Praktikum iz programiranja 3

2023/24

O kursu

- **Model 70+30**
  - Redovna izrada domaćih zadataka = 4 poena
  - 1 kolokvijuma = 66 poena
  - Završni deo ispita = 30 poena
- **Uslov za izlazak na završni ispit – više od polovine poena na predispitnim obavezama**

Najvažnije teme

- Osnove Python-a – podsećanje
  - Aritmetičke operacije, upravljačke komande, standardni ulaz/izlaz, ugrađene funkcije, stringovi, liste, funkcije
- Strukture - liste, skupovi, torke, rečnici, manipulacija podacima
- NumPy, MathPlotLib, Pandas, regex
- Standardni algoritamski problemi

Python
Promenljive

- Ime promenljive sme sadržati samo
  - Brojeve
  - Velika i mala slova
  - Donju crticu _
- Ime ne sme početi brojem
- Python razlikuje velika i mala slova

Neka od dozvoljenih imena:

```
jabuka, slova123, A4, ime_psa
```

Neka od nedozvoljenih imena:

```
ime psa, 123slova, jabuka#
```

Python
Aritmetičke operacije

- Python kao kalkulator

Znak	Operacija
+	Sabiranje
-	Oduzimanje
*	Množenje
/	Deljenje
%	Ostatak pri deljenju
//	Celobrojno deljenje
**	Stepenovanje

```
>>> 2 + 3
5
>>> 2 - 3
-1
>>> 2 * 3
6
>>> 9 / 4
2.25
```

```
>>> 2 + 3 * 6
20
>>> 2 + 4 * 9 - 3
35
>>> 27 // 10
2
>>> 27 / 10
2.7
>>> 27 % 10
7
```

Python
Tipovi i vrednosti

- Dodelom vrednosti se određuje tip neke promenljive. Ukoliko nismo sigurni kog je tipa neki podatak, interpreter nam to može reći

```
>>> poruka = "Programiranje je lepo"
>>> pi = 3.14
>>> n = 5
>>> poruka
'Programiranje je lepo'
>>> pi
3.14
>>> n
5
```

```
>>> type(poruka)
<class 'str'>
>>> type(pi)
<class 'float'>
>>> type(n)
<class 'int'>
```

tip	oznaka	domen vrednosti	primeri vrednosti
celobrojni	int	celi brojevi	0, 1, -1243
realni	float	realni brojevi	1.25, .02, -145.25
komleksi	comlex	kompleksi brojevi	-3j, -0.5 – 4.25j
logički	bool	{ False, True }	False, True
tekstualni	str	niz znakova	'Ucim programiranje'
torka	tuple	nepromenljiv niz proizvoljnih objekata	(), (1.25, True, 'abc')
lista	list	promenljiv niz proizvoljnih objekata	[],[1, 2], ['A', 1, 1 + 2j]
skup	set	neuređen skup proizvoljnih nepromenljivih objekata. Duplikati nisu dozvoljeni	{},{1, 2}, {'A', 1, {1, 2}}
rečnik	dict	parovi ključ-vrednost	{11: 'Beograd', 21:'Novi Sad', 34:'Kragujevac'}
nedefinisano	NoneType	{ None }	None

Python
Logičke operacije

Znak	Relacija
And	Logičko „i“
Or	Logičko „ili“
Not	Logičko „ne“

```
x     | y     | x and y | x or y
True  | True  | True    | True
True  | False | False   | True
False | True  | False   | True
False | False | False   | False
```

- Tabela istinitosti logičkih operacija and i or

```
print(' x     | y     | x and y | x or y')
print('True  | True  |', True and True, '   |', True or True)
print('True  | False |', True and False, '  |', True or False)
print('False | True  |', False and True, '  |', False or True)
print('False | False |', False and False, '  |', False or False)
```

Python
Operatori

Znak	Relacija
>	Veće
<	Manje
>=	Veće i jednako
<=	Manje i jednako
==	Jednako
!=	Različito

- Ispisati veći od uneta dva realna broja

```
x = float(input('x= '))
y = float(input('y= '))
print('veći broj je', x * (y < x) + y * (x <= y) )
```

Python
Grananje

- Neke naredbe se izvršavaju samo ako je neki uslov ispunjen. Da bi se opisalo uslovno izvršavanje nekih naredbi koristi se naredba `if`

```
if uslov:
    naredba_1
    ...
    naredba_k
```

```
if uslov:
    naredba_1
    ...
    naredba_m
else:
    naredba_1
    ...
    naredba_n
```

```
if uslov1:
    naredba_1
    ...
    naredba_m
elif uslov2:
    naredba_1
    ...
    naredba_n
else:
    naredba_1
    ...
    naredba_k
```

- Zagrade u uslovu nisu OBAVEZNE.
- Dvotačka je obavezna nakon navođenja uslova
- Naredba u uslovu će vratiti tačno/netačno
- Ako je tačan uslov koji se zadaje izvršiće se naredba_1 ...

