Programski jezik Python - prezentacija 2

–Rad sa stringovima i listama–

Miloš Brajović

Elektrotehnički fakultet
Univerzitet Crne Gore

Osnovi računarstva II

Stringovi

Miloš Brajović

Elektrotehnički fakultet
Univerzitet Crne Gore

Osnovi računarstva II

Rad sa stringovima

- Stringovi su nizovi karaktera i predstavljaju tip podatka.
- Za definisanje stringa, mogu se koristiti jednostruki navodnici (`'`), dvostruki navodnici (`"`) kao i trostruki navodnici (`"""`), koji se koriste u slučaju kada se string proteže u više linija.
- Navedimo nekoliko primjera **kreiranja** stringova:

```
s1='OR2'
s2="Dobar dan"
s3="""Život je neshvatljivo čudo, stalno se troši i osipa,
a ipak traje i stoji čvrsto - kao na Drini ćuprija"""
```

- Za **unošenje** stringa sa tastature, koristi se `input` funkcija, ali bez dodatne upotrebe `eval` funkcije:

```
s=input('Unesite neki tekst: ')
```

- **Prazan string**, odnosno, string koji nema karaktera, jednostavno se definiše u obliku `''`.

Osnovne operacije sa stringovima

- **Dužina stringa** se dobija korišćenjem funkcije `len`. Tako `len('Herman Hese')` kao rezultat daje 11.
- Već je istaknuto da se **nadovezivanje** stringova obavlja korišćenjem operatora `+`. Tako `'jedan'+'2'+'tri'` kao rezultat daje `'jedan2tri'`.
- **Ponavljanje** stringova se obavlja operatorom `*`. Tako `'#'*5` formira string `'#####'`.
- *Zadatak.* Napisati program koji od korisnika traži unos 10 slova, i koji formira string sastavljen samo od unijetih samoglasnika.

```
s = '';
for i in range(10):
  t = input('Unesite slovo: ')
  if t=='a' or t=='e' or t=='i' or t=='o' or t=='u':
    s = s + t
print(s) #A da su bile cifre umjesto samoglasnika?
```

Operator `in`

- Operator `in` omogućava ispitivanje da li neki string sadrži nešto ili ne.

```
if 'a' in s:
  print('U stringu s nema slova a).
```

- Regularno je koristiti `in` u kombinaciji sa `not`, ako se želi ispitati da li string nešto *ne* sadrži.

```
if '@' not in s:
  print('U stringu s nema karaktera @.')
```

- A može i:

```
s="""Tri Prstena za prste Kraljeva vilin-vrste
pod nebesima sto sjaju,
Sedam za vladare Patuljaka u dvoru njihovom kamnom,
Devet za Smrtne Ljude koje smrt čeka na kraju,
Jedan za Mračnog Gospodara na njegovom prijestolu tamnom
U Zemlji Mordor gdje Sjenke traju..."""
if 'Gospodar' in s:
  print('U stringu s postoji "Gospodar".')
```

Operator `in`. Indeksiranje

- *Zadatak.* Napisati program koji od korisnika traži unos 10 karaktera, i koji u string `s` smješta unijete cifre.

```
s = '';
for i in range(10):
  t = input('Unesite karakter: ')
  if t in '0123456789':
    s = s + t
print(s)
```

- **Indeksiranje**
  - Elementima stringa se može pristupiti korišćenjem zagrada `[]`.
  - Indeksiranje ide od 0.
  - U riječi `'Hajzenberg'`, `s[0]` ima vrijednost `'H'`, dok `s[1]` ima vrijednost `'a'`.
  - Karakteristično je da se posljednji element stringa može dobiti kao `s[-1]` (uporediti sa `end` u MATLAB-u) i to je u našem slučaju `'g'`, dok `s[-2]` daje vrijednost `'r'`. **Sa druge strane, `s[20]` će dati grešku!**

Segmentacija stringa i operator dvotačka (:)

- Python nudi mogućnost pristupanja djelovima stringa.
- Osnovna struktura je oblika [početna pozicija: krajnja pozicija + 1: korak]. Bilo koja od pozicija može biti izostavljena. Ako se izostavi početna pozicija, podrazumijevana vrijednost je 0 (ide se od početka stringa). Ako se izostavi krajnja pozicija – podrazumijeva se kraj stringa. Negativni indeksi odnose se na kraj stringa (udaljenost od kraja). Treći, opcioni, argument je korak.
- Posmatrajmo string `'Matematika'`.

```
indeks:     0 1 2 3 4 5 6 7 8 9
karakteri: M a t e m a t i k a
```

Kod	Rezultat	Objašnjenje
s[2:5]	`'tem'`	Slova čiji je indeks $2, 3, 4$
s[:5]	`'Matem'`	Prvih 5 slova (indeks 0 do 4)
s[5:]	`'atika'`	Slova od indeksa 5 do kraja
s[-2:]	`'ka'`	Posljednja dva slova
s[:]	`'Matematika'`	Cijeli string
s[1:7:2]	`'aea'`	Slova od 1 do 6 sa korakom 2.
s[: :-1]	`'akitametaM'`	Obrnut zapis stringa.

