OSNOVE PROGRAMIRANJA
PYTHON

Profesor
Milan Paroški

Novi Sad, 2024/2025

Poglavlje 8

Polja i neuređene liste u jeziku Python

Sadržaj

1. Uvod
2. Polja
3. Neuređene liste
4. Primeri programa

SORT

- sort(key=None, reverse=False)
- Sortira elemente liste u zadanom redosledu, definisanom argumentima key i reverse

```
brojevi = [33, 21, 5, 15, -3, 0]
brojevi.sort()
print(brojevi)
```

[-3, 0, 5, 15, 21, 33]

```
slova = ['A','a', 'b','E']
slova.sort()
print(slova)
```

['A', 'E', 'a', 'b']

Primer sortiranja po dužini reči?

```
words = ["pera", "Ana", "Zoran", "jovica"]
words.sort(key=len)
print("Sortirano po dužini:", words)
```

Sortirano po dužini: ['Ana', 'pera', 'Zoran', 'jovica']

Primer sortiranja po obrnutoj dužini reči?

```
words = ["pera", "Ana", "Zoran", "jovica"]
words.sort(key=len,reverse=True)
print("Sortirano po dužini:", words)
```

Sortirano po dužini: ['jovica', 'Zoran', 'pera', 'Ana']

Sortiranje liste brojeva po apsolutnim vrednostima?

```
numbers = [3, -1, -4, 2, 0, -2]
sorted_numbers = sorted(numbers, key=abs)
print("Sortirano po apsolutnoj vrednosti:", sorted_numbers)
```

Sortirano po apsolutnoj vrednosti: [0, -1, 2, -2, 3, -4]

Sortiranje liste brojeva po obrnutim apsolutnim vrednostima?

```
numbers = [3, -1, -4, 2, 0, -2]
sorted_numbers = sorted(numbers, key=abs,reverse=True)
print("Sortirano po apsolutnoj vrednosti:", sorted_numbers)
```

Sortirano po apsolutnoj vrednosti: [-4, 3, 2, -2, -1, 0]

Sortiranje stringova ignorisanjem veličine slova?

```
words = ["Banana", "apple", "Cherry", "date"]
words.sort(key=str.lower)
print("Sortirano ignorisanjem veličine slova:", words)
```

Sortirano ignorisanjem veličine slova: ['apple', 'Banana', 'Cherry', 'date']

8.1 Uvod

Struktura podataka je ?

kolekcija više delova podataka strukturiranih na takav način da im se može efikasno pristupati.

Linearne strukture podataka u jeziku Python su nizovi:

1. stringovi,
2. jednostavne liste i
3. n-torke.

1. **String** je niz karaktera:

Primer:

Rec = "Python 3.11!"
print(Rec)
Python 3.11!

2. ***Lista*** predstavlja listu elemenata, odvojenih zarezima, a okruženu uglastim zagradama. Primer:

Lista = [1, 2, 3]

print('Lista=', Lista, 'je dužine', len(Lista))

Lista=[1, 2, 3] je dužine 3

3. **N-torka** (tuple) je tip podatka vrlo sličan listi. N-torka je nepromenljiva :

- nema dodavanje sadržaja u n-torku,
- nema brisanja sadržaja iz n-torke
- nema zamena elemenata unutar nje

ntorka = (1, 2, 3)

```
t1 = tuple([1,2,3])  # konverzija liste u ntorku
print(t1)
(1, 2, 3)
```

```
t2 = tuple("abcde")  # konverzija stringa u ntorku
print(t2)
('a', 'b', 'c', 'd', 'e')
```

```
ntorka = (1,2,3,4,5,6)
lista = [1,2,3,4,5,6]
print("ntorka",ntorka)
print("lista",lista)
print(len(ntorka))
print(len(lista))
print("stampa se nulti element")
print(ntorka[0])

print("stampa se od drugog do 4tog elementa")
print(ntorka[2:4])

print("stampa se sa pozicije 5")
print(ntorka.index(5))
print(lista.index(5))
```

Vrednost **5** nalazi se na **četvrtoj** poziciji, što znači da je njen indeks **4**.

```
ntorka (1, 2, 3, 4, 5, 6)
lista [1, 2, 3, 4, 5, 6]
6
6
stampa se nulti element
1

stampa se od drugog do 4tog elementa
(3, 4)

stampa se sa pozicije 5
4
4
```

```
ntorka = (1,2,3,4,5,6)

print("stampa se najmanji element")
print(min(ntorka))

print("stampa se najveci element")
print(max(ntorka))

print("stampa se suma svih elemenata")
print(sum(ntorka))


print("da li postoji elemenat 6 u ntorki")
print(6 in ntorka)
```

```
stampa se najmanji element
1
stampa se najveci element
6

stampa se suma svih elemenata
21

da li postoji elemenat 6 u ntorki
True
```

```
lista = [11,2,3,4,5,6]

lista.append(7)                    [11, 2, 3, 4, 5, 6, 7]
print(lista)

lista.insert(0,77)                 [77, 11, 2, 3, 4, 5, 6, 7]
print(lista)

lista.remove(11)                   [77, 2, 3, 4, 5, 6, 7]
print(lista)

lista.clear()                      []
print(lista)
```

N-torke (tuples) su tip sekvenci, poput stringova. Ali za razliku od stringova, koji mogu sadržavati samo znake, Ntorke mogu sadržavati elemente bilo kojih tipova.

U Ntorke se mogu smestiti stringovi, brojevi.... Bilo šta što se može dodeliti promenjivoj, može se grupisati i smestiti kao sekvenca u ntorci.

Nelinearne strukture podataka u jeziku Python su :

1. Ugnježdene liste – *predmet ovog poglavlja*
2. Neuređene liste,
3. Rečnici (dictionaries) – *predmet ovog poglavlja*

1.Ugnježdene liste imaju elemente koji su sami liste

Oceneizpredmeta= [[1,3,4,5],[5,5,4,2]]

print(*Oceneizpredmeta)*

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

2. Neuredjene liste

```
#Prikaz brojeva iz neuredjene liste
for broj in [10, 5, -3]:
    print(broj)
```

```
10
5
-3
```

```
#Prikaz brojeva iz uredjene liste
for broj in [1,2,3]:
    print(broj)
```

```
1
2
3
```

3. Rečnik

Rečnik je neuređena kolekcija stvari ili elementa.