Python
Ponavljanje

- Python pruža mehanizam da se jedna ili više naredbi ponovi:

- određen broj puta

```
for brojač in opseg :
    telo_petlje
```

početna vrednost

završna vrednost koja se ne uzima u obzir

```
for i in range(a,b,k) :
    naredba 1
    naredba 2
```

korak

- dok je neki uslov ispunjen

```
while uslov :
    telo_petlje
```

Python
Standardni ulaz/izlaz

- Upisivanje tekstualne ili brojčane vrednosti je moguće pomoću naredbe `input()`.
- Rezultat izvršenja naredbe `input()` mora se pridružiti nekoj promenljivoj.
- Tip prihvaćenih podataka je uvek `string`, ukoliko se ne zahteva drugačije.

```
a=tip_podatka(input("Unesi vrednost"))
```

naziv promenljive

tip promenljive: int, float, bool ili string

naredba za unos

tekstualni deo kao poruka, ne mora da postoji

- Funkcija za ispis je `print()`:

```
print("Ovo je tekst ","a ovo je broj ",30)
```

Python
Ugrađene funkcije

- `min, max, abs`

```
print(min(2.3,5.8),max(8,12))
broj = -3
print(max (abs (broj), min (2, 4)))
```

- Matematičke funkcije

```
import math as m
print(m.factorial(4))
```

`pow, sqrt, floor, trunc, exp, log, sin, cos,...`

Python
Stringovi

- String je niz karaktera ograničen jednostrukim ili dvostrukim navodnicima, ali mora početi i završiti se istim navodnicima

```
s = 'On je rekao, "Zdravo, svete!"'
```

- Specijalni karakteri
  - \n – završava trenutnu liniju i nastavlja u sledećoj
  - \t – ubacuje tab u string
  - \' – ubacuje ' u string
  - \" – ubacuje " u string
  - \\ – ubacuje \ u string

- String se može štampati u više linija
  - Ako se započne sa tri navodnika (jednostruka ili dvostruka) i tako i završi
  - Ako se u string umetne specijalni karakter \n

```
s1 = """Ovo
je string koji ima
vise linija."""
```

```
s2 = 'Ovo\ntakodje'
```

Python
Stringovi

- Dva stringa se spajaju u jedan korišćenjem operatora +
- Više stringova se spajaju u jedan, na isti način, korišćenjem operatora +

```
s0 = "Spajanje"
s1 = "vise"
s3 = "zajedno"
s = s0 + " " + s1 + " " + s2 + " " + s3
```

- Množenje stringa nulom i negativnim brojem ne daje nikakav rezultat
- Nije dozvoljeno stringa množenje razlomljenim brojem i sabiranje sa celim brojem

```
s = 'Ha'
print (s * 10)
a = 'Program'
print(a * 3)
```

```
HaHaHaHaHaHaHaHaHaHa
ProgramProgramProgram
```

Python
Deljenje stringova

- Delovima stringa može se pristupiti preko indeksa navedenog u zagradama [ ]

```
s = 'programiranje'
print ('s = ', s)
print ('s[0] = ', s[0],' s[3] = ', s[3])
print ('s[-1] = ', s[-1])
print ('s[1:5] = ', s[1:5])
print ('s[5:-2] = ', s[5:-2])
```

```
s =  programiranje
s[0] =  p  s[3] =  g
s[-1] =  e
s[1:5] =  rogr
s[5:-2] =  amiran
```

Python
String funkcije

- Dužina stringa – `len`
- String u stringu – `find, index`
- Broj ponavljana stringa u stringu – `count`
- Konvertovanje svih slova u velika, odnosno mala – `upper, lower`
- Promena dela stringa novim stringom – `replace`
- Za određivanje tipa podataka koristi se funkcija `type`
- Konverzija u string – `str`
- Konverzija u ceo broj – `int`
- Konverzija u realan broj – `float`

