Univerzitet Singidunum
www.singidunum.ac.rs

Osnove programiranja (Python)

Dr Milan Paroški
mparoski@singidunum.ac.rs
Univerzitet Singidunum
2024/2025

Sadržaj

1. Uvod
2. Promenljive i dodela vrednosti
3. Izrazi
4. Grananje (selekcija)

1.UVOD

1. Osnovni elementi proceduralnih programskih jezika
2. Pregled poznatijih proceduralnih programskih jezika
3. Sintaksa jezika Python

1.1 Osnovni elementi proceduralnih programskih jezika

- Imperativni programi se sastoje od nizova naredbi
- Naredbe mogu biti

- **dodela vrednosti promenljivoj**, pri čemu vrednost može biti i izračunata na osnovu nekog izraza (formule)
  - a=4, a= 5+2
- **različite deklaracije, npr. definicije konstanti** ili tipova promenljivih
  - a= "Zdravo, svete! " print(type(a))
    - <class 'str'>
  - a=34 print(type(a))
    - <class 'int'>
  - a=1.234 print(type(a))
    - <class 'float'>
- Ugrađena funkcija **print()** se koristi za štampanje. Funkciji **print()** prosleđuje ime promenljive kao argument, a ispisuje se njena vrednost, tj. vrednost koju promenljiva sadrži.

Naredbe mogu biti

- **naredbe grananja** – njima se menja redosled izvršavanja niza naredbi
  - If
- **naredbe ponavljanja** (iteracija), kojom se izabrani niz naredbi ponavlja više puta
  - for
- **definicija funkcije** ili procedure (potprograma)
- **naredbe za rad s fajlovima**

1.2 Pregled poznatijih proceduralnih programskih jezika

Programski jezik	Kratki opis
Ada	programski jezik za izradu pouzdanih programa za posebne namene, nazvan po Adi Lovelace, prvom programeru
C	jezik viskog nivoa koji omogućava izradu sistemskih programa
C++	objektno-orijentisan programski jezik zasnovan na jeziku C
C#	programski jezik kompanije Microsoft, hibrid jezika Java i C++
Java	programski jezik namenjen izradi prenosivih aplikacija, nezavisnih od računara i operativnog sistema (Interent)
Pascal	jednostavni strukturirani programski jezik opšte namene, koji se danas uglavnom koristi za učenje programiranja
Python	jednostavni programski jezik opšte namene, pogodan za izradu svih tipova aplikacija
Visual Basic	programski jezik kompanije Microsoft, za brzu izradu Windows aplikacija

1.3 Sintaksa jezika Python: naredbe i linije programa

- Program u jeziku Python je niz naredbi, koje mogu da budu u jednoj ili više *logičkih linija*
- Logička linija se može sastojati od jedne ili više *fizičkih linija*,

```
x = 1CR LF
if x > 0:CR LF
    print("Tri fizičke i logičke linije")CR LF
```

Ovo su tri fizičke i tri logičke linije

i = 10; print(i);

Šta je ovo, koliko ima logičkih a koliko fizičkih linija?

Znači ; se koristi za razdvajanje logičkih linija

Može li obrnuto: više fizičkih linija a jedna logička linija?

```
u=0
v=1
w=2
x=3
y=4
z=5
if u==0 and v>0 \
   and w>1 and x>2 \
   and y>3 and z>4:
   print ("Ovo je primer spajanja linija.“)
```

bold: 3 fizičke i jedna logička linija

Šta je LF?

Oznaka kraja linije teksta: LF (*Line Feed*) za OS *Unix*

The code is \n.

Pritiskom na ovaj taster glava za štampanje ili kursor na ekranu se pozicioniraju na sledeći red

Šta je CR

CR(*Carriage Return*).

carriage return means moving the cursor to the beginning of the line. The code is \r.

Pritiskom na ovaj taster glava za štampanje ili kursor na ekranu se pozicioniraju na levu marginu

Koristi se CR+LF za OS *Windows . Šta ovo znači?*

ZAKLUČAK

Više logičkih linija može se uneti u jednu fizičku liniju pomoću oznake ?

