku vrstu strukture u podacima kao i različite operacije za baratanje sa podacima**
- elementi listi se zadaju u okviru uglastih zagrada [i] i međusobno su odvojeni zapetama
- kao što je već rečeno, lista može da sadrži elemente različitog tipa i to u okviru jedne iste liste, a takođe liste mogu sadržati i druplikate, tj. više elemenata sa istom vrednošću
- liste su, za razliku od stringova, mutabilni objekti, što znači da se listi može menjati sadržaj i nakon njenog kreiranja (postojećim elementima liste mogu se menjati vrednosti, kao i dodavati novi elementi i uklanjati postojeći elementi liste)
- moguće je definisati i tzv. *prazanu listu*, tj. listu koja ne sadrži ni jedan element
- nad objektima tipa **list** mogu se primeniti operator indeksiranja (operator []), slajsing operator (operator [:]), operatori pripadnosti (operatori **in** i **not in**), kao i operator konkatiranja (operator +)
- indeksiranje elemenata **list** objekata funkcioniše po istom principu kao indeksiranje karaktera kod stringova

Metodi klase list

- klasa koja definiše tip **list** sadrži razne zgodne metode za rad sa listama
- spisak svih metoda definisanih klasom **list** možete naći na sledećem linku:
https://www.w3schools.com/python/python_ref_list.asp
- neki od najčešće korišćenih metoda su:
 - **append** – dodaje element na kraj liste
 - **insert** – umeće element u listu na dati indeks (zadat kao argument metoda); u odnosu na indeks mesta dodavanja novog elementa svi postojeći elementi liste počev od tog indeksa pa do kraja liste se pomeraju za jedno mesto u desno
 - **pop** – uklanja element iz liste na datom *indeksu* (zadat kao argument metoda); ukoliko nije naveden indeks, tj. metod je pozvan bez argumenata, podrazumevano se uklanja element sa kraja liste
 - **remove** – uklanja element iz liste sa datom *vrednošću* (zadata kao argument metoda); ukoliko lista sadrži više elemenata sa datom vrednošću biće uklonjen samo prvi takav element (sa najmanjim indeksom); ukoliko lista ne sadrži element sa datom vrednošću dolazi do **ValueError** greške
 - **index** – vraća indeks elementa sa datom *vrednošću* (zadata kao argument metoda); ukoliko lista sadrži više elemenata sa datom vrednošću biće vraćen indeks prvog takvog elementa; ukoliko lista ne sadrži element sa datom vrednošću dolazi do **ValueError** greške
 - **NAPOMENA**: za obrnuti slučaj, kada znamo *indeks* elementa a hoćemo da dobijemo njegovu vrednost, ne postoji namenski metod, već se tada koristi operator indeksiranja
 - **count** – vraća broj elemenata u listi sa datom vrednošću
 - **reverse** – obrće redosled elemenata u listi
 - **sort** – sortira elemente liste u rastućem poretku; sortiranje u obrnutom, opadajućem poretku dobija se postavljanjem parametra **reverse** na **True** (podrazumevana vrednost je **False**); kako bi uopšte bilo moguće sortiranje liste, elementi moraju imati mogućnost poređenja pomoću operatora poređenja
 - **copy** – pravi kopiju liste, novi objekat liste sa istim sadržajem kao lista za koju je pozvan metod

- svi metodi ove klase **rade u mestu**, tj. ukoliko metod vrši nekakvu izmenu liste (dodavanje ili uklanjanje elemenata) ili transformaciju sadržaja liste (obrtanje ili sortiranje liste), to će biti izvršeno nad samim objektom nad kojim je metod i pozvan
- shodno tome, svi metodi koji vrše nekakvu izmenu/transformaciju liste nemaju povratnu vrednost
- nasuprot tome, postoje metodi koji **ne rade u mestu** – metodi koji izmene ne vrše nad samim objektom za koji su pozvani već nad njegovom kopijom, a novi objekat sa izvršenom izmenom vraćaju kao povratnu vrednost (kao što je slučaj kod metoda klase **str**)

Ugrađene funkcije za rad sa kolekcijama

- u standardnoj biblioteci dostupno je nekoliko ugrađenih funkcija za rad sa kolekcijskim tipovima, uključujući i liste:

- **len** – vraća broj elemenata u kolekciji
 - **sum** – vraća zbir elemenata u kolekciji; elementi kolekcije moraju imati mogućnost primene operatora **+**
 - **min** – vraća minimalni element iz kolekcije; elementi moraju imati mogućnost poređenja pomoću operatora poređenja
 - **max** – vraća maksimalni element iz kolekcije; elementi moraju imati mogućnost poređenja pomoću operatora poređenja
 - **sorted** – sortira elemente kolekcije u rastućem poretku; sortiranje u obrnutom, opadajućem poretku dobija se postavljanjem parametra **reverse** na **True** (podrazumevana vrednost je **False**); elementi moraju imati mogućnost poređenja pomoću operatora poređenja
 - **enumerate** – vrši enumeraciju elemenata kolekcije; enumeracija elemenata kolekcije podrazumevano počinje od 0, što odgovara indeksiranju elemenata kod listi, a može se promeniti zadavanjem parametra **start**; funkcija vraća objekat tipa **enumerate** koji predstavlja tzv. *iterator*, objekat koji omogućava iteriranje kroz kolekciju na određeni način; **enumerate** iterator izdvaja elemente kolekcije u obliku uređenih parova (**tuple** objekti) redni broj elementa i sam element; po potrebi rezultujući iterator se može kastovati u neki kolekcijski tip (npr. u listu)
 - **zip** - vrši uparivanje (objedinjavanje) elemenata dve ili više kolekcija po pozicijama (indeksima); ukoliko su dužine kolekcija različite, elementi će biti uparivani do kraja kolekcije najmanje dužine; funkcija vraća objekat tipa **zip** koji predstavlja tzv. *iterator*, objekat koji omogućava iteriranje kroz kolekciju na određeni način; **zip** iterator izdvaja elemente kolekcija u obliku uređenih parova, tj. opštije uređenih torki (**tuple** objekti) elemenata iz datih kolekcija; ova funkcija se najčešće koristi za paralelni prolazak (iteriranje) kroz elemente dve ili više kolekcija istovremeno; po potrebi rezultujući iterator se može kastovati u neki kolekcijski tip (npr. u listu)
- spisak svih ugrađenih funkcija standardne Python biblioteke možete naći na sledećem linku:
<https://docs.python.org/3/library/functions.html>

