**Liste**

Recimo da želimo da uzmemo trideset rezultata kolokvijuma i da nešto sa njima uradimo, recimo da ih sortiramo po broju poena. Mogli bi da kreiramo trideset varijabli score1, score2, …, score30, ali bi to bilo vrlo zamorno i dosadno. Još bi bilo teže urediti te varijable po velič**in**i. Rešenje za ovo su liste.

Evo jednog prostog primera:

L **=** [1, 2, 3]

Koristimo uglaste zagrade da označimo početak i kraj liste, a podatke u listi odvajamo zarezima.

Prazna liste je []. To je analogno sa 0 kod brojeva ili sa ‘’ “ kod stringova.

Ako imate neku dugačku listu, takvu listu može da napišete u više linija, kao što je prikazano:

nums **=** [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,

17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,

32, 33, 34, 35, 36, 37, 38, 39, 40]

Možemo koristiti funkciju **eval**(**input**()) kada od korisnika tražimo da unese listu.

Možete koristiti **print** funkciju za štampanje cele liste: **print**(L)

Lista može da sadrži bilo šta, pa čak i nove liste. Na primer ovo je potpuno ispravna lista:

[1, 2.718, ‘abc’, [5,6,7]]

Postoji niz stvari koje funkcionišu na isti način u listama kao u stringovima.

**len** — Funkcija koja daje dužinu liste L je **len**(L). **in** — Operator **in** određuje da li lista sadrži traženi podatak. Evo primera:

**if** 2 **in** L:

**print**('Vaša lista sadrži broj 2.')

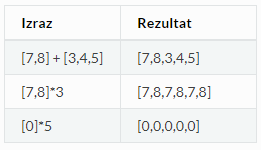
**if** 0 **not** **in** L:

**print**('Vaša lista ne sadrži nulu.')

Indeksiranje i segmentacija (slicing) — I ovo funkcioniše kao kod stringova. Na primer L[0] je prvi podatak liste L, a L[:3] daje prva tri podatka.

index i count — Ove metode rade na isti način kao kod stringova: L.index(x) vraća prvu poziciju podatka x u listi L (vraća -1 ako x nije u L), a L.count(x) vraća broj pojavljivanja x u L.

**`+`** i **`\*`** — Operator + dodaje jednu listu na kraj druge liste. Operator \* ponavlja listu određeni broj puta. Evo primera:



Poslednji primer je posebno koristan kada treba brzo napraviti listu nula.

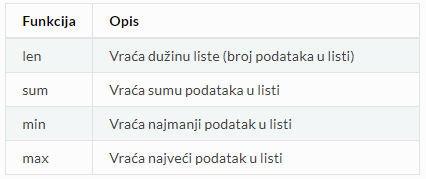
Petlje — Iste dve vrste petlji rade kako kod stringova tako i kod lista. U oba ova slučaja štampaju se podaci iz liste, jedan po jedan, svaki u posebnoj liniji.

**for** i **in** **range**(**len**(L)): **for** podatak **in** L:

**print**(L[i]) **print**(podatak)

Leva petlja je korisna kada želimo u petlji da koristimo i varijablu petlje. Ako to nije potrebno dovoljna je desno prikazana petlja, koja je nešto jednostavnija.

Postoji više ugrađenih funkcija koje rade sa listama. Ovde prikazujemo neke od najkorisnijih:



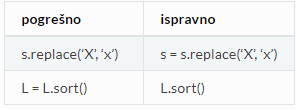
Na primer, sledeća naredba izračunava srednju vrednost svih brojeva iz liste L:

average **=** **sum**(L)**/len**(L)

Evo nekih metoda sa listama:



Postoji velika razlika između list i string metoda: String metode ne menjaju originalni string, dok list metode menjaju originalnu listu. Da sortiramo listu L, koristimo L.sort() a ne L=L.sort(). U stvari ova poslednja naredba neće uopšte raditi.



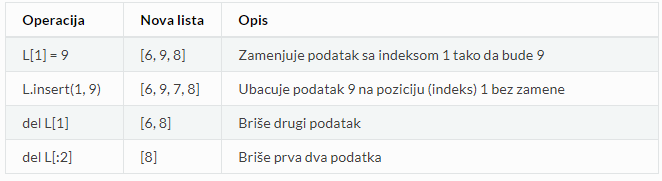
Postoji još nekoliko list metoda. Ukucajte help(list) u Python shell-u da vidite dokumentaciju o njima.