";".

```
a=5;b=6
print(a,b)
```

5 6

Jedna logička linija može se produžiti u više fizičkih pomoću znake nastavka "\", na primer:

```
udaljenost=210
brzina=120
vreme=udaljenost/brzina
print (udaljenost, brzina,
\vreme)
```

210 120 1.75

Sintaksa jezika Python:uvlačenje

- Posebnu ulogu ima uvlačenje teksta programa (indentacija),

koje služi za definisanje *blokova naredbi* u funkcijama,

klasama i naredbama grananja i ponavljanja

- Dubina uvlačenja nije definisana, ali mora biti ista za sve naredbe jednog bloka, npr.

```
x=-2
if (x>0):
    print(x)
else:
    print("X je manje od nule")
```

- **Algoritam**

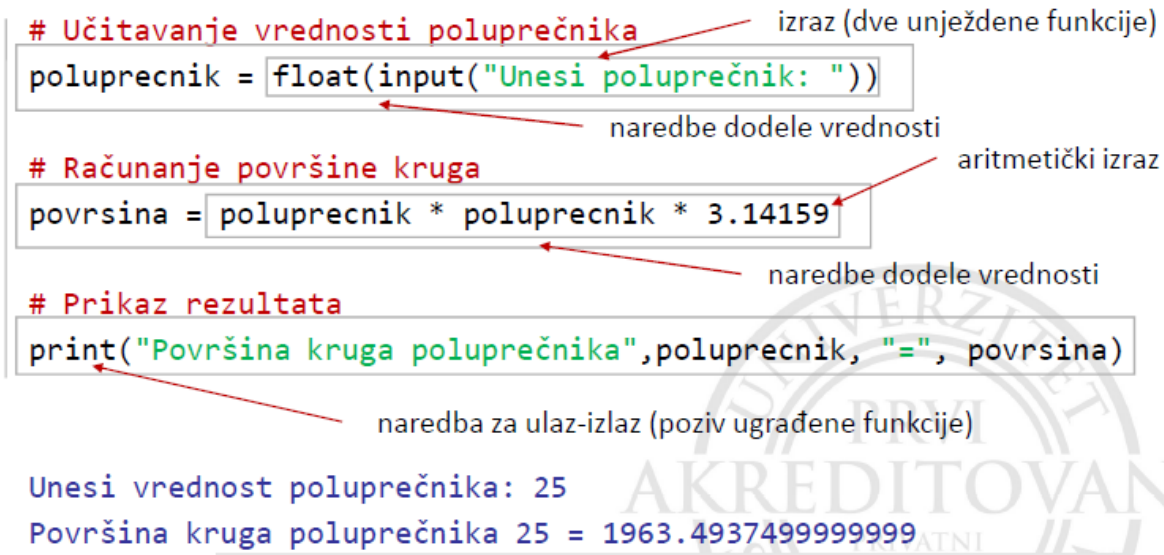
1. učitaj poluprečnik kruga *r*
2. izračunaj površinu po formuli

   $P = r^2 \cdot \pi$
3. prikaži rezultat

- Pitanja implementacije algoritma

– Kako se učitava podatak?
(naredba input)
– Kako se pamti podatak?
(dodela vrednosti =)
– Kako se računa formula?
(*aritmetički izraz*)
– Kako se prikazuje rezultat?
(naredba print)

Elementi programa za računanje površine kruga u jeziku Python



naredba (ugrađena funkcija) **input** čita uneseni tekst i vraća vrednost tipa ***string*.**

```
R=float(input("Unesi poluprecnik kruga:"))
P=R*R*3.14
print("Površina kruga je:",P)
```

Ugrađena funkcija **float** vrši konverziju ove vrednosti **string** u numerički tip **float**

```
Unesi poluprecnik kruga:6.3
Površina kruga je: 124.6266
```

2. Promenljive i dodela vrednosti

1. Nazivi promenljivih
2. Imenovane konstante
3. Naredba dodele vrednosti
4. Oblast definisanosti promenljivih

2.1. Nazivi promenljivih

- Promenljiva je ?

naziv za vrednost zapamćenu u memoriji računara, koja se može menjati

– imena promenljivih u jeziku Python mogu se formirati od malih i velikih slova, znaka "_" i brojeva, ali prvi znak imena mora biti slovo (ili "_")