Svaka stavka ima ključ i određenu vrednost.

To je izraženo kao par.

Dok vrednosti mogu biti bilo kog tipa podataka i mogu se ponavljati, ključevi moraju biti nepromenljivog tipa

```
auto = {'boja' : 'bela', 'ccm' : '999','KW' : 55}
print(auto)
```

```
{'boja': 'bela', 'ccm': '999', 'KW': 55}
```

Ugnježdene liste i rečnici su rekurzivne strukture, jer se kao element structure javljaju opet one same.

Rekurzivne strukture podataka omogućavaju predstavljanje i rad s podacima povezanim rekurzivnim relacijama, npr. :

- Grafovima (vektorska grafika, Kohova kriva, Zmajeva kriva)
- stablima (fraktalno stablo…)
- Matricama (danas)

Liste su strukture kod kojih elementi imaju numeričku poziciju, odnosno elementu liste se pristupa pomoću indeksa koji predstavlja broj.

Lista = [11, 12, 13]

print(Lista[0]) 11

Rečnici su strukture slične listama, kod kojih je indeks opštijeg tipa, a sama struktura nema linearno uređenje.

8.2 Polja

Polja ili **matrice su :**

- **višedimenzionalne strukture podataka,**
- **istog tipa,**
- **čijim elementima se pristupa preko skupa indeksa,**
- **po jedan indeks za svaku dimenziju.**

Jezik Python nema posebnu strukturu polja ili matrice, već se za prikaz višedimenzionalnih polja ili matrica **koriste ugnježdene liste**

8.2.1 Ugnježdene liste

Ugnježdene liste imaju elemente koji su sami liste.

Primer lista je: [10,20,25,20]

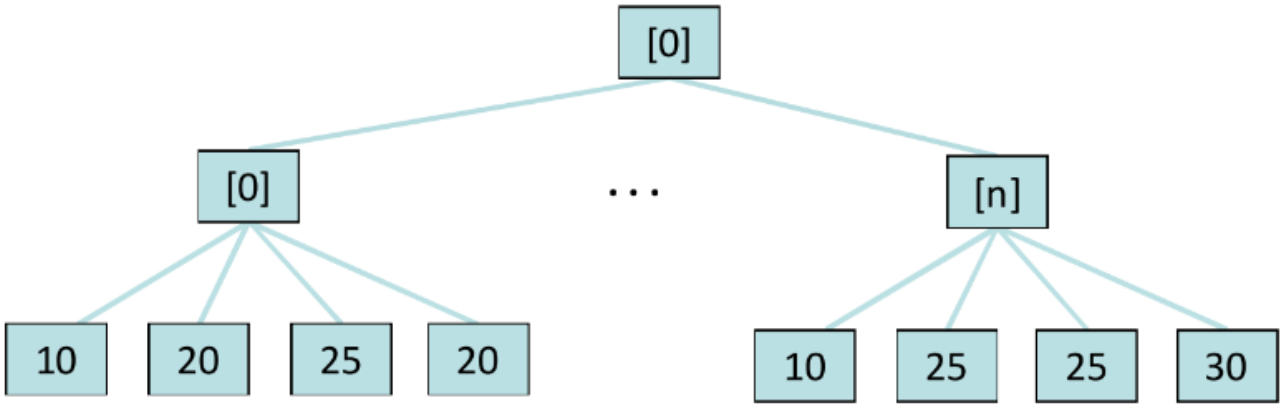
Ugnježdene liste se mogu upotrebiti npr. za smeštanje sledećih podataka o poenima ostvarenim na ispitu iz nekog predmeta:

aktivnost, kolokvijum 1, kolokvijum 2 i završni ispit:

uspeh_na_ispitu/ima= [[10,20,25,20],..., [10,25,25,30]]

Ovako definisana struktura podataka može se posmatrati kao plitka hijerarhija, u dva nivoa:

Ovo mogu biti ocene iz jednog predmeta za više studenata ili?



8.2.2 Predstavljanje matrica

Dvodimenzionalne matrice dimenzija m x n mogu se predstaviti kao ugnježdene liste s dva nivoa indeksa, npr.

>>> matrica = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Pristup elementima matrice vrši se navođenjem indeksa

Od kog broja kreće brojanje redova i kolona?

Šta će biti rezultat upita: matrica [1][2] ?

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

a=mat [1][2]
print(a)

```
          0   1  2
mat = [[1, 2, 3],      0
       [4, 5, 6],      1
       [7, 8, 9]]      2

a=mat [0][1]
2
a=mat [2][3]
IndexError: list index out of range
```

Kvadratna matrica 5x5 može se zadati sledećom listom:

```
matrica = [[1, 2, 3, 4, 5],
[6, 7, 0, 0, 0],
[0, 1, 0, 0, 0],
[1, 0, 0, 0, 8],
[0, 0, 9, 0, 3]]

a=matrica [0][4]
print(a)
```

5

Šta je rezultat ovog upita: *a=matrica [1]*

[6, 7, 0, 0, 0]

Kako se može ispisati prva kolona matrice?

A bez korišćenja biblioteke NumPy?

```
matrica = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]
kolona = [red[1] for red in matrica]
print("Druga kolona:", kolona)
```

Druga kolona: [2, 5, 8]

8.2.3 Suma svih elemenata matrice

Šta je zbir matrice?