Rad sa stringovima

- **Pristup pojedinačnim karakterima**
  - Zanimljivo je naglasiti da u Python-u nije dozvoljeno mijenjati sadržaj stringa prostim pristupom u obliku: `s[4]='A'`.
  - Ako se želi u nekom stringu `s` karakter sa indeksom 4 (peto slovo) promijeniti u `'#'`, to se može postići kao u primjeru ispod.

```
s='Palindrom'
s=s[:4]+'#'+s[5:]
```

- **Petlje u radu sa stringovima**
  - Petlje nam mogu omogućiti da pristupimo pojedinim karakterima stringa.
  - *Zadatak:* Štampati slovo po slovo stringa `s`, svako u zasebnoj liniji:

```
for i in range(len(s)):
  print (s[i])
```

  - U argumentu funkcije `range` pozvana je funkcija `len`, koja vraća dužinu stringa u vidu broja.
  - Međutim, ako pozicija (indeks) nije bitna, isto se može postići i petljom:

```
for c in s: #Za svako slovo c u s
  print(c) # Stampaj slovo.
```

String metode

Metod – u Python terminologiji, to je funkcija koja je pridružena objektu. Više o objektima i metodima će biti riječi kasnije.

Metod	Opis
`lower()`	Vraća string sa svim slovima konvertovanim u mala slova
`upper()`	Vraća string sa svim slovima konvertovanim u velika slova
`replace(x,y)`	Vraća string kod kojeg su zamijenjena sva pojavljivanja `x` sa `y`
`count(x)`	Broji koliko se puta pojavljuje `x` u stringu
`index(x)`	Kao rezultat vraća indeks pozicije prvog pojavljivanja `x` u stringu
`isalpha()`	Vraća `True` ako je svaki karakter u stringu slovo (malo ili veliko)
`isdigit()`	Vraća `True` ako je svaki karakter u stringu cifra (a da nije prazan)
`isalnum()`	Vraća `True` ako je svaki karakter u stringu cifra ili slovo

- Metodi `lower`, `upper` i `replace` ne mijenjanju originalni string, već se izmjene moraju odraditi pridruživanjem:

```
s=s.lower()
```

- Još primjera:
  - `print(s.count('a'))` – štampa broj karaktera `a` u stringu `s`
  - `print(s.count(' '))` – štampa broj razmaka u stringu `s`

String metode

- Još primjera sa metodima:
  - `s = s.upper()` – mijenja string `s` tako što sva mala slova pretvara u velika
  - `s = s.replace('Zdravo', 'Hello')` – mijenja svako `'Zdravo'` u stringu `s` sa `'Hello'`
  - `print(s.index('a'))` – Štampa poziciju prvog pojavljivanja slova `'a'` u stringu `s`
- `isapha`, `isdigit`,`isalnum`
  - Navedene tri funkcije će vratiti vrijednost `True` ako je specifičan uslov ispunjen (vidjeti prethodnu tabelu) odnosno `False` ako nije ispunjen.
  - Vrijednosti `True` i `False` su **booleans** vrijednosti.
  - Jasno je da će se ove funkcije koristiti u `if` uslovima.

```
s = input('Unesite string')
if s[0].isalpha():
  print('Vas string pocinje slovom')
if not s.isalpha():
  print('U Vasem stringu ima karaktera koji nisu slova.')
```

String metode – još detalja

- `index`
  - Ako se metoda `index` pozove za nešto što ne postoji u stringu, doći će do greške.
  - Na primjer, `s='Marko', pozicija=s.index('b')` javiće grešku.
  - Stoga je poželjna upotreba selekcije kako bi se ovakve greške spriječile:

```
if 'b' in s:
  pozicija = s.index('b')
```

- Detalji oko string metoda u Python-u:
  - `dir(str)`
  - `help(str)`
  - `help(str.isalpha)`
- Escape karakter
  - Backslash karakter, `\`, koristi se za aktivaciju specijalnih karaktera.
  - `\n` označava prelazak u novi red:

```
print('Zdravo.\nKo ste Vi?')

Zdravo.
Ko ste Vi?
```