– neformalna stilska konvencija je da se za imena promenljivih koriste mala slova

– imena se moraju razlikovati od rezervisanih reči jezika Python, koje su za verziju 3:

and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield

Struktura podataka u memoriji

- Promenljive u jeziku Python, za razliku od strogo tipiziranih programskih jezika (a), ne predstavljaju naziv za deo memorije u kome se čuvaju podaci, već *adresu* ili *pokazivač* na takav memorijski prostor s podacima (b)
- Zbog toga promenljive u jeziku Python *nije potrebno deklarisati*

a) Strogo tipizirani jezici

```
x = 1.5
```


```
s = "abcdefgh"
```

b) Python

```
x = 1.5
```


```
s = "abcdefgh"
```

Python je **dinamički tipiziran jezik** - tip promenljive ne mora biti unapred definisan.

Kada dodelite vrednost promenljivoj, Python automatski određuje njen tip na osnovu dodeljene vrednosti.

Ovo omogućava da promenljive menjaju tipove tokom izvršavanja programa bez potrebe za eksplicitnim deklarisanjem tipa kao što je to slučaj u jezicima poput C, C++ ili Java.

Umesto da unapred odlučujete kakav tip podatka će promenljiva imati, možete ga samo koristiti i menjati prema potrebi:

```
age = 30              # age je tipa int
age = "thirty„        # sada je age string
```

Python je **interpretirani jezik**, što znači da interpreter izvršava kod liniju po liniju.

Kada naiđe na novu promenljivu, interpreter je automatski kreira i dodeljuje joj odgovarajući tip, bez potrebe za eksplicitnom deklaracijom.

Deklaracija promenljive-uopšteno

Svaka promenljiva koja se koristi u programu može se deklarisati pre upotrebe.

Deklaracijom promenljive definiše se:

- naziv,
- tip,
- opciono njena početna vrednost i
- vidljivost tj. prava pristupa

Na primer:

```
int  godine;
godine=33
public:godine

private int BrojUcenika = 1;
```

2.2.Imenovane konstante

• Imenovana konstanta je naziv za vrednost zapamćenu u memoriji računara, koja se načelno *ne može* menjati

• Jezik Python *nema* posebne oznake za konstante, već se one kreiraju kao *promenljive* čiju vrednost program dobrovoljno neće menjati,

Konstante se stilski pišu velikim slovima

PI = 3.14159

E = 2.71828

BRZINA_SVETLOSTI = 299792458

Kako ispisati konstantu PI?

```
import math;
print(math.pi)
```

2.3. Naredba dodele vrednosti

• Naredba je oblika

<promenljiva> = <izraz>

• Tip promenljive definisan je tipom dodeljene vrednosti:

x = 1

x = 1.0

x = 2*x

ime = "Jelena"

pozdrav = "Zdravo " + ime

– tip vrednosti promenljive može se ispitati naredbom/funkcijom type(),
npr. type(x) daje <class 'float'>,
type(ime) daje rezultat <class 'str'>

- Jezik Python dozvoljava višestruke dodele vrednosti?

a b c = 1 2 3

a, b, c = 1, 2, 3

a: b : c = 1 : 2 : 3

a; b ; c = 1 ; 2 ; 3

Šta je tačno?

a, b, c = 1, 2, 3

Da li je ovo dozvoljeno? a = b = c = 0

Šta se dobija sa:print(a,b,c)

0 0 0

Šta se dobije sa: a,b,c=1,2,3; print(a,c)

1 2 3

2.4. Oblast definisanosti promenljivih

- Oblast definisanosti promenljve (*scope*) je deo programa u kome je vrednost promenljive definisana i može se koristiti
- Npr. ako je na početku programa naredba

brojac= brojac + 1

print(brojac)

Šta će biti ispisano?

sistem dojavljuje grešku jer vrednost promenljive *brojac* tada još nije definisana

NameError: name 'brojac' is not defined

Kada će biti izraz dobar (bez greške)?

brojac=5
brojac=brojac+1
print(brojac)