### Pravljenje kopije liste

Pravljenje kopije liste je po malo čudno zbog načina na koji to Python radi. Recimo da imamo listu L i da želimo da od nje napravimo kopiju koju ćemo zvati M. Izraz M=L neće raditi zbor razloga koji će biti kasnije objašnjeni. Za sada, umesto M=L koristite sledeću naredbu:

M **=** L[:]

Promena podataka u listi je mnogo jednostavnija nego kod stringova. Da promenimo podatak na poziciji 2 u listi L tako da bude 100, prosto napišemo L[2]=100. Ako želimo da ubacimo podatak čija je vrednost 100 na poziciju 2, a da pri tome ne uništimo podatak koji se tamo već nalazi, u tu svrhu koristimo metod insert. Da obrišemo podatak iz liste koristimo del operator. Neki primeri su prikazani dole. Pretpostavimo da je L=[6,7,8] za svaku od prikazanih operacija.



**Primer 1**

Program koji generiše listu L od 50 slučajnih brojeva između 1 i 100.

**from** random **import** randint

L **=** []

**for** i **in** **range**(50):

L**.**append(randint(1,100))

Koristili smo metod append za dodavanje podataka u praznu listu, []. Alternativa append metodu je sledeća naredba:

L **=** L **+** [randint(1,100)]

**Primer 2**

Menjanje svakog podatka iz liste L sa njegovim kvadratom.

**for** i **in** **range**(**len**(L)):

L[i] **=** L[i]**\*\***2

**Primer 3**

Prebrojavanje koliko podataka iz liste L je veće od 50.

brojac **=** 0

**for** pod **in** L:

**if** pod**>**50:

brojac **=** brojac**+**1

**Primer 4**

Data je lista L koja sadrži brojeve između 1 i 100. Kreiranje nove liste čiji je prvi element broj jedinica u listi L, drugi podataka, broj dvojki u listi L, i tako dalje.

frekvencije **=** []

**for** i **in** **range**(1,101):

frekvencije**.**append(L**.**count(i))

Ključna stvar je metoda count koja nam kazuje koliko se puta neki podatak pojavljuje u listi.

**Primer 5**

Program koji štampa dva najveća i dva najmanja broja iz liste koja se zove scores.

scores**.**sort()

**print**('Dva najmanja: ', scores[0], scores[1])

**print**('Dva najveća: ', scores[**-**1], scores[**-**2])

Jednom kada je lista sortiranja najmanje vrednosti se nalaze na početku a najveće na kraju liste.

**Primer 6**

Ovaj program služi za igranje jednostavnog kviza.

broj\_tacnih **=** 0

*# Pitanje 1*

**print**('Koji je glavni grad Francuske?', end**=**' ')

odgovor **=** **input**()

**if** odgovor**.**lower()**==**'pariz':

**print**('Tačno!')

broj\_tacnih**+=**1

**else**:

**print**('Pogrešno. Tačan odgovor je Pariz.')

**print**('Imate', broj\_tacnih, 'od 1 tačnih odgovora.')

*#Pitanje 2*

**print**('Sa koliko država se graniči Mađarska?', end**=**' ')

odgovor **=** **input**()

**if** odgovor**==**'7':

**print**('Tačno!')

broj\_tacnih**+=**1

**else**:

**print**('Pogrešno. Tačan odgovor je 7.')

**print**('Imate', broj\_tacnih, 'od 2 tačnih odgovora.')

Gornji program radi ali je naporan za pisanje. Ako želimo da dodamo još pitanja, moramo da kopiramo čitave blokove naredbi i da u njima menjamo mnoge stvari. Ako odlučimo da izmenimo strukturu kviza, kao na primer da ne prikazujemo tačne odgovore, tada se svaki blok koda mora menjati. Ovakvi programi se mogu značajno pojednostaviti korišćenjem lista i petlji, kao što je prikazano:

pitanja **=** ['Koji je glavni grad Francuske?',

'Sa koliko država se graniči Mađarska?']

odgovori **=** ['Pariz','7']

broj\_tacnih **=** 0

**for** i **in** **range**(**len**(pitanja)):

odgovor **=** **input**(pitanja[i])

**if** odgovor**.**lower()**==**odgovori[i]**.**lower():

**print**('Tačno')

broj\_tacnih**=**broj\_tacnih**+**1

**else**:

**print**('Pogrešno. Tačan odgovor je', odgovori[i])

**print**('Imate', broj\_tacnih, 'od', i, 'tačnih odgovora.')

Ako pažljivo pogledate ovaj program, videćete da je petlja skoro ista kao one u prethodnom programu, osim što koristi pitanja[i] i odgovori[i] umesto teksta pitanja.

**Ovo nam pokazuje jednu generalnu tehniku:** Ako vidite da ponavljate isti kod više puta, probajte da koristite liste i petlje.

Korist od svega toga (osim što je program kraći ) je i u tome kada menjamo ili dodajemo pitanja, treba samo promeniti liste pitanja i odgovori .Takođe, ako želite da menjate strukturu programa, kao na primer da ne kazujete tačne odgovore, sve što treba da uradite je da modifikujete jednu jedinu liniju koda, umesto mnogo kopija te iste linije u starom programu.