Kolekcijska **for** petlja i liste

- kolekcijskom **for** petljom može se iterirati kroz liste, prilikom čega tzv. *promenljiva petlje* uzima redom vrednosti jednog po jednog elemenata liste i za svaku od tih vrednosti se izvršavaju naredbe u okviru tela petlje
- preciznije rečeno, prilikom iteriranja **for** petljom kroz listu, promenljivoj petlje se redom dodeljuju vrednosti elemenata liste, tj. *kopije* objekata elementa liste
- shodno tome, sve eventualne izmene koje se vrše nad objektom promenljive petlje (promene vrednosti promenljive petlje) u okviru tela petlje neće se odraziti na same elemente liste
- dakle, ukoliko želimo da menjamo vrednosti elemenata liste, to se ne može izvršiti posredstvom kolekcijske **for** petlje, već brojačkom **while** ili **for** petljom po indeksima elemenata liste i *direktnim pristupom elementima liste pomoću operatora indeksiranja*

Generisanje listi (eng. *list comprehension*)

- list comprehension je mehanizam za generisanje novih listi na osnovu postojećih listi
- opšta struktura izraza za generisanje listi:
`newlist = [item_expression for item in existinglist if condition]`

- glavna komponenta izraza za generisanje listi je postojeća lista (**existinglist**) na osnovu koje se generiše nova lista (**newlist**) filtriranjem elemenata koji zadovoljavaju dati uslov (**condition == True**) uz primenu izraza **item_expression** koji na osnovu vrednosti izdvojenih elemenata postojeće liste izračunava vrednosti elemenata nove liste
- ovakav konstrukt Python jezika nam omogućava da kombinujemo iteriranje, modifikovanje i uslovno filtriranje elemenata liste u jednoj naredbi
- prethodno navedeni izraz za generisanje listi je ekvivalentan sledećem bloku koda:


```
newlist = []
for item in existinglist:
    if condition:
        newlist.append(item_expression)
```
- prednosti izraza za generisanje listi – jednostavnost i efikasnost kôda
- generisanje listi se koristi za:
 1. izdvajanje podliste elemenata koji zadovoljavaju neki uslov
 2. za kreiranje liste sa izmenjenim vrednostima elemenata u odnosu na elemente date liste
 3. kombinacija prethodna dva
- na mestu liste u okviru izraza za generisanje listi zapravo se može naći bilo koja druga vrste kolekcija ili iterator, tj. ovim mehanizmom se mogu generisati bilo koje vrste kolekcija, ne samo liste, ali ćemo mi ograničiti razmatranje ovog koncepta samo na liste

Varijante izraza za generisanje listi

- ukoliko ne želimo nikakvu modifikaciju vrednosti iz postojeće liste, tada se izraz **item_expression** sastoji samo od promenljive **item** (identičko preslikavanje, $f(x) = x$)


```
newlist = [item for item in existinglist if condition]
```
- deo izraza za generisanje listi koji se odnosi na filtriranje elemenata na osnovu nekog uslova je opcion
- varijanta izraza za generisanje listi bez filtriranja elemenata po uslovu:


```
newlist = [item_expression for item in existinglist]
```
- filtriranje elemenata liste na osnovu nekog uslova rezultuje izdvajanjem podliste od polazne liste, s tim što se taj uslov može odnositi isključivo na *vrednosti* elemenata liste, a ne i na njihove *indekse*
- ukoliko bismo želeli da izdvojimo samo one elemente liste čiji indeksi zadovoljavaju neki uslov, to bismo mogli da uradimo kombinovanjem funkcije **enumerate** sa generisanjem listi:


```
newlist = [item_expression for (index, item) in enumerate(existinglist) if index_condition]
```
- izraz **item_expression** može biti proizvoljnog oblika – može uključivati primenu svih operatora i funkcija, štaviše može sadržati u sebi i proveru nekog uslova, tj. može predstavljati uslovni izraz


```
newlist = [item_expression_1 if condition else item_expression_2 for item in existinglist]
```
- vrednost uslovnog izraza je **item_expression_1** ili **item_expression_2**, u zavisnosti od ispunjenosti uslova **condition** za pojedinačne elemente liste **item**
- dakle, ovde se uslovni izraz ne koristi u kontekstu filtriranja elemenata postojeće liste, već u kontekstu uslovnog pridruživanja vrednosti elementima nove liste na osnovu elemenata postojeće liste