Zbir elemenata matrice dobija se sumiranjem svih elemenata po redovima i kolonama:

```
matrica = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
suma = 0
for red in matrica:
    for element in red:
        suma = suma + element
        print("Red=",red,"  Element=",element,"  Suma=",suma)
print("Zbir elemenata matrice=", suma)
```

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

Zbir elemenata matrice= 45

Red= [1, 2, 3] Element= 1 Suma= 1
Red= [1, 2, 3] Element= 2 Suma= 3
Red= [1, 2, 3] Element= 3 Suma= 6
Red= [4, 5, 6] Element= 4 Suma= 10
Red= [4, 5, 6] Element= 5 Suma= 15
Red= [4, 5, 6] Element= 6 Suma= 21
Red= [7, 8, 9] Element= 7 Suma= 28
Red= [7, 8, 9] Element= 8 Suma= 36
Red= [7, 8, 9] Element= 9 Suma= 45
Zbir elemenata matrice= 45

KAKO NACI NAJMANJI I NAJVECI ELEMENT MATRICE:

```
matrica = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
minimum = matrica[0][0]
maksimum = matrica[0][0]
print("Minimum je:",minimum)
print("Maksimum je:",maksimum)
for red in matrica:
    for element in red:
        if element < minimum:
            minimum = element
            print(minimum)
        if element > maksimum:
            maksimum = element
print("Najmanji element matrice=", minimum)
print("Najveci element matrice=", maksimum)
```

Minimum je: 1
Maksimum je: 1
Najmanji element matrice= 1
Najveci element matrice= 9

Broj redova matrice?

matA = [[1, 2], [4, 5], [7, 8], [4, 5]]

print(len(matA))

4

Broj kolona matrice?

print(len(matA[0]))

2

print(len(matA[2]))?

Zasto je formula za kolone bas takva:

matA = [[1, 2], [4, 5], [7, 8], [4, 5]]
print(matA[0])
Rezultat:[1, 2]
a
print(len([1, 2]))
je : 2

```
matA = [[1, 2], [4, 5], [7, 8], [4, 5]]
matB = [[1, 2, 3], [4, 5, 6]]

print("Broj redova A=", len(matA))
print("Broj kolona A=", len(matA[0]))

print("Broj redova B=", len(matB))
print("Broj kolona B=", len(matB[0]))
```

Broj redova A= 4
Broj kolona A= 2
Broj redova B= 2
Broj kolona B= 3

Transponovanje matrica?

X = [[12,7],
[4 ,5],
[3 ,8]]

[12, 4, 3]
[7, 5, 8]

```
# Program za transponovanje matrice
X = [[12,7],
    [4 ,5],
    [3 ,8]]
result = [[0,0,0],
          [0,0,0]]
# petlja kroz redove
for i in range(len(X)):  #Šta ovo znači
   # petlja kroz kolone
   for j in range(len(X[0])):  #Šta ovo znači?
       result[j][i] = X[i][j]
        print("i=",i,"j=",j,"Rezultat(",j,",",i,")=",result[j][i])
for r in result:
   print(r)
```

[12, 4, 3]
[7, 5, 8]

2.čas

i= 0 j= 0 Rezultat(0 , 0)= 12

i= 0 j= 1 Rezultat(1 , 0)= 7

i= 1 j= 0 Rezultat(0 , 1)= 4

i= 1 j= 1 Rezultat(1 , 1)= 5

i= 2 j= 0 Rezultat(0 , 2)= 3

i= 2 j= 1 Rezultat(1 , 2)= 8

[12, 4, 3]

[7, 5, 8]

Sabiranje matrica

X = [[12,7,3],
[4 ,5,6],
[7 ,8,9]]

Y = [[5,8,1],
[6,7,3],
[4,5,9]]

Rezultat?:

[17, 15, 4]
[10, 12, 9]
[11, 13, 18]

```
# Program za sabiranje 2 matrice
X = [[12,7,3],    [4 ,5,6],    [7 ,8,9]]
Y = [[5,8,1],    [6,7,3],    [4,5,9]]

result = [[0,0,0],  [0,0,0],  [0,0,0]]

# petlja kroz redove
for i in range(len(X)):
   # petlja kroz kolone
   for j in range(len(X[0])):
       result[i][j] = X[i][j] + Y[i][j]

for r in result:
   print(r)
```

```
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
```

8.2.4 Množenje matrica

Množenje dve matrice M = AB je operacija koja je definisana samo kada?

Kad je broj redova jedne matrice jednak broju kolona druge, odnosno množenje matrica dimenzija m×n i n×p daje matricu dimenzija m×p

Množenje matrica dimenzija 4×2 i 2×3 daje matricu dimenzija 4×3:

```
matA = [[1, 2], [4, 5], [7, 8], [4, 5]]
matB = [[1, 2, 3], [4, 5, 6]]
matC = [[0,0,0],[0,0,0],[0,0,0],[0,0,0]]
for i in range(len(matA)):              # petlja po redovima A
    for j in range(len(matB[0])):       # petlja po kolonama B
        for k in range(len(matB)):      # petlja po redovima B
            matC[i][j]=matC[i][j] + matA[i][k]*matB[k][j]
print(matC)
```

```
[[9, 12, 15], [24, 33, 42], [39, 54, 69], [24, 33, 42]]
```

```
matA = [[1, 2],
4x2     [4, 5],
        [7, 8],
        [4, 5]]
```

```
matB = [[1, 2, 3],
2x3     [4, 5, 6]]
```

Koliko su a,b,c?

a=1x1 + 2x4=9

b=1x2 + 2x5=12

c=1x3 + 2x6=15

```
matC = [[a,b,c],
4x3     [0,0,0],
        [0,0,0],
        [0,0,0]]
```