Rezultat je 6

3. Izrazi

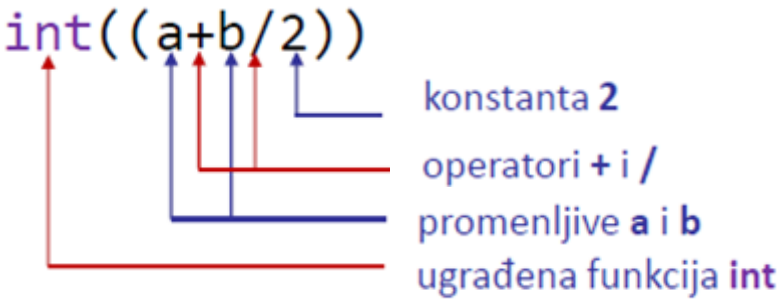
1. Pojam
2. Evaluacija izraza
3. Operatori

3.1. Pojam izraza

• Izrazi su pravilna kombinacija *vrednosti* (**konstanti**), ***promenljivih***, ***funkcija*** i ***operatora***, npr.

Šta je 2? Šta je /? Šta je b? Šta je int?

```
a,b=1,2
c=int((a+b/2))
print(c)
```



int((a+b/2))

• Izrazi nakon izračunavanja (evaluacije) daju rezultat određenog tipa, koji se može dodeliiti nekoj promenljivoj

– rezultat može biti numerički, nenumerički ili logički (podtip int)

– u gornjem primeru, za a=1 i b=2,2 rezultat evaluacije je ? Tip je?

2 (tip int)

3.2.Evaluacija izraza

- Redosled operacija prilikom izračunavanja izraza zavisi od tipa vrednosti i operatora
- Za aritmetičke izraze, određuje se prema redosledu operacija u matematici, po prioritetu operatora i zagradama, npr.

3 + 4 * 5 ** 2 ?

103. Da li se razlikuje od

(3 + 4) * 5 ** 2 ?

175

- Kod logičkih, operacija ko ima prednost (and ili or)?

and ima prednost nad or , npr.: True or False and False daje ?

True

False or True and False daje ?

False

- Svi operatori istog prioriteta evaluiraju se s leva u desno.

Zagrade menjaju redosled svih tipova operatora

2.čas

operand 1	operand 2	and	or
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

		and	or
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

And je kao množenje
Or je kao sabiranje

3.3.Operatori

• Operatori u jeziku Python definisani su za određeni tip vrednosti na koje se mogu primeniti

• Razlikuju se:

1. Aritmetički operatori
2. Operatori poređenja
3. Specijalni operatori (uz dodelu vrednosti)
4. Logički operatori
5. Operatori za rad s bitovima
6. Operatori pripadnosti
7. Operatori identifikacije

Šta znače UNARNI operatori?

• U odnosu na broj argumenata operatori mogu biti *unarni* (jedan argument) i *binarni* (dva argumenta)

3.3.1. Aritmetički operatori

- Aritmetički operatori su

+ sabiranje numeričkih vrednosti

- oduzimanje numeričkih vrednosti

***** množenje numeričkih vrednosti

/ deljenje numeričkih vrednosti, decimalni rezultat

// šta je ovo?

celobrojno deljenje, decimalni deo se odbacuje

****** stepenovanje

% šta je ovo?

ostatak od deljenja (modul)

1) ostatak od deljenja sa 2 za parne brojeve jednak je nuli

2) operator % je primer jednostavne jednosmerne funkcije, za koju ne postoji inverzna funkcija . Za čega je onda pogodna?

za šifrovanje - bez podatka o deliocu, ne može se znati deljenik

```
5/2
2.5
5//2
2
5%2
1
5%5
0
5%2.45
0.09999999999999964
```

Napomena

U jeziku Python operatori **+** i ***** koriste se još kao oznake za

operacije s nenumeričkim vrednostima (stringovima)

- Za argumente tipa *string, šta je +?*

operator **+** označava konkatenaciju (nastavljanje) stringova

Šta je *?

operator ***** u kombinaciji s brojem označava višestruku konkatenaciju, odnosno ponavljanje (repeticiju) stringa