String metode – još detalja

- Dodavanje apostrofa u stringu
  - Da bi se dodao apostrof u stringu, treba koristiti `\'`

    ```
    s = 'I didn\'t hear.'
    ```

  - Moguće je koristiti i kombinaciju apostrofa sa duplim navodnicima:

    ```
    s = "I didn't hear."
    ```

  - Potpuno analogno se može koristiti i oblik `\"`.
- Dodavanje tab-a se postiže pomoću `\t`
- Ponekad je korisno znati kako se dodaje kosa crta, odnosno, `\\`, kao na primjer:

  ```
  s ='c:\neka_putanja\\proba.py'
  ```

- Poslednji primjer posebno će biti značajan pri radu sa fajlovima.
- Za vježbu, istražiti ostale metode u sklopu `str`.

Primjeri

- *Zadatak.* Odštampati deset praznih redova bez korišćenja `for` petlje.

```
print('\n'*9)
```

- Ne zaboraviti da `print` i sama daje jednu prazninu.
- *Zadatak.* Napisati program koji traži od korisnika da unese string, a onda štampa sve pozicije na kojima se nalazi slovo 'a' u stringu. String na kraju treba i da štampa broj pojavljivanja slova 'a'.

slovo_a.py

```
brojac=0
s = input('Unestite string: ')
for i in range(len(s)):
  if s[i]=='a':
    print(i)
    brojac=brojac+1
print('Imamo ukupno',brojac,'slova \'a\'')
```

Primjeri

- `slovo_a.py` – komentari
  - Kroz string se prolazi slovo po slovo.
  - Promjenljiva `i` prati indeks karaktera u stringu.
  - U četvrtoj liniji koda se radi provjera da li je tekući karakter na poziciji `i` zapravo slovo `'a'` i ako jeste, štampa mu se pozicija, dok se brojačka promjenljiva inkrementira.
  - Obratiti upotrebu escape karaktera u naredbi `print` na kraju programa.
- *Zadatak:* Napisati program koji od korisnika traži unos stringa, a koji zatim kreira (i štampa) novi string u kojem je ponovljeno svako slovo iz stringa koji je unio korisnik.

```
s=input('Unesite string: ')
duplirani=''
for i in range(len(s)):
  duplirani=duplirani+s[i]*2
print(duplirani)
```

Primjeri

- **`dupliraj.py`** – komentari
  - Uočiti da se u drugoj liniji koda novi string **`duplirani`** inicijalizuje na prazan string
  - Unutar **`for`** petlje se prolazi kroz string, karakter po karakter.
  - Uočiti da se nadovezivanje radi operatorom **`+`**, a dupliranje karaktera **`s[i]`** pomoću **`*2`**.
  - Ako korisnik unese **`pozdrav`**, kao rezultat se dobija **`ppoozzddrraavv`**.
  - Šta će se desiti ako se izostavi **`[i]`**?
- *Zadatak:* Napisati program koji od korisnika traži unos stringa, a koji zatim string štampa, prvo štampajući njegov prvi karakter, zatim prva dva, zatim prva tri itd. Na primjer, ako korisnik unese **`Marko`**, kao rezultat se dobija:

  `M Ma Mar Mark Marko`

```
ss=input('Unesite string: ')
for i in range(len(s)):
  print(s[:i+1],end=' ')
```

Primjeri

- **`stampanje_imena.py`** – komentari
  - U ovom zadatku smo koristili segmentiranje stringa. Broj slova koje štampamo se mijenja, pa se varijabla petlje, **`i`** mora iskoristiti pri segmentiranju. Pošto **`i`** počinje od 0, nama treba da štampamo prvih **`i+1`** karaktera, odnosno, **`s[:i+1]`**.
  - Kako bi svi segmenti bili štampani u istoj liniji, koristili smo opcioni argument **`end=' '`** u **`print`** funkciji.
- *Zadatak:* Napisati program koji od korisnika traži unos stringa **`s`** i koji smanjuje sva velika slova i uklanja sve znake interpunkcije iz unijetog stringa.

```
s=input('Unesite string: ')
s=s.lower()
for kar in ',.:;?!()\'"':
  s=s.replace(kar,'')
print(s)
```