Postupno prikazivanje

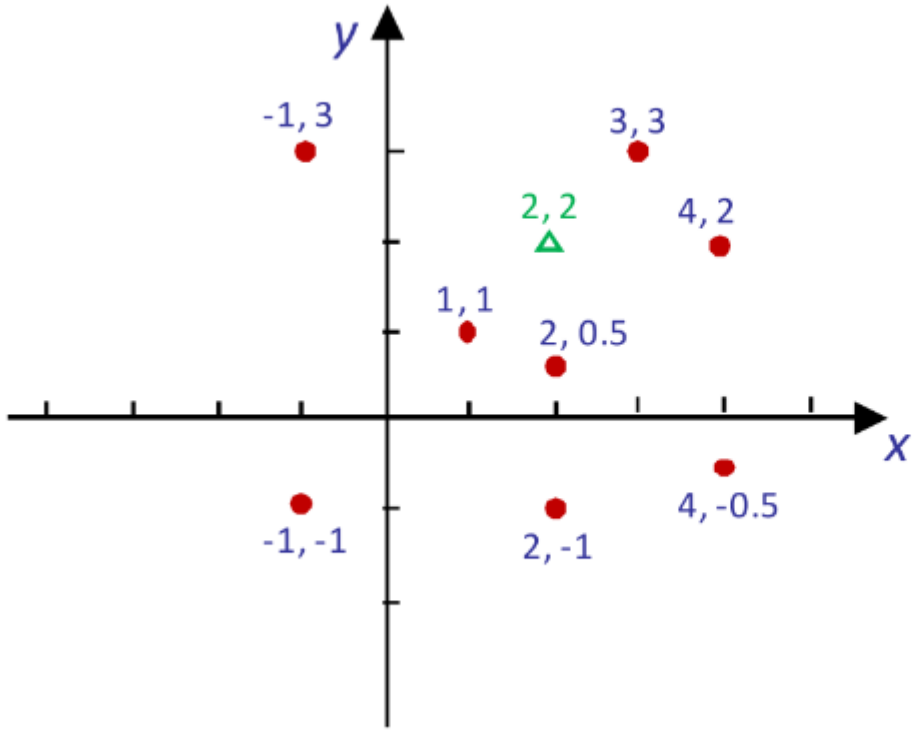
```
matA = [[1, 2], [4, 5], [7, 8], [4, 5]]
#print(len(matA))
matB = [[1, 2, 3], [4, 5, 6]]
#print(len(matB[0]))
matC = [[0,0,0],[0,0,0],[0,0,0],[0,0,0]]
for i in range(len(matA)): # petlja po redovima A
    for j in range(len(matB[0])):# petlja po kolonama B
        for k in range(len(matB)): # petlja po redovima B
         matC[i][j]=matC[i][j] + matA[i][k]*matB[k][j]
         print("i=",i,"j=",j,"k=",k,"   C=",matC[i][j])

print(matC)
```

i= 0 j= 0 k= 0 C= 1
i= 0 j= 0 k= 1 C= 9
i= 0 j= 1 k= 0 C= 2
i= 0 j= 1 k= 1 C= 12
i= 0 j= 2 k= 0 C= 3
i= 0 j= 2 k= 1 C= 15
i= 1 j= 0 k= 0 C= 4
i= 1 j= 0 k= 1 C= 24
i= 1 j= 1 k= 0 C= 8
i= 1 j= 1 k= 1 C= 33
i= 1 j= 2 k= 0 C= 12
i= 1 j= 2 k= 1 C= 42
i= 2 j= 0 k= 0 C= 7
i= 2 j= 0 k= 1 C= 39
i= 2 j= 1 k= 0 C= 14
i= 2 j= 1 k= 1 C= 54
i= 2 j= 2 k= 0 C= 21
i= 2 j= 2 k= 1 C= 69
i= 3 j= 0 k= 0 C= 4
i= 3 j= 0 k= 1 C= 24
i= 3 j= 1 k= 0 C= 8
i= 3 j= 1 k= 1 C= 33
i= 3 j= 2 k= 0 C= 12
i= 3 j= 2 k= 1 C= 42

[[9, 12, 15], [24, 33, 42], [39, 54, 69], [24, 33, 42]]

Primer: Najbliža tačka u ravni

Matrica n×2 može se upotrebiti za prikaz rasporeda n tačaka u ravni.
Elementi u kolonama matrice su koordinate tačaka x i y.



tačka	x	y
0	-1	3
1	-1	-1
2	1	1
3	2	0.5
4	2	-1
5	3	3
6	4	2
7	4	-0.5

Pronalaženje tačke koja je prostorno najbliža zadanoj tački može se izvršiti linearnim pretraživanjem tabele (matrice) koordinata svih tačaka:

1. Postavi se da je prva tačka najbliža (-1,3)

2. Za tu tačku i sve preostale tačke:

na svakom koraku se računa udaljenost od tačke b do zadane tačke

$$d_{ab} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

3. ako je udaljenost manja od do tada najbliže tačke, zapamte se koordinate x,y i udaljenost d nove najbliže tačke.

1. Tačke u ravni mogu se predstaviti ?

matricom

Koje dimenzije

8 x 2, koja se u jeziku Python realizuje ugnježdenom listom.

tacke = [[-1, 3], [-1, -1], [1,1], [2, 0.5],[2, -1], [3, 3], [4, 2], [4, -0.5]]

zadana = [2, 2]

2. Postavi se da je prva tačka najbliža (-1,3) i izračuna udaljenost:

najbliza = tacke[0]

mind = euklid(zadana, najbliza)

3. Pošto se često koristi, računanje euklidskog rastojanja dobro je realizovati pomoću posebne funkcije.

```
def euklid(x,y):
```

4. Za računanje kvadratnog korena u ovoj funkciji pogodno je upotrebiti namensku funkciju sqrt (square root) iz modula math,

```
return math.sqrt( (x[0]-y[0])**2+(x[1]-y[1])**2 )
```

kome se pristupa naredbom import

```
import math
```

1.deo

```
import math
tacke = [[-1, 3], [-1, -1], [1,1], [2, 0.5],[2, -1], [3, 3], [4, 2], [4, -0.5]]
zadana  = [2, 2]
def euklid(x,y):
    return math.sqrt( (x[0]-y[0])**2+(x[1]-y[1])**2 )

najbliza = tacke[0]
mind = euklid(zadana, najbliza)

print("Najbliža je na rastojanju od", mind)
```

Najbliža je na rastojanju od 3.162277601683795

5.Pretraživanje je pogodno realizovati ?

for petljom u kojoj se, počev od druge tačke, računa euklidska udaljenost tačaka (druge i zadane):

```
for i in range(1,len(tacke)):
    d = euklid(zadana, tacke[i])
```

a zatim poredi s do tada ustanovljenom najmanjom udaljenošću i zapamti ako je udaljenost najmanja

```
if d < mind:
        najbliza = tacke[i]
        mind = d
```

6. Štampanje rezultata:

```
print("Najbliza tačka tački", zadana, \
"je", najbliza,"udaljena", mind)
```