Python
Liste

- Lista je niz podataka proizvoljnog tipa. Svaki podataka u listi se naziva `element`

```
[1, 4, 17, 256, 3, 59, 45]- lista celih brojeva
['april', 'jun', 'septembar', 'novembar']    - lista stringova
[]    - prazna lista
['Marina', 28, 101, 'PMF']- lista elemenata različitog tipa
[1, 4, [7, 6, [13]], [9, 5]]    - lista čiji elementi su liste
```

- Elementim liste se pristupa preko rednog broja pozicije – `indeksa`
- Indeks prvog elementa u listi je `0`

Python
Liste

- Spajanje listi

```
A = [1,2,3,4,5]
B = [6,7,8,9]
C = A + B
```

- Direktna promena vrednosti elemenata liste

```
lista = [2, 4, 6, 8]
lista[1] = 0
lista[-1] = 12
```

Python
Liste

- Elementi liste se mogu ponavljati određen broj puta operatorom *

```
C = [0]
A = [1,2,3]
B = 3 * A
D = 4 * C
print(B, D)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3] [0, 0, 0, 0]
```

Python
Izdvajanje elemenata liste

```
lista = [10, 9, 8, 7,-1,-2,-3]
print (lista[2], lista[-1])
print (lista[1:5])
print (lista[1:5:2])
print (lista[12])
```

```
8 -3
[9, 8, 7, -1]
[9, 7]
IndexError: list index out of range
```

Python
Funkcije za rad sa listama

- Dužina liste, odnosno broj elemenata liste – `len`
- Dodavanje elemenata na kraj liste – `append, extend`
- Dodavanje elemenata na željenu poziciju – `insert`
- Minimalni i maksimalni element liste – `min, max`
- Broj ponavljanja elementa – `count`
- Brisanje elementa iz liste – `del, pop, remove`
- Sortiranje i okretanje elemenata liste – `sort, reverse`

Python
Filtriranje listi

```
temperature = [2, -1, 0, -8, -10, -1, 4, 5, 8, 6]
pozitivne_temperature = [t for t in temperature if t > 0]
print(pozitivne_temperature)
[2, 4, 5, 8, 6]
```

- Zadati tekst ispisati tako što bi se izostavila pojava samoglasnika

```
tekst = "Praktikum iz programiranja"
novi = [a for a in tekst if a not in {'a','e','i','o','u'}]
for slovo in novi:
        print(slovo, end='')
```

Python
Funkcije

- Python pruža mogućnost definisanja korisničkih funkcija

```
def ime_funkcije(parametri_funkcije) :
    telo funkcije
```

- Funkcija mora biti definisana pre prvog korišćenja(poziva)

```
def stampaj_izjavu():
    print ("Moje ime je Ana i OK sam.")
    print ("Spavam celu noc i radim ceo dan.")

stampaj_izjavu()

print (stampaj_izjavu)
print (type(stampaj_izjavu))
```

```
Moje ime je Ana i OK sam.
Spavam celu noc i radim ceo dan.
<function stampaj_izjavu at 0x0341FE40>
<class 'function'>.
```

Python
Funkcije sa parametrima

```
def stampaj_2puta(param):
    print(param)
    print(param)

stampaj_2puta('Spam')
stampaj_2puta(math.pi)
stampaj_2puta('Spam'*4)
stampaj_2puta(17)
```

```
Spam
Spam
3.141592653589793
3.141592653589793
SpamSpamSpamSpam
SpamSpamSpamSpam
17
17
```

Python
Funkcije

- Funkcija može da ima jednu ili više povratnih vrednosti
- Rezultat izvršenja funkcija koje imaju povratne vrednosti mora biti dodeljen promenljivima na mestu poziva

```
def saberi (a,b):
    c = a + b
    return c

def deli (a,b):
    c = a // b
    d = a % b
    return (c, d)
```

Python
Rečnici

- Rečnici su generalizovana verzija liste. Posmatrajmo listu koja sadrži broj dana u mesecima godine

```
dana = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

- Ako hoćemo broj dana u mesecu januaru koristimo `dana[0]`, ili u decembru `dana[11]` ili `dana[-1]`
- Rečnik dana u mesecima godine:

```
dani ={'Januar':31, 'Februar':28, 'Mart':31, 'April':30,
       'Maj':31, 'Jun':30, 'Jul':31, 'Avgust':31,
       'Septembar':30, 'Oktobar':31, 'Novembar':30, 'Decembar':31}
```

- Sada pristupamo sa:

```
dani['Januar']
```

- Prednost: kod je čitljiv i ne treba razmišljati o indeksu

Python
Osnovne operacije sa rečnicima

- Da označimo da je nešto rečnik koristimo zagrade `{}`
- Svaki element u rečniku je par podataka razvojen sa dve tačke
- Prvi deo u paru se naziva `ključ (key)`, drugi je `vrednost (value)`
- Ključ igra ulogu sličnu indeksu

```
d = {'A':100, 'B':200}
```

ključ

vrednost

```
d['A'] = 100
```

- Ključevi su često stringovi, mogu biti i celi i realni brojevi
- U istom rečniku mogu se naći ključevi različitog tipa

Python
Osnovne operacije sa rečnicima

```
d = {'A':100, 'B':200}
```

- Promena vrednosti:

```
d['A'] = 400
```

- Dodavanje novog elementa u rečnik:

```
d['C'] = 500
```

Ovo nije moguće sa listama. Ako napišem L[2]=500 za listu L koja ima samo dva elementa L[0] i L[1], dobićemo grešku: „index out of range“

- Brisanje elementa iz rečnika:

```
del d['A']
```

Python
Osnovne operacije sa rečnicima

- Prazan rečnik je `{}`, čemu je analogno `[]` za praznu listu, ili `''` za prazan string.
- Redosled elemenata u rečniku ne mora biti isti kao redosled dodavanja elemenata u rečnik
- Python rearanžira elemente rečnika u cilju optimizacije performansi.

Python
Osnovne operacije sa rečnicima

- U igri Scrabble, svakom slovu je pridružena vrednost. Možemo koristiti sledeći rečnik za vrednost slova u njemu:

```
points = {'A':1, 'B':3, 'C':3, 'D':2, 'E':1, 'F':4, 'G':2,
          'H':4, 'I':1, 'J':8, 'K':5, 'L':1, 'M':3, 'N':1,
          'O':1, 'P':3, 'Q':10, 'R':1, 'S':1, 'T':1, 'U':1,
          'V':4, 'W':4, 'X':8, 'Y':4, 'Z':10}
```

- Da izračunamo skor za neku reč koristimo kod

```
skor = sum([points[c] for c in word])
```

Python
Osnovne operacije sa rečnicima

Rečnik omogućava lepu varijantu za prikaz špila karata:

```
špil = [{'vrednost':i, 'boja':c}
        for c in ['pik', 'tref', 'herc', 'karo']
        for i in range(2,15)]
```

Špil je lista sa 52 rečnika. Metod `shuffle` se može iskoristiti za mešanje špila:

```
shuffle(špil)   //  from random import shuffle
```

Prva karta u špilu je `špil[0]`. Boja i vrednost karte:

```
špil[0]['vrednost']
špil[0]['boja']
```

Python
Rad sa rečnicima

```
d = {'A':100, 'B':200}
```

- Kopiranje rečnika:

```
d2=d.copy()
```

- Operator `in` se koristi da nam kaže da li je neki ključ u rečniku ili nije.
- Ako pokušamo da dobijemo vrednost nekog ključa koji nije u rečniku dobićemo grešku.
- `print(d['C'])  - greška`

- Možemo da sprečimo grešku korišćenjem operatora `in`. Pre upotrebe nekog ključa, možemo da proverimo da li je u rečniku.

Python
Rad sa rečnicima

```
slovo = input('Unesite slovo: ')
if slovo in d:
    print('Vrednost je', d[slovo])
else:
    print('Nije u rečniku')
```

Može se koristiti i `not in`

Python
Rad sa rečnicima

```
tabela = {1 : 'Srbija', 2 : 'Italija', 3 : 'Francuska', 4 : 'Spanija’}
```

Metoda `pop()` briše pridruživanje za zadati ključ i tom prilikom vraća vrednost koja se izbacuje iz rečnika.