Primjeri

- **`interpunkcija.py`** – komentari
  - U drugoj liniji koda, metod **`lower`** sva velika slova iz stringa **`s`** pretvara u mala.
  - U narednoj liniji koda, u **`for`** petlji, koristi se klauzula **`in`** pomoću koje promjenljiva **`kar`** uzima redom vrijednosti iz stringa **`',.:;?!()\'"'`**
  - **`s.replace(kar,'')`** mijenja pojavljivanje karaktera **`kar`** sa praznim stringom, **`''`**. Na taj način se karakter **`kar`** eliminiše iz stringa **`s`**.
- *Zadatak:* Napisati program koji od korisnika traži unos stringa **`s`** koji sadrži decimalni broj, i koji zatim štampa samo decimalni dio tog broja. Ako korisnik, na primjer, unese **`'2.7182818'`** (unosi se u vidu stringa!), kao rezultat se štampa: 7182818.

decimale.py

```
s=input('Unesite neki broj: ')
print(s[s.index('.')+1:])
```

Liste

Miloš Brajović

Elektrotehnički fakultet
Univerzitet Crne Gore

Osnovi računarstva II

Liste

- Često u praksi imamo niz vrijednosti koje želimo obrađivati. Na primjer, imamo 100 rezultata testa koje želimo sortirati. Jedan pristup bi bio da uvedemo skup varijabli oblika **`rezultat1`**, **`rezultat2`**,...
- U praksi se, međutim, uvode novi tipovi podataka, a jedan od njih su i **liste**.
- Za definisanje listi koriste se uglaste zagrade, odnosno, one označavaju početak i kraj liste, dok se pojedinačni elementi (*items*) liste odvajaju zarezima. Navedimo primjer:

```
lista=[100,200,300]
```

- Ponekad je, posebno u slučaju inicijalizacija, potrebno definisati praznu listu:

```
lista=[]
```

- Lista se može bez problema zapisivati u više redova:

```
lista=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
    17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
```

Liste

- Unošenje liste od strane korisnika obavlja se korišćenjem funkcije `input` u kombinaciji sa `eval`, u sljedećem obliku:

```
lista=eval(input('Unesite listu:'))
print('Unijeta lista je: ', lista)
```

- Korisnik unosi listu sa tastature uz korišćenje uglastih zagrada, sa elementima razdvojenim zarezom: `[1,2,3]`.
- Pojedinačnim elementima **se pristupa** slično kao kod stringova:

```
lista=[1,2,3,4,5,6,7]
print('Prvi element liste je: ', lista[0])
```

- Tipovi podataka u listi mogu biti raznorodni. Zato listu ne nazivamo nizom. Niz je specijalni slučaj liste u Python-u. Potpuno je regularno definisati listu oblika:

```
lista=[1,2.4343,'Marko Aurelije',[4,5,6,71]]
```

- Šta će biti smješteno u varijablama `x` i `a` ako je `lista` definisana iznad?

```
x=lista[2]
a=x[3]
```

Sličnost sa stringovima

- Funkcija `len`
  - Funkcija `len(lista)` daje broj elemenata liste `lista`.
  - Šta je rezultat u slučaju liste `len(lista)` sa prethodnog slajda?
- Klauzula `in`
  - I u ovom slučaju – određuje da li posmatrana lista sadrži posmatrani element:

```
L=eval(input('Unesite listu: '))
if 55 in L:
  print('Lista sadrzi broj 55.')
if 'Marko' not in L:
  print('Lista ne sadrzi rijec Marko.')
```

- Indeksiranje i segmentacija
  - `L[4]` kao rezultat daje peti element liste.
  - `L[:2]` kao rezultat daje prva dva elementa liste
- Funkcije (tj. metode) `index` i `count`
  - `lista.index(4)` kao rezultat vraća poziciju prvog pojavljivanja broja 4 u listi `lista`. U slučaju da broja 4 nema u listi – kao rezultat se dobija -1.
  - `lista.count('Marko')` kao rezultat daje broj pojavljivanja elementa `'Marko'` u posmatranoj listi.

Sličnost sa stringovima – nastavak

- Operator nadovezivanja (+) i ponavljanja (*)
  - Navedimo nekoliko primjera upotrebe:

Kod	Rezultat
`[5,3,4,2]+[11,4,2]`	`[5,3,4,2,11,4,2]`
`['Tev.',5.4]+[11,'t.']`	`['Tev.',5.4,11,'t.']`
`[1,2]*3`	`[1,2,1,2,1,2]`
`[22]*3`	`[22,22,22]`

- Upotreba `for` petlje
  - Jedan oblik podrazumijeva upotrebu brojačke varijable `i`:

```
for i in range(len(lista)):
  print(lista[i])
```

  - Dozvoljena je upotreba klauzule `in`

```
for podatak in L:
  print(podatak)
```

  - Jasno je da je prvi oblik petlje pogodno koristiti u situacijama kada se želi direktno koristiti indeks, odnosno, brojačka varijabla.