7. Rezultat:

```
Najbliza tačka tački [2, 2] je [1, 1] udaljena
   1.4142135623730951
```

```
import math
tacke = [[-1, 3], [-1, -1], [1,1], [2, 0.5],[2, -1], [3, 3], [4, 2], [4, -0.5]]
zadana  = [2, 2]
def euklid(x,y):
    return math.sqrt( (x[0]-y[0])**2+(x[1]-y[1])**2 )
najbliza = tacke[0]
mind = euklid(zadana, najbliza)
for i in range(1,len(tacke)):
    d = euklid(zadana, tacke[i])
    if d < mind:
        najbliza = tacke[i]
        mind = d
print("Najbliza tačka tački", zadana, \
"je", najbliza,"udaljena", mind)
```

Boldovano smo već uradili i probali

Najbliza tačka tački [2, 2] je [1, 1] udaljena 1.414213562373095

```
import math
def euklid(x,y):
    print(x[0])
    print(y[0])
    print(x[1])
    print(y[1])
    print("dxd=",( (x[0]-y[0])**2+(x[1]-y[1])**2 ))
    return math.sqrt( (x[0]-y[0])**2+(x[1]-y[1])**2 )
tacke = [[-1, 3], [-1, -1], [1,1], [2, 0.5],[2, -1], [3, 3], [4, 2], [4, -0.5]]
zadana  = [2, 2]
najbliza = tacke[0]   # prvi red matrice
print(najbliza)
print(zadana)
mind = euklid(zadana, najbliza)
for i in range(1,len(tacke)):
    d = euklid(zadana, tacke[i])
    if d < mind:
        najbliza = tacke[i]
        mind = d
print("Najbliza tačka tački", zadana, \
"je", najbliza,"udaljena", mind)
```

Za prve tri tacke:

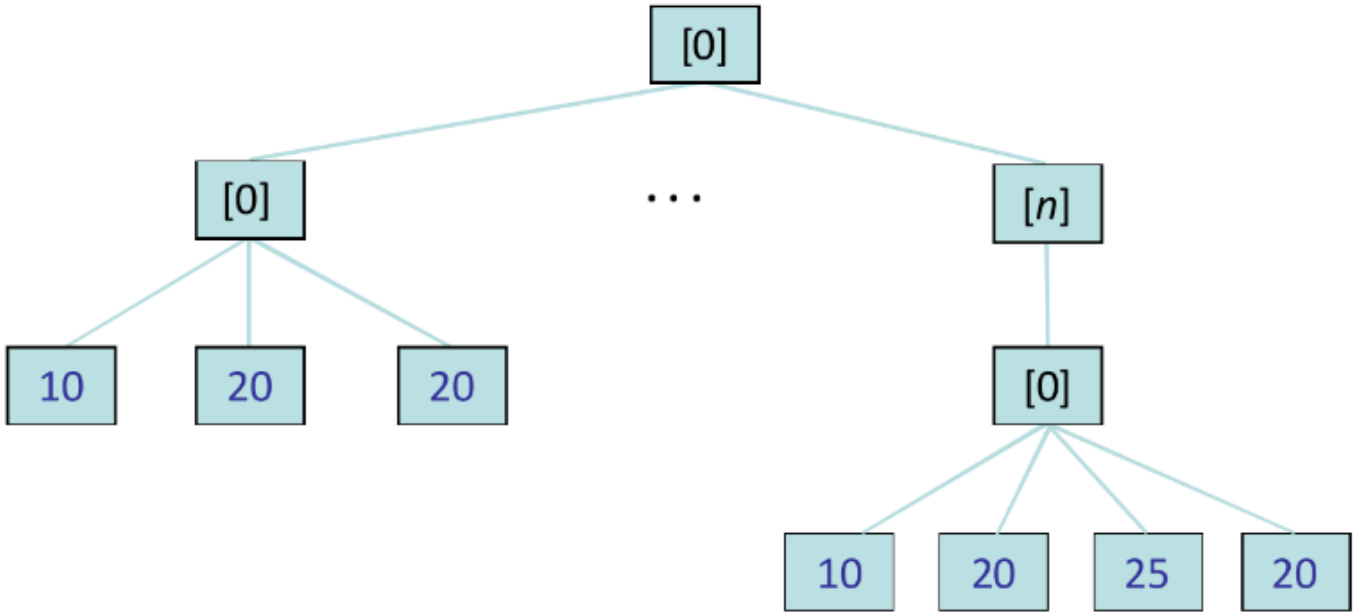
```
[-1, 3]
[2, 2]
2
-1
2
3
dxd= 10
2
-1
2
-1
dxd= 18
2
1
2
1
dxd= 2
```

8.2.5 Predstavljanje hijerarhija (stabala)

Hijerarhije su rekurzivne strukture, koje se mogu predstaviti ugnježdenim listama s različitim brojem elemenata na svakom nivou, npr. sledeća lista

[[10,20,20] ... [[10,20,25,20]]]

može se grafički predstaviti hijerarhijom prikazanom na slici:

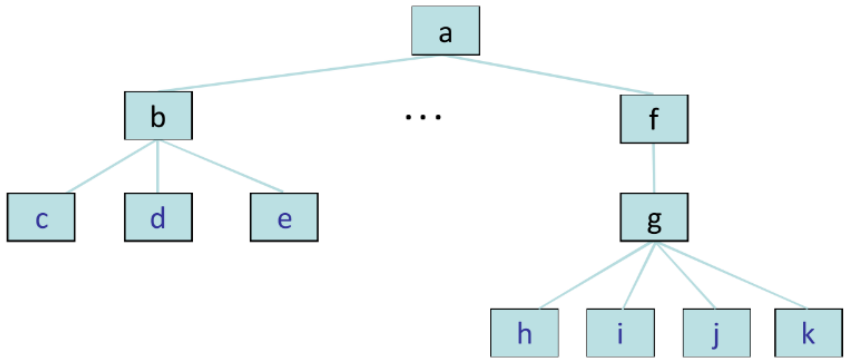


Struktura stabla se može predstaviti na različite načine, npr. tako da svaki čvor bude predstavljen listom, koja sadrži sam čvor i listu njegovih sledbenika:

[<čvor>, [<sledbenik1>, ... ,<sledbenikn>]]

Sledbenik može biti podatak ili ugnježdena lista.