```
print(tabela.pop(2), tabela)
Italija {1: 'Srbija', 3: 'Francuska', 4: 'Spanija’}
tabela.pop(5)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    tabela.pop(5)
KeyError: 5
```

Python
Rad sa rečnicima

```
tabela = {1 : 'Srbija', 2 : 'Italija', 3 : 'Francuska', 4 : 'Spanija’}
```

- Kada se u metodi navede opcioni parameter greška se ne pojavljuje:

```
tabela.pop(5, None)
None
```

- Brisanje svih elemenata iz rečnika:

```
tabela.clear()
print(tabela)
{}
```

Python
Rad sa rečnicima

```
d = {'A':100, 'B':200}
```

- Može se napraviti petlja kroz rečnik na sličan način kao i kod liste.
- Štampanje svih ključeva rečnika:

```
for kljuc in d:
    print(kljuc)
```

- Štampanje vrednosti iz rečnika:

```
for kljuc in d:
    print(d[kljuc])
```

Python
Rad sa rečnicima

```
d = {'A':100, 'B':200}
```

- Kako od rečnika dobiti liste ključeva i vrednosti:

```
list(d)   -  ['A','B'], ključevi iz rečnika d
list(d.values())  -  [100,200], vrednosti iz rečnika d
list(d.items()) -  [('A',100),('B',200)], parovi (ključ,vrednost)
```

- Parovi koje vraća `d.items()` nazivaju se `torke (tuples)`. Torke su slične listama.

Python
Rad sa rečnicima

- Svi ključevi rečnika kojima odgovara vrednost 100.

```
d = {'A':100, 'B':200, 'C':100}
L = [x[0] for x in d.items() if x[1]==100] #filtriranje liste ključeva
```

```
['A', 'C']
```

Python
Rad sa rečnicima

- Funkcija `dict` je dodatan način kreiranja rečnika.

```
d = dict([('A',100),('B',300)])
{'A':100, 'B':300}
```

- `Sastavljanje rečnika` je slično sastavljanju liste. Sledeći jednostavan primer kreira rečnik od liste reči u kojem će vrednosti u rečniku biti dužine odgovarajućih reči:

```
reči=['jabuka','banana','jagoda']
d = {s : len(s) for s in reči}
{'jabuka': 6, 'banana': 6, 'jagoda': 6}
```

Python
Torke (Tuples)

- Torka je nepromenljiva lista – ne može da se menja po formiranju

```
L = [1, 2, 3]
T = (1, 2, 3)
```

- Torke se navode unutar malih zagrada ili bez njih.
- Indeksiranje i segmentacija su isti kao i kod liste.
- Kao i kod liste funkcijom `len` možete dobiti dužinu torke.
- Torke imaju kao i liste `count` i `index` metode.
- Međutim, pošto su torke nepromenljive, torke nemaju neke metode koje imaju liste, kao što su `sort` ili `reverse`.

Python
Torke (Tuples)

```
          nole = ('Novak', 'Đoković', 1987, 'Srbija', 12)
>>> nole[4]
12
>>> nole[4]=13     #greška
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    nole[4]=13
TypeError: 'tuple' object does not support item assignment
>>> nov_nole = nole[0:3] + (13,)
>>> nov_nole
('Novak', 'Đoković', 1987, 13)
>>>
```

Python
Torke (Tuples)

- Jedan od razloga zašto postoje i liste i torke je što su nam u nekim situacijama potrebne nepromenljive liste. Na primer:
  - lista ne može da bude ključ u rečniku (lista se može menjati, problem je kako pratiti promene ključa)
  - torke mogu da igraju ulogu ključa u rečnicima.

```
ocene = {('Petar', 'Ana'): 95, ('Milovan', 'Igor'): 87}
```

Python
Torke (Tuples)

- Brže od listi
- Za konverziju nekog objekta u torku koristimo funkciju `tuple`.

```
t1 = tuple([1,2,3])     (1, 2, 3)
t2 = tuple('abcde’)     ('a', 'b', 'c', 'd', 'e')
```

- Prazna torka je `()`.
- Torka sa jednim elementom `(1,)`.
- `(1)` matematički izraz koji ima vrednost 1.

- Torke se koriste za vraćanje više vrednosti iz funkcije
- Torke mogu biti elementi liste

Python
Torke (Tuples)

- Torke mogu biti elementi liste

```
meseci = [("јануар", 31), ("фебруар", 28), ("март", 31), \
          ("април", 30), ("мај", 31), ("јун", 30), \
          ("јул", 31), ("август", 31), ("септембар", 30), \
          ("октобар", 31), ("новембар", 30), ("децембар", 31)]
broj = int(input("Унеси редни број месеца:"))
mesec = meseci[broj - 1]
print("Назив:", mesec[0], "Број дана:", mesec[1])
```

Python
Skupovi (Sets)

- Python ima i strukturu koja se zove `set (skup)`.
- Set struktura radi slično matematičkim skupovima.
- Skupovi su poput listi u kojima nije dozvoljeno ponavljanje elemenata.
- Skupovi se označavaju vitičastim zagradama, `{}`

```
S = {1,2,3,4,5}
```

- Prazan skup dobijamo pomoću funkcije `set()`

```
S = set()
```

Python
Skupovi (Sets)

- Funkcija set se može koristiti i za konverziju objekata u skupove.