Ugrađene funkcije za rad sa listama

- Postoji više korisnih ugrađenih funkcija koje se mogu koristiti pri radu sa listama:

Naziv funkcije	Opis funkcije
`len`	Vraća dužinu liste, odnosno, broj elemenata liste
`sum`	Vraća sumu elemenata liste
`min`	Vraća podatak sa najmanjom vrijednošću
`max`	Vraća podatak sa najvećom vrijednošću

- Obratiti pažnju da nije dozvoljeno `min([1,3,"a",3])`
- Zanimljivo je uočiti i da `min(['a','aa','A','b'])` kao rezultat daje `'a'`. Šta mislite, zbog čega?
- Prosječnu vrijednost niza je, na primjer, moguće izračunati kao zbir elemenata niza `niz` podijeljen sa brojem elemenata tog niza `sum(niz)/len(niz)`
- Obratiti pažnju da se funkcijama, za razliku od metoda, pristupa direktno, bez korišćenja operatora tačka (`.`)
- Ne zaboraviti na mogućnost korišćenja pomoći – `help(min)`

Metode za rad sa listama

- Za razliku do funkcija, metodama pristupamo pomoću tačke.
- Navedimo nekoliko često korišćenih metoda (`help(list)`):

Naziv metoda	Opis metoda
`append(x)`	Podatak iz argumenta dodaje na kraj liste
`sort()`	Sortira listu
`count(x)`	Vraća broj pojavljivanja argumenta `x` u posmatranoj listi
`index(x)`	Vraća indeks prvog pojavljivanja elementa `x` u listi
`reverse()`	Obrće redosled elemenata liste
`remove(x)`	Briše prvo pojavljivanje elementa `x` iz liste
`pop(poz)`	Briše el. sa ind. `poz` iz liste i vraća njegovu vrijednost
`insert(poz,x)`	Umeće element `x` na poziciji `poz` u listi

- Kod list metoda postoji niz značajnih razlika u odnosu na string metode.
- String metode ne mogu promijenti string, dok list metode mogu promijeniti liste.
- Kod stringova je pogrešno `s.replace('X','x')` dok je ispravno `s = s.replace('X','x')`.
- Kod lista je pogrešno `L=L.sort()` dok je `L.sort()` ispravno.

Operacije sa listama – primjeri

- **Kopiranje liste**
  - Kopiranje liste se obavlja sa `M=L[:]`
  - Može i `M=L`
- **Promjena sadržaja liste**
  - Promjene u listama su jednostavnije nego u slučaju stringova
  - Pojedinačne elemente je moguće mijenjati direktno. Na primjer, treći element liste `L` mijenjamo u 5: `L[2]=5`. Umetanje se obavlja primjenom metoda `insert`.
  - Najlakši način za brisanje podrazumijeva korišćenje operatora `del`: tako `del L[1]` briše drugi element posmatrane liste.
  - Nekoliko primjera je navedeno u tabeli ispod, za slučaj liste `L=[4,3,5]`:

Operacija	Rezultat	Opis
`L[1]=14`	`L=[4,14,5]`	Element sa indeksom 1 zamjenjuje sa 14
`L.insert(1,14)`	`L=[4,14,3,5]`	Umeće 14 na poziciji 1, bez zamjene
`del L[1]`	`L=[4,5]`	Briše element na poziciji 2
`del L[2:]`	`L=[4,3]`	Briše poslednji element

Primjeri

Zadatak. Napisati program koji generiše listu od 30 slučajnih brojeva između 1 i 100 .

- Prvi način:

```
from random import randint
lista=[]
for i in range(30):
  lista.append(randint(1,100))
print('lista je:', lista)
```

- Drugi način:

```
from random import randint
lista=[]
for i in range(30):
  lista=lista+[randint(1,100)] #obratiti paznju na []
print('lista je:', lista)
```

- Prodiskutujte u čemu je razlika između navedena dva pristupa.