Na ovaj način se npr. sledećom strukturom podataka može da predstaviti opšte stablo, s proizvoljnim brojem grana u čvorovima, prikazano na slici:



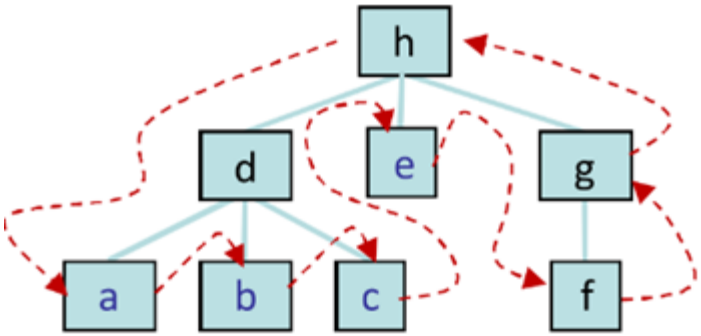
['a',
['b', ['c', 'd', 'e']],
['f' ['g' ['h' , 'i', 'j' , 'k']]]]

8.2.6 Obilazak stabla

Radi izvršenja određenih operacija, struktura opšteg stabla može se obilaziti na dva načina:

(a) u dubinu (depth-first), tako da se prvo se obiđu grane, a zatim koren stabla,

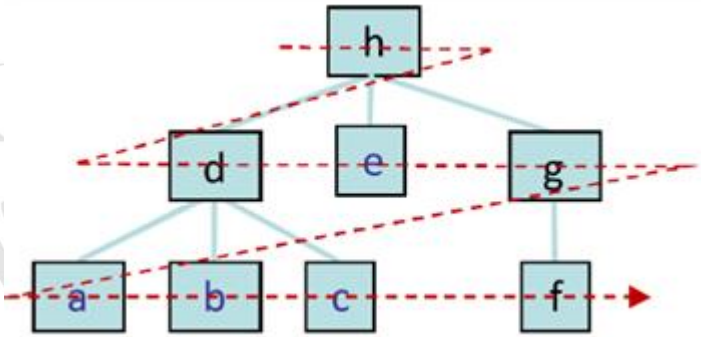
Koji je redosled za a)?



Redosled: a,b,c,d,e,f,g,h

(b) u širinu (breadth-first), tako što se prvo se obiđu svi čvorovi istog nivoa/dubine, a zatim se obilaze čvorovi na sledećem nivou.

Koji je redosled za b)?



redosled: h, d,e,g, a,b,c, f

3.čas

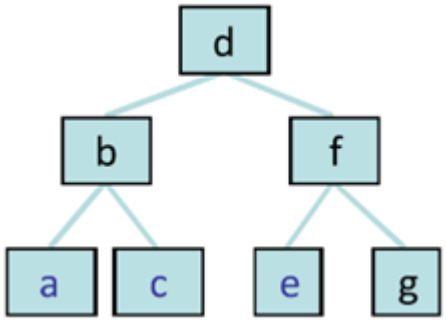
8.2.7 Binarna stabla

Binarna stabla imaju najviše ?

dve grane svakog čvora stabla.

Za binarna stabla, obilazak u dubinu može se realizovati na više načina, u odnosu na redosled operacija koje se izvršavaju na svakom čvoru može biti, sledeća slika:

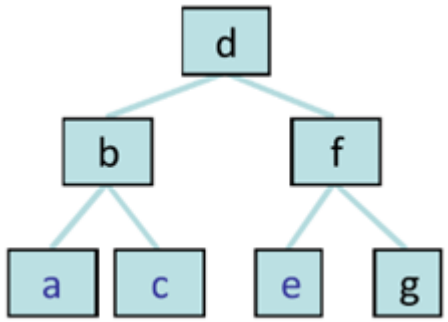
- **preorder**: prvo se obiđe čvor, pa zatim oba njegova podstabla,



Koji je redosled ?

preorder: d, b, a, c, f, e, g

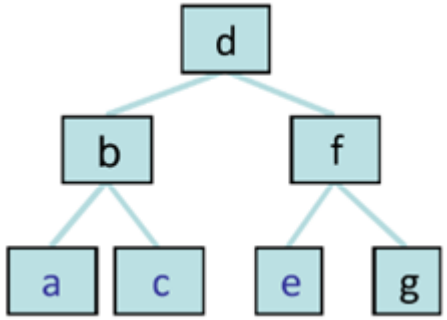
- **postorder**: prvo obiđu oba podstabla nekog čvora, a zatim sam čvor,



Koji je redosled ?

postorder: a, c, b, e, g, f, d

•**inorder,** po unutrašnjem uređenju: prvo se obiđe levo podstablo, zatim čvor, pa desno podstablo.



Koji je redosled ?

inorder: a, b, c,d, e, f, g

8.3 Neuređene liste: rečnici i skupovi

Neuređene liste ili strukture rečnik i skup su takve strukture podataka kod kojih **redosled elemenata nije od značaja**, jer im se **pristupa na osnovu nenumeričkog indeksa** (ključa) ili samo pomoću njihove vrednosti, koja je tada ključ za pristup.

8.3.1 Struktura rečnik (dictionary)

Sintaksa za definisanje strukture rečnika je:

{<ključ>:<vrednost>, ..., <ključ>:<vrednost>}

Ključ je podatak na osnovu kojeg se pristupa vrednosti elementa u rečniku.

Vrednost elementa može biti bilo kog tipa, uključujući sam rečnik, tako da se pomoću rečnika može predstaviti hijerarhijska struktura podataka.

Pristup podacima pomoću zadanog ključa može se prikazati na primeru podataka o nekretninama, gde je pojedinačna nekretnina prikazana rečnikom, na primer:

```
kuca = {'boja' : 'bež', 'stil' : 'pedesetih',
'brojSoba' : 4,'imaGarazu' : True, 'imaAlarm': False,
'kucniBroj' : 123,'ulica' : 'Trešnjina', 'grad' :
'Beograd', 'cena' : 125000}
```

Pristup određenom podatku o nekretnini iz prethodne strukture vrši se zadavanjem vrednosti njegovog ključa, npr.

```
print(kuca['cena'])
```

125000

8.3.2 Primena operatora 'in' na strukturu rečnika

Neka su podaci o broju stanovnika nekoliko gradova u Srbiji prikazani u sledećoj strukturi rečnika:

gradovi = {'Beograd':1659440, 'Novi Sad':341625,

'Niš':373404, 'Kragujevac':179417}

Pošto rečnik definiše skup elemenata, za obilazak rečnika može se koristiti petlja for:

```
for grad in gradovi:
    print(grad)
```

```
for a in kuca:
    print(a)
```

Beograd
Novi Sad
Niš
Kragujevac

Za prethodni primer:

for a in kuca:

print(a)

boja
stil
brojSoba
imaGarazu
imaAlarm
kucniBroj
ulica
grad
cena

Prethodna petlja prikazuje samo spisak gradova (ključeva).