1. Konkatenacija
"Novi"+" Sad"
'Novi Sad'

2. Repeticija
"Novi Sad je najlepši grad" + "!"*7
Šta je rezultat?
Novi Sad je najlepši grad!!!!!!!

3.3.2. Operatori poređenja

- Operatori poređenja u izrazima su

< manje od

<= manje ili jednako

> veće od

>= veće ili jednako

==

jednako

!=

različito **(**nije jednako)

```
2>3
False
2<3
True
2=3
  File "<stdin>", line 1
SyntaxError: can't assign to literal
2==3
False
2!=3
True
```

3.3.3. Specijalni operatori dodele vrednosti

- Neki aritmetički operatori mogu se kombinovati s operatorom dodele vrednosti (=) radi skraćivanja teksta programa:

+, **-**, ***, **/**, **//**, %, i **

- Npr. umesto naredbe

brojac = brojac + 1

dozvoljeno je pisati kraće ŠTA?

brojac += 1

- Između ovako kombinovanih operatora ne sme biti praznina

```
a=0
a=a+2
print(a)
2
b=0
b +=2
print(b)
2
```

3.3.4 Logički operatori

- Logički operatori nad celobrojnim vrednostima su:

not unarni operator negacije

and binarni operator *logičko I*

or binarni operator *logičko ILI*

```
a=True
not a  je ?
False
a and True
True
a and False
False
a or True
True
a or False
True
```

3.3.5. Operatori za rad s bitovima

- Operatori za rad s bitovima su

~ negacija na nivou poslednjeg bita (*bitwise* NOT)

& operacija I nad bitovima (*bitwise AND*)

^ operacija ekskluzivno ILI nad bitovima (*bitwise XOR*)

```
a=8;print(bin(a)) je?
'0b1000    ZAŠTO?
b=~a; print(b)   je?
-9
```

```
a=8
print(a, bin(a))
b=9
print(b, bin(b))
c=a&b
print(c, bin(c))
Rezultat:
8 0b1000
9 0b1001
8 0b1000
```

```
a=8;b=9
print(a, bin(a))
print(b, bin(b))
c=a^b
print(c, bin(c))
Rezultat:
8 0b1000
9 0b1001
1 0b1
```

A = 10 => 1010 (Binary)

B = 7 => 111 (Binary)

A & B = 1010

&

0111

= 0010

= 2 (Decimal)

Bitwise AND Operator

| operacija ILI nad bitovima (*bitwise OR*)

<< pomeranje bitova ulevo određen broj mesta (*shift left*)

>> pomeranje bitova udesno određen broj mesta (*shift right*)

– operacije nad nizovima bitova koriste se za kodiranje i dekodiranje, npr. u kriptologiji, kao i za rad s uređajima i komunikacionim protokolima

```
a=2
e=a<<2
print(e)
8
a=3
e=a<<2
print(e)
12
a=2
e=a>>2
print(e)
0
a=12
e=a>>2
print(e)
3
```

3.3.6. Operatori pripadnosti

- Operatori pripadnosti su?

in provera pripadnosti neke vrednosti nizu vrednosti – nizu znakova (*string*), listi ili n-torci

not in negacija pripadnosti nizu vrednosti

10 in (10,20,30)
True
10 not in (1,2,3)
True

3.3.7. Operatori identifikacije

- Operatori identifikacije su ?

is provera da li dve promenljive pokazuju na isti objekt u memoriji

is not negacija provere identičnosti objekata

```
a="a"
print(a+a)
Rezultat?:
aa
print("aa" is a+a)
Rezultat?:
False

print("aa" is not a+a)
True
```

PREGLED

** eksponent (exponentiation)

+x, **-**x, **~**x unarni operatori predznaka i negacija na nivou bita (*bitwise* NOT)

*, **@**, **/**, **//**, **%** množenje, množenje matrica, deljenje i ostatak deljenja

+, **-** dodavanje i oduzimanje (a*ddition*, *subtraction*)

<<, **>>** pomeranje bitova (*shifts*)

& operacija AND nad bitovima (*bitwise AND*)