```
set([1,4,4,4,5,1,2,1,3])
set('this is a test')
{1, 2, 3, 4, 5}
{'a', ' ', 'e', 'i', 'h', 's', 't’}
```

- Skupove možemo sastavljati na različite načine:

```
s = {i**2 for i in range(12)}
{0, 1, 4, 100, 81, 64, 9, 16, 49, 121, 25, 36}
```

Python
Skupovi (Sets)

- Nad skupovima je moguće primeniti relacijske operatore:
  - == (jednakost)
  - <= (podskup)
  - < (pravi podskup)
  - >= (nadskup)
  - > (pravi nadskup)

Python
Skupovi (Sets)

- Python smešta podatke u skup u proizvoljnom redosledu, i ne uvek u redosledu koji ste vi postavili.
- Kod skupova nije važan redosled, već samo podaci koji ga čine. To znači da indeksiranje kod skupova nema smisla. Zato se ne može , na primer, tražiti s[0], za set s.
- Operacije za rad sa skupovima

Operator	Opis	Primer
|	Unija	{1,2,3} | {3,4}→{1,2,3,4}
&	Presek	{1,2,3}&{3,4}→{3}
-	Razlika	{1,2,3}-{3,4}→{1,2}
^	Simetrična razlika	{1,2,3}^{3,4}→{1,2,4}
in	Jeste u skupu	3 in {1,2,3}→True

Python
Skupovi (Sets)

- Metode koje se primenjuju na skupove

Operator	Opis
S.add(x)	Dodaje x u skup
S.remove(x)	Sklanja x iz skupa
S.issubset(A)	Vraća True ako je $S \subset A$, a False ako nije
S.issuperset(A)	Vraća True ako je $A \subset S$, a False ako nije.

Python
Skupovi (Sets)

```
>>> {'crvena', 'plava'} == {'plava', 'crvena'}
True
>>> a, b, c = {1, 2, 3, 4, 5}, {3, 4, 5}, {1, 2}
>>> c.add(3) # metod za ubacivanje elementa u skup
>>> c
{1, 2, 3}
>>> c.discard(3) # uklanja element ako postoji
>>> c
{1, 2}
>>> a & b # presek skupova
{3, 4, 5}
>>> b & c # presek je prazan
set()
```

Python
Skupovi (Sets)

```
>>> b | c # unija skupova
{1, 2, 3, 4, 5}
>>> a - b # razlika skupova
{1, 2}
>>> a >= b # a nadskup b
True
>>> b <= a # b podskup a
True
>>> a.clear() # uklanja sve elemente iz skupa
>>> a
set()
```

Python
Skupovi (Sets)

- Brisanje duplih elemenata iz liste
- Možemo iskoristiti činjenicu da skupovi ne mogu imati ponovljene elemente.

```
L = [1,4,4,4,5,1,2,1,3]    # lista sa ponavljanjem elemenata
L = list(set(L))           # konvertujemo listu u skup, pa nazad u listu
```

```
[1,2,3,4,5]
```

Python
Skupovi (Sets)

- Napisati program u kojem se unosi broj n a zatim n celih brojeva. Za sve cifre koje su se javile u zapisu unetih brojeva odrediti koliko puta su se ukupno javile.

```
n = int(input())
brojevi = []
for i in range(n):
    brojevi = brojevi + [int(input())]
print(brojevi)

lista_cifara = []
for x in brojevi:
    while x > 0:
        lista_cifara = lista_cifara + [x % 10]
        x = x // 10
print(lista_cifara)

skup_cifara = set(lista_cifara)
for x in skup_cifara:
    print(x," : ", lista_cifara.count(x))
```

```
3
121
23568
5
[121, 23568, 5]
[1, 2, 1, 8, 6, 5, 3, 2,
5]
1  :  2
2  :  2
3  :  1
5  :  2
6  :  1
8  :  1
>>>
```