Primjeri

- *Zadatak.* Napisati program koji od korisnika traži unos niza (liste). Svaki element niza zamijeniti njegovom kvadriranom vrijednošću.

kvadriraj.py

```
lista=eval(input('Unesite niz: '))
for i in range(len(lista)):
  lista[i]=lista[i]**2
print(lista)
```

- Nemojte zaboraviti da upotrijebite funkciju `eval`.
- *Zadatak.* Napisati program koji od korisnika traži unos niza, i koji određuje i štampa koliko ima elemenata niza koji su veći od 10. Program takođe računa sumu ovih elemenata.
- Rješenje je dato na sljedećem slajdu.
- Za vježbu modifikovati rješenje tako da se koristi varijanta `for` petlje sa brojačkom varijablom.

Primjeri

broji.py

```
niz=eval(input('unesite niz: '))
brojac=0
suma=0
for el in niz:
  if el>10:
    brojac=brojac+1
    suma=suma+el
print('Elemenata ima:',brojac,'a suma je:',suma)
```

- *Zadatak.* Napisati program koji generiše listu od 100 slučajnih brojeva između 1 i 100. Program zatim broji koliko u ovoj listi ima brojeva 1, brojeva 2 itd. Ove podatke smješta u novu listu, čiji je prvi element broj jedinica, drugi element broj dvojki itd. Neka program takođe štampa dva najveća i dva najmanja elementa polazne liste.
- Rješenje je dato na narednom slajdu.

Primjeri

broji_sve.py

```
from random import randint
ponavljanja = []
L=[]
for i in range(1,101):
  L.append(randint(1,100))
for i in range(1,101):
  ponavljanja.append(L.count(i))
print(ponavljanja)
L.sort()
print('Dva najmanja su: ',L[0],L[1])
print('Dva najveca su: ',L[-1],L[-2])
```

- Metod `append` koristimo za dodavanje novih elemenata liste.
- `L.count(i)` daje koliko se puta broj `i` pojavljuje u listi `L`
- Uočiti da metod `sort` sortira elemente niza od najmanjeg do najvećeg.

Liste – naprednije teme

Miloš Brajović

Elektrotehnički fakultet
Univerzitet Crne Gore

Osnovi računarstva II

Liste – naprednije teme

- Liste i `random` modul

Naziv funkcije	Opis funkcije
`choice(L)`	Vraća slučajno pozicionirani element liste
`sample(L,n)`	Vraća grupu od `n` slučajnih elemenata iz liste
`shuffle(L)`	Miješanje (slučajni redosljed) elemenata liste (modifikuje listu!).

- **Primjer.** Slučajni odabir stringa iz liste strigova.

```
from random import choice
imena=['Marko','Jasmina','Mirza','Petar','Ana']
slucajno_ime=choice(imena)
print(slucajno_ime)
```

- **Primjer.** Sada modifikujmo program tako da daje dva slučajno odabrana imena.

```
from random import sample
imena=['Marko','Jasmina','Mirza','Petar','Ana']
slucajno_ime=sample(imena,2)
print(slucajno_ime)
```

Liste i random modul – nastavak

- **Primjer.** Napisati program koji iz zadatog stringa sastavljenog od svih karaktera, na ekranu ispiše 1000 slučajnih karaktera.

```
from random import choice
karakteri='abcdefghijklmnopqrstuvwxyz1234567890!@#$%^&*()'
for i in range(1000):
  print(choice(karakteri), end='')
```

- **Primjer.** Pretpostavimo da igrate igricu i da je iz skupa igrača potrebno slučajnim rasporedom birati jednog po jednog. Napisati program kojim se to realizuje.

```
from random import shuffle
imena=['Marko','Jasmina','Mirza','Petar','Ana']
shuffle(imena)
for igrac in imena:
  print('Sada je na redu',igrac)
```

Liste i random modul – nastavak

- **Primjer.** Napraviti program kojim se, iz zadatog skupa imena, slučajno biraju parovi igrača koji predstavljaju timove.

```
from random import shuffle
imena=['Marko','Jasmina','Mirza','Petar','Ana','Azra']
shuffle(imena)
timovi=[]
for i in range(0,len(imena),2):
    timovi.append([imena[i],imena[i+1]])
print(timovi)
```

- Obratiti pažnju da su elementi liste `timovi` zapravo liste od po dva elementa. U tom smislu, obratiti pažnju na uglaste zagrade u argumentu metode `append`.
- Treći argument funkcije range predstavlja korak, 2.
- U redu `shuffle(imena)` dolazi do izmjene redosljeda elemenata liste `imena`. Dakle, primjena ove funkcije mijenja samu listu.