^ operacija XOR nad bitovima (*bitwise XOR*)

| operacija OR nad bitovima (*bitwise OR*)

in, **not in**, **is**, **is not**,

<, **<=**, **>**, **>=**, **!=**, **==**

poređenja i testovi pripadnosti (in) i identiteta (is)

not x logička operacija NOT (*boolean NOT*)

and logička operacija AND (*boolean AND*)

or logička operacija OR (*boolean OR*)

Konverzija tipova

- Jezik Python prilikom operacija u izrazima vrši prilagođavanje tipova vrednosti
- Eksplicitna konverzija tipa vrednosti vrši se ugrađenim funkcijama: ?

int(),

float(),

bool() i

str()

```
print(int(3.1))
print(float(3))
print(bool(True))
print(5+1)
print(str(5)+1)
```

3
3.0
True
6
Traceback

3.čas

Zaokruživanje round int

- Aritmetičke operacije proizvode decimalne rezultate određene preciznosti
- Zaokruživanje decimalnih vrednosti vrši se ugrađenom fukcijom round(), npr.

round(3.14) daje rezultat

3

round(2.71) daje rezultat

3

round(3.5) daje rezultat

4

round(2.71, 1) daje rezultat

2.7

round(2.71, 2) daje rezultat

2.71

- Poseban vid zaokruživanja je odbacivanje decimalnog dela pomoću ugrađene funkcije int(), npr.

int(3.14) daje rezultat

3

int(2.71) daje rezultat

2

4. Grananje (selekcija)

1. Šta je grananje
2. Sintaksa naredbe grananja
3. Primer

4.1.Šta je grananje

• Grananje je naredba za izbor (selekciju) grupe naredbi koje će se izvršiti ako je zadovoljen određeni uslov

<i**f_naredba**>::= if <uslov> : <blok naredbi>
(**elif** <uslov> : <blok naredbi>)*
[**else** : <blok naredbi>]

– uslov je izraz bilo kog tipa koji se tretira kao istinit (*True*) za svaku vrednost osim za numeričku vrednost nula, prazan string, listu i sl.
– ako je logički uslov istinit (*True*), izvršava se blok naredbi
– ako logički uslov *nije* istinit (*False*), ispituju se sledeći uslovi, ako su navedeni u dodatnim stavkama **elif**
– ako *nijedan* navedeni uslov nije istinit, izvršava se blok naredbi iza stavke **else**, ako je naveden

Šta je grananje

- Naredba selekcije može da ima jednostavnu ili složenu formu,

za izbor jedog, dva ili više blokova naredbi, npr.

Jednostavna selekcija	Dve grupe naredbi	Tri i više grupa naredbi

`if uslov:`
`  blok naredbi`
	`if uslov:`
	`  blok naredbi 1`
	`else:`
	`  blok naredbi 2`
		`if uslov1:`
		`  blok naredbi 1`
		`elif uslov2:`
		`  blok naredbi 2`
		`...`
		`else:`
		`  blok naredbi n`

Primer

```
lepo_vreme=input("Unesite reč da ili ne:")

if lepo_vreme == "da":
    print('Nije vam potreban kišobran.')
```

Šta staviti u elif?

```
lepo_vreme=input("Unesite reč da ili ne:")

if lepo_vreme == "da":
    print('Nije vam potreban kišobran.')
elif lepo_vreme =="ne":
    print("Ponesite kisobran")
```

Šta staviti u else?

```
lepo_vreme=input("Unesite reč da ili ne:")

if lepo_vreme == "da":
    print('Nije vam potreban kišobran.')
elif lepo_vreme =="ne":
    print("Ponesite kisobran")
else:
    print("niste dobro odgovorili")
```

Još dodatni uslovi: gde se mogu staviti?

```
lepo_vreme=input("Unesite reč da ili ne:")
if lepo_vreme == "da":
    print('Nije vam potreban kišobran.')
elif lepo_vreme =="ne":
    print("Ponesite kisobran")
elif lepo_vreme ==“mozda":
    print(“Ipak ponesite kisobran")
else:
    print("niste dobro odgovorili")
```