Prikaz ostalih podataka može se dobiti upotrebom ključa kao indeksa, na primer:

```
for grad in gradovi:
    print(grad, gradovi[grad])
```

Novi Sad 341625
Niš 373404
Beograd 1659440
Kragujevac 179417

```
kuca = {'boja' : 'bež', 'stil' : 'pedesetih',
'brojSoba' : 4,'imaGarazu' : True, 'imaAlarm': False,
'kucniBroj' : 123,'ulica' : 'Trešnjina', 'grad' :
'Beograd', 'cena' : 125000}
```

```
for a in kuca:
    print(a,kuca[a])
```

```
boja bež
stil pedesetih
brojSoba 4
imaGarazu True
imaAlarm False
kucniBroj 123
ulica Trešnjina
grad Beograd
cena 125000
```

8.3.3 Kombinovanje struktura rečnika i uređene liste

Rečnici su generalizovane liste, a obe strukture mogu da sadrže kao elemente druge strukture, tako da omogućavaju hijerarhijsku organizaciju podataka.

Strukture podataka rečnika i liste mogu se međusobno kombinovati, zavisno od vrste podataka i operacija koje su potrebne na različitim nivoima hijerarhije, na primer:

- struktura rečnika koja ima vrednosti elemenata koji su liste podataka

{<ključ>:[<vrednost>,...], ... } - primer 1

- ugnježdene liste koje predstavljaju strukturu stabla koje ima čvorove u obliku rečnika

[{<ključ>:<vrednost>},[{<ključ>:<vrednost>},...]] - primer 2:

Pre je bilo:{<ključ>:<vrednost>, ..., <ključ>:<vrednost>}

primer 1 :

Dodate 3 liste, kao da ima 2 adrese:

- 2 podatka za sobnost : 4 i 5
- 2 podatka za ulicu
- 2 podatka za kucni broj

```
kuca = {'boja' : 'bež', 'stil' : 'pedesetih',
'brojSoba' : [4,5],'imaGarazu' : True, 'imaAlarm': False,
'kucniBroj' : [12,123],'ulica' : ['Trešnjina','Radnicka'], 'grad' :
'Beograd', 'cena' : 125000}

for a in kuca:
    print(a,kuca[a])
```

boja bež
stil pedesetih
→ brojSoba [4, 5]
imaGarazu True
imaAlarm False
→ kucniBroj [12, 123]
→ ulica ['Trešnjina', 'Radnicka']
grad Beograd
cena 125000

8.3.4 Spisak promenljivih u jeziku Python

Promenljive programa u jeziku Python čuvaju se u strukturi tipa rečnika.

Prilikom pokretanja interpretera, postoji unapred definisan skup sistemskih promenljivih, koje se mogu prikazati pomoću ugrađene funkcije vars():

```
print(vars())
{'__name__': '__main__', '__doc__': None, '__package__': None,
    '__loader__': <class '_frozen_importlib.BuiltinImporter'>,
    '__spec__': None, '__annotations__': {}, '__builtins__':
    <module 'builtins' (built-in)>, '__file__':
    'C:/Users/Milan/Documents/test.py'}
```

Kreiranjem novih promenljivih, u ovaj rečnik se dodaju novi elementi, kojima se može pristupiti na standardni način.

Npr. vrednost neke promenljive može se dobiti kao:

```
a = b = c=123
D='Python'
print (vars())
```

```
{'__name__': '__main__', '__doc__': None, '__package__': None,
    '__loader__': <class '_frozen_importlib.BuiltinImporter'>,
    '__spec__': None, '__annotations__': {}, '__builtins__': <module
    'builtins' (built-in)>, '__file__':
    'C:/Users/Milan/Documents/test.py', 'a': 123, 'b': 123, 'c': 123,
    'D': 'Python'}
```

Dodate su vrednosti za promenljive :a,b,c,D

8.3.5 Struktura skup (set)

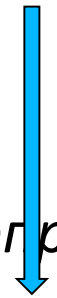
Struktura ***skup*** u jeziku Python služi za predstavljanje neuređene liste jedinstvenih elemenata, tako da ne sadrži duplikate elemenata. Skup se zadaje nabrajanjem njegovih elemenata u velikim zagradama {} ili pomoću funkcije

set(<lista_elemenata>), na primer:.

```
set()                 # funkcija kreira prazan skup
set()
a=set([2,4,6])        # zadavanje elemenata u zagradama
print(a)
{2, 4, 6}
a=set("abcd")         # zadavanje skupa znakova
print(a)
{'a', 'd', 'c', 'b'}
```

```
a="abcd"   # bez SET
print(a)
abcd
```

Напиши програм који за дату реченицу проверава да ли је панграм тако што исписује број различитих слова која се у њој јављају.




```
recenica = "Фијуче ветар"
slova = "абвгдђежзијклљмнњопрстћуфхцчџш"
slova_u_recenici = set(recenica) & set(slova)
a=len(slova_u_recenici)
print("Broj različitih slova u rečenici je:", a)
```

Broj različitih slova u rečenici je: 9

Zašto 9 a ne 10 ili 11? *Ф* *е*

Šta bi dobili kada bi u trećem redu izbrisali: *& set(slova)*

```
recenica = "фијуче ветар "
slova = set("абвгдђежзијклљмнњопрстћуфхцчџш")
slova_u_recenici = set(recenica) & slova
a=len(slova_u_recenici)
print("Broj različitih slova u rečenici je:", a)
print(slova_u_recenici)
```

Broj različitih slova u rečenici je: 10

Zašto je sada 10?