Liste (naprednije teme) – `split`

- Metod `split` daje listu riječi u stringu. Podrazumijevani separator riječi je *space*.
- Ako je `s='Za mobilne aplikacije koristimo Kivy.'`, tada upotreba `L=s.split()` generiše listu `L` sastavljenu od pojedinačnih riječi:

  `['Za','mobilne','aplikacije','koristimo','Kivy.']`
- Uočimo da znaci interpunkcije ostaju dio riječi. Jedan zanimljiv način da oni budu eliminisani jeste upotreba `punctuation` stringa koji se može učitati iz biblioteke `string`:

```
from string import punctuation
for c in punctuation:
  s = s.replace(c, '')
```

- Metod `split` ima opcioni argument, karakter kojim se, umjesto podrazumijevanog *space*-a, definiše separator riječi:

```
s='89.188.33.45'
print(s.split('.'))
['89', '188', '33', '45']
```

Liste (naprednije teme) – `split` i `join`

- **Primjer.** Napisati program kojim se broji koliko se puta neka riječ pojavljuje u stringu. Korisnik unosi string i zadatu riječ po pozivu programa.

```
from string import punctuation

s = input('Unesite neki string: ')
for c in punctuation:
  s = s.replace(c, '')
s = s.lower()
L = s.split()
r = input('Unesite rijec: ')
print(r, 'se pojavljuje', L.count(r), 'puta.')
```

- Funkcija `list(s)`
  - string `s` konvertuje u listu čiji su elementi pojedinačni karakteri stringa.
- Metod `join`
  - Metod `join` omogućava spajanje liste stringova u jedan string.
  - String iz kojeg se poziva metod `join` služi kao separator spojenih stringova.

Liste (naprednije teme) – `join`

- Navedimo nekoliko primjera. Neka se posmatra lista stringova `L=['Jedan','Dva','Tri']`.
  1. `''.join(L)` daje `'JedanDvaTri'`
  2. `' '.join(L)` daje `'Jedan Dva Tri'`
  3. `','.join(L)` daje `'Jedan,Dva,Tri'`
  4. `'--'.join(L)` daje `'Jedan--Dva--Tri'`
- **Primjer.** Napisati program koji od riječi koju unosi korisnik, kreira anagram.

```
from random import shuffle
rijec=input('Unesite rijec: ')
lista_kar=list(rijec)
shuffle(lista_kar)
anagram = ''.join(lista_kar)
print(anagram)
```

- U prethodnom primjeru, prvo se uz pomoć funkcije `list`, polazni string se konvertuje u listu karaktera, a zatim se funkcijom `shuffle` vrši njihova preraspodjela tako da imaju slučajni raspored.

Liste (naprednije teme) – sastavljanje

- Sastavljanje (*comprehension*) lista predstavlja alat za formiranje lista.
- Listu je moguće napraviti, na primjer, u obliku

  `L=[i for i in range(1,5)]`. Rezultat je lista sa elementima:

  `[1,2,3,4]`.
- Posmatrajmo sljedeće varijable:

```
string = 'Pozdrav'
L = [5,24,5,19,12]
M = ['jedan', 'dva', 'tri', 'cetiri', 'pet', 'sest']
```

- Na osnovu njih, formiraju se i štampaju :

Naredba	rezultat
`[0 for i in range(10)]`	`[0,0,0,0,0,0,0,0,0,0]`
`[i**2 for i in range(1,8)]`	`[1,4,9,16,25,36,49]`
`[i*10 for i in L]`	`[50,240,50,190,120]`
`[c*2 for c in string[0:4]]`	`['PP','oo','zz','dd']`
`[m[0] for m in M]`	`['j','d','t','c','p','s']`
`[i for i in L if i<10]`	`[5,5]`
`[m[0] for m in M if len(m)==3]`	`['d','t','p']`

Liste (naprednije teme) – sastavljanje

- Zanimljivo je uočiti da je ekvivalent naredbi

  `[m[0] for m in M if len(m)==3]` zapravo:

```
L = []
for m in M:
  if len(m)==3:
    L.append(m[0])
```

  što znači da *comprehension* nudi mnogo elegantniju sintaksu.

- **Višestruki `for`**
  - Uvedena notacija može poslužiti i za zamjenu višestrukih petlji. Na primjer `L = [[i,j] for i in range(2) for j in range(2)]` kao rezultat daje `[[0,0],[0,1],[1,0],[1,1]]`. Korišćenjem klasičnog pristupa, pisali bi:

```
L = []
for i in range(2):
  for j in range(2):
    L.append([i,j])
```

  - Slično tome, `[[i,j] for i in range(4) for j in range(i)]` daje `[[1, 0], [2, 0], [2, 1], [3, 0], [3, 1], [3, 2]]`.