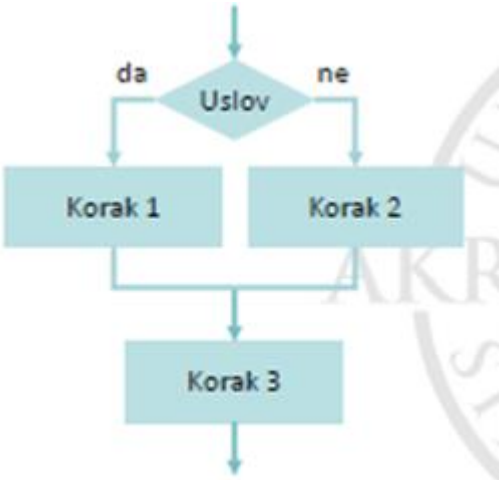
Slozeniji primer granjanja

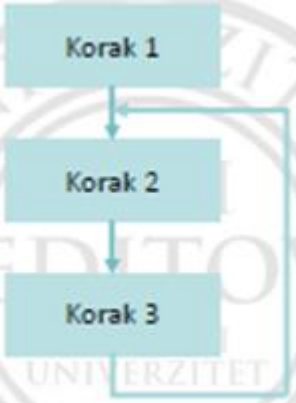
```
starost=int(input("Unesi broj godina :"))
if starost < 18:
    print('popust za decu i učenike')
elif starost > 65:
    print('nije potrebna karta')
else:
    print('puna cena karte')
```

Algoritmi i tok izvršavanja programa

• Za realizaciju bilo kog algoritma dovoljna su tri načina upravljanja izvršavanjem pojedinih koraka algoritma

1. sekvencijalno izvršavanje, jedan korak za drugim (podrazumeva se)
2. uslovno izvršavanje, gde naredni korak zavisi od određenih uslova
3. ponavljanje, gde se određeni niz koraka izvršava više puta



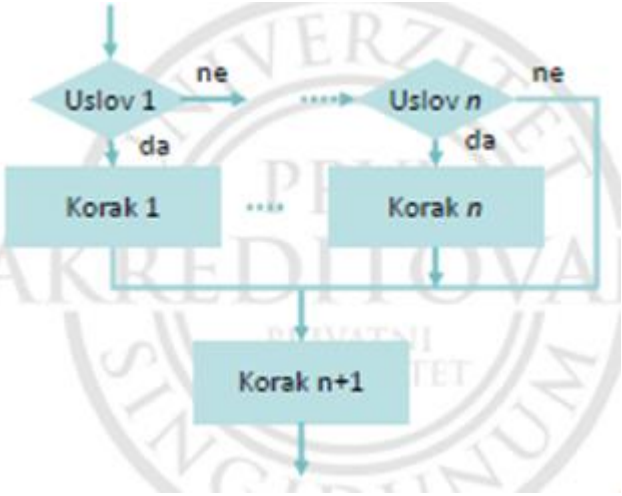


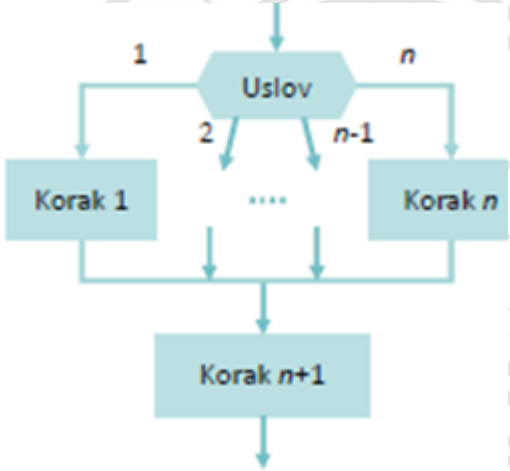
Grananje u proceduralnim jezicima

• U proceduralnim programskim jezicima grananje se može realizovati prema :

- logičkoj vrednosti (dve alternative) ili
- na osnovu vrednosti nekog drugog tipa (više od dve alternative)

Python podržava grananja prema logičkom uslovu, odnosno nizu povezanih logičkih uslova (if ... elif .. else)





Ponavljanje u proceduralnim jezicima

• U proceduralnim programskim jezicima ponavljanje se realizuje u odnosu na uslov okončanja ponavljanja (petlje)

– ponavljanje unapred određen broj puta.

• Python podržava ponavljanje unared određen broj puta sa komandom ?