Poredi/Broji samo mala slova, F nije isto što i f

```
recenica = "Фијуче ветар у шибљу, леди пасаже и куће иза
    њих и гунђа у оџацима"
slova = set("абвгдђежзијклљмнњопрстћуфхцчџш")
slova_u_recenici = set(recenica.lower()) & slova
a=len(slova_u_recenici)
print("Broj različitih slova u rečenici je:", a)
print("Ukupan broj slova je " ,  len(recenica))
```

Broj različitih slova u rečenici je: 30

Ukupan broj slova je 66

Za razliku od rečnika, elementi strukture **skup** nemaju posebne ključeve za pristup, već se pristup elementima strukture vrši pomoću njih samih i operatora pripadnosti in i not in.

Osnovne operacije nad strukturom skupa su dodavanje elemenata skupa metodom **add()** i uklanjanje metodom **remove().**

Osnovne operacije nad skupovima u jeziku Python su:

- unija (union, operator |),
- presek (intersection, operator &),
- razlika(difference, operator -) i
- simetrična razlika (symmetric difference, XOR, operator ^), npr.

UNIJA:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1.union(s2)) # s1 unija s2

Rezultat?

{1, 2, 3, 4, 5}

PREKO OPERATORA:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1|s2)

RAZLIKA:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1.difference(s2)) *# s1 razlika s2*

{1, 2}

PREKO OPERATORA:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1-s2)

print(s2-s1)

{1, 2}
{4, 5}

PRESEK:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1.intersection(s2)) # isto kao: s1 presek s2

{3}

PREKO OPERATORA:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1&s2)

Šta bi se dobilo sa: print(s2&s1)?

SIMETRIČNA RAZLIKA:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1.symmetric_difference(s2)) *# s1 simetricna razlika s2*

{1, 2, 4, 5}

PREKO OPERATORA:

s1 = {1,2,3}

s2 = {3,4,5}

print(s1^s2)

Sve isto važi i za slova:

s1 = {'Зрењанин', 'Сомбор'}

s2 = {'Сомбор', 'Суботица'}

s3 = {'Суботица', 'Београд'}

s4 = {'Београд', 'Нови Сад'}

UNIJA SVA 4 SKUPA:

print(s1.union(s2).union(s3).union(s4))

{'Зрењанин', 'Сомбор', 'Суботица', 'Београд', 'Нови Сад'}

s1 = {'Зрењанин', 'Сомбор'}

s2 = {'Сомбор', 'Суботица'}

s3 = {'Суботица', 'Београд'}

s4 = {'Београд', 'Нови Сад'}

(S1 U S2) & S3

S1 unija S2 pa presek sa S3

print(s1.union(s2).intersection(s3))

{'Суботица'}

s1 = {'Зрењанин', 'Сомбор'}

s2 = {'Сомбор', 'Суботица'}

s3 = {'Суботица', 'Београд'}

s4 = {'Београд', 'Нови Сад'}

((S1 U S2) & S3) – S4

print(s1.union(s2).intersection(s3).difference(s4))

{'Суботица'}

Rečenice korišćenjem preseka

```
recenica = "Фијуче ветар"
slova = "абвгдђежзијклљмнњопрстћуфхцчџш"
slova_u_recenici = set(recenica.lower()).intersection(set(slova))
a=len(slova_u_recenici)
print("Broj različitih slova u rečenici je:", a)
#postupno je na sledećm slajdu
```

Broj različitih slova u rečenici je: 10

```
recenica = "Фијуче ветар"
slova = "абвгдђежзијклљмнњопрстћуфхцчџш"
print(set(recenica.lower()))
print(set(slova))
slova_u_recenici = set(recenica.lower()).intersection(set(slova))
a=len(slova_u_recenici)
print("Broj različitih slova u rečenici je:", a)
```

```
{'ј', 'т', 'у', 'е', 'в', 'а', ' ', 'и', 'ч', 'ф', 'р'}
{'ш', 'х', 'п', 'в', 'њ', 'и', 'д', 'ф', 'ј', 'с', 'ч', 'ђ', 'н', 'о', 'б', 'л', 'г', 'р',
    'џ', 'т', 'ц', 'љ', 'м', 'к', 'ћ', 'ж', 'у', 'з', 'а', 'е'}
Broj različitih slova u rečenici je: 10
```

```
recenica = "Фијуче ветар у шибљу, леди пасаже и куће иза
    њих и гунђа у оџацима"
slova = "абвгдђежзијклљмнњопрстћуфхцчџш"
slova_u_recenici = set(recenica.lower()).intersection(set(slova))
a=len(slova_u_recenici)
print("Broj različitih slova u rečenici je:", a)
```

Broj različitih slova u rečenici je: 30

8.4.1 Geografija

Struktura rečnika treba da sadrži podatke o površini dvadesetak najvećih Svetskih država (u km2) :

- Rusija 16.377.742
- Kanada 9.093.507
- Kina 9.569.901
- Sad 9.158.960

Program daje informacije o površini izabrane države prema sledećem algoritmu:

1. Korisnik zadaje ime države za koju se prikažu podaci o njenoj površini.
2. Program pronalazi u rečniku površinu prema imenu države i prikazuje na ekranu računara. Ako država nije u rečniku, program ispisuje poruku "Nažalost nemamo informacija o toj državi".

```
# Površina najvećih zemaija sveta (km2)
drzave = { 'Rusija' : 16377742, 'Kanada' : 9093507,'Kina' : 9569901, 'SAD' : 9158960 }
kraj = False
while not kraj:
    nazivDrzave = input('Unesi naziv države: ')
    if nazivDrzave == '':                    #ENTER označava kraj programa
        kraj = True                          #može za početak bez ova tri reda
    else:
        if nazivDrzave in drzave:
            povrsina = drzave[nazivDrzave]
            print('Površina države', nazivDrzave, 'je',povrsina, ' (km2) ')
        else:
            print('Nažalost nemamo informacija o: ', nazivDrzave)
        print()
```

Unesi naziv države: SAD

Površina države SAD je 9158960 (km2)

Unesi naziv države: Srbija

Nažalost nemamo informacija o: Srbija

Unesi naziv države:

>>>

Program se prekida kada se pritisne ENTER pri unosu države