Liste (naprednije teme) – sastavljanje – primjeri

- **Zadatak.** Generisati listu od 30 slučajnih brojeva između 1 i 50.

```
from random import randint
L=[randint(1,50) for i in range(30)]
print(L)
```

- **Zadatak.** Zamijeniti svaki član liste sa njegovom kvadriranom vrijednošću.

```
L=eval(input('Unesite listu: '))
L=[i**2 for i in L]
print(L)
```

- **Zadatak.** U listi koju unese korisnik, prebrojati koliko je elemenata koji su veći od 3.

```
L=eval(input('Unesite listu: '))
print(len([i for i in L if i>3]))
```

- Sve prethodne primjere riješiti i primjenom `for` petlje.

Liste (naprednije teme) – sastavljanje – primjeri

- **Zadatak.** Generisati listu od 100 slučajnih brojeva između 1 i 100. Zatim na osnovu nje generisati drugu listu čiji je prvi element broj jedinica, drugi broj dvojki itd., u unijetoj listi.

```
from random import randint
L=[randint(1,100) for i in range(100)]
brojeva = [L.count(i) for i in range(1,101)]
print(brojeva)
```

- **Zadatak.** Generisati string sa 1000 slučajno raspoređenih slova.

```
from random import choice
abeceda='abcdefghijklmnopqrstuvwxyz'
s=''.join([choice(abeceda) for i in range(1000)])
print(s)
```

- **Zadatak.** Neka je data lista u obliku `L = [[1,2], [3,4], [5,6]]`. Zamijeniti redosljed elemenata u svakoj od podlista.

```
L=[[1,2],[3,4],[5,6]]
O=[[x[1],x[0]] for x in L] #ili O=[[y,x] for x,y in L]
print(O)
```

Dvodimenzione liste i matrice

- Matrice nijesu definisane kao ugrađeni tip podatka u Python-u. Međutim, liste mogu biti tretirane kao matrice:

```
L = [[1,2,3],
     [4,5,6],
     [7,8,9]]
```

- Pojedinim elementima vrste i kolone se pristupa u obliku `L[i][j]`.
- Na primjer, `L[0][1]` predstavlja element 2.
- **Štampanje dvodimenzione liste.** Prethodno kreiranu 2D listu možemo štampati na sljedeći način:

```
for i in range(3):
  for j in range(3):
    print(L[i][j],end=' ')
  print()
```

- Alternativno, može se koristiti i funkcija `pprint`:

```
from pprint import pprint
pprint(L)
```

Dvodimenzione liste

- **Zadatak.** Napisati program koji pravi 2D listu od 10×5 slučajnih brojeva iz opsega od 1 do 100, koji pronalazi i štampa koliko ima parnih brojeva.

```
from random import randint
M=10
N=5
L=[]
for i in range(M):
  L.append([randint(1,100) for i in range(N)])
BP=0#Odredimo koliko ima parnih elemenata:
for i in range(M):
  for j in range(N):
    if L[i][j]%2==0:
      BP=BP+1
print('Parnih elemenata je: ',BP)
print('Razmatra se lista: ')#stampanje liste:
for i in range(M):
  for j in range(N):
    print(L[i][j],end=' ')
  print()
```

Dvodimenzione liste

- Prethodno prebrojavanje je moglo biti obavljeno korišćenjem:

```
BP = sum([1 for i in range(M) for j in range(N) if L[i][j]%2==0])
```

- Kreiranje velike liste sastavljene od nula (pogodno za inicijalizaciju), može biti obavljeno pomoću

```
L = [[0]*50 for i in range(100)]
```

- Obratiti pažnju da `[0]*50` označava da se element ponavlja 50 puta. Dakle, rezultujuća lista ima 100 vrsta i 50 kolona.
- Pristup jednoj vrsti matrice se može obaviti pomoću `L[i]`
- Pristup koloni matrice je nešto složeniji:

```
[L[i][j] for i in range(len(L))]
```

- Pretvaranje 2D liste u 1D listu: `[j for M in L for j in M]`
- Lista dimenzija $5 \times 5 \times 5$ sastavljena od nula može biti kreirana u obliku

```
L = [[[0]*5 for i in range(5)] for j in range(5)]
```

- Prvi element ove liste je: `L[0][0][0]`.

Literatura

- Brian Heinold, *A Practical Introduction to Python Programming*, 2012, [online] https://www.brianheinold.net/python/python_book.html
- Allen B. Downey, *Python for software design*, Cambridge University Press, 2009.
- https://www.w3schools.com/python/default.asp, poslednji put pristupano 5. marta 2020. godine