(for)

– ponavljanje prema logičkom uslovu, koji se proverava pre početka ponavljanja ili nakon svakog ponavljanja

• U ponavljanju prema uslovu, niz naredbi koji se ponavlja obezbeđuje *promenu uslova ponavljanja*, tako da ponavljanje pod određenim uslovima okonča

• Python podržava ponavljanje ponavljanje prema logičkom uslovu koji se proverava pre početka ponavljanja sa komandom ?

(while)

• Naredba grananja omogućava izbor na osnovu logičkog uslova sledeće naredbe koja će se izvršiti, npr.

```
if True:
    print("da")
else:
    print("ne")
```

• Blokovi ili grupe naredbi u jeziku Python označavaju se uvlačenjem (indentacijom)

• Grananje nije ograničeno na rezultate ispitivanja jednog uslova, već se mogu koristiti složeni logički izrazi

4.2. Sintaksa naredbe grananja

- Opšti oblika naredbe grananja (selekcije) je: if,else,if,else

Primer:napisati program da da ocenu na osnovu broja bodova

```
poen =input("Unesi broj poena na ispitu:")
poen=int(poen)
if poen<51:
    print("Ocena je :",5)
else:
    if poen<61:
        print("Ocena je :",6)
    else:
        print ("Ocena je izmedju 6 i 10")
```

Logički uslov

- Logički uslov je izraz koji nakon evaluacija proizvodi logičku vrednost (True ili False)
- To može biti rezultat primene različitih operatora, kao što su operatori poređenja, logički operatori i operatori pripadnosti
- Izraz se izračunava u skladu s prioritetima operatora

```
a= 2
if a > 10 or a < 0 or a == 6 :
    print("Neispravno")
else:
    print("Ispravno")
```

pojašnjenje 

Evaluacija uslova

a > 10 or a < 0 or a == 6

Šta će se pre računati: 10 or a ili a > 10 ?

- **Relacioni operatori imaju prioritete** u odnosu na logičke operatore, tako da se uslov iz prehodnog primera:

a > 10 or a < 0 or a == 6

- izračunava kao:

(a > 10) or (a < 0) or (a == 6)

- Npr. za vrednost a=2, nakon evaluacije izraza u zagradama, evaluira se False or False u False, a konačni rezultat je ispis poruke "Ispravno"

4.3. Primer upotrebe naredbe while

• Primer upotrebe je petlja koja menja brojač s korakom 3, sve dok je vrednost brojača veća od nule:

```
broj = 10
while broj > 0:
    print(broj)
broj = broj-3
```

```
broj=10
while broj>0:
    print(broj)
    broj=broj-3
```

```
10
7
4
1
```

Šta će biti rezultat?

Zašto ovako radi?

• Modifikacija uslova petlje omogućava okončavanje ponavljanja, jer se brojač monotono smanjuje do nule

• Kada bi se modifikacija izmenila tako u broj = broj+1,

Šta bi se desilo?

Petlja ne bi mogla normalno da okonča

Beskonačna petlja:

```
broj=10
while broj>0:
   print("broj=:",broj)
   broj=broj+3
```

```
broj=: 10
broj=: 13
broj=: 16
broj=: 19
broj=: 22
broj=: 25
broj=: 28
broj=: 31
broj=: 34
broj=: 37
broj=: 40
broj=: 43
```

Praktične preporuke za upotrebu naredbi selekcije

Pažljivo koristiti indentaciju za definisanje bloka naredbi

```
r = -10
if r >= 0:
    p = r * r * PI
print("Površina =", p)
```

```
r = -10
if r >= 0:
    p = r * r * PI
    print("Površina =", p)
```

Koje je pogrešno?
Da li je uopšte nešto tačno?

Znači potrebno je do sledećeg ELIF ili ELSE pomerati pisanje za 4 znaka

Praktične preporuke za upotrebu naredbi selekcije

```
i=1
j=2
k=3
if i> j:
    if i > k:
        print("A")
else:
    print("B")
```

Šta se štampa?

B

```
i=1
j=2
k=3
if i> j:
    if i > k:
        print("A")
    else:
        print("B")
```

Ništa se ne štampa