Upravljanje tokom izvršavanja programa - grananje i ponavljanje

Dr Milan Paroški

2024/2025

Sadržaj

1. Ponavljanje (iteracija)
2. Obrada izuzetaka
3. Primeri programa

1. PONAVLJANJE (ITERACIJA)

1. Pojam
2. Naredba for
3. Naredbe za prekid ponavljanja
4. Naredba while
5. Primer

2.1. Pojam

- Naredba ponavljanja u jeziku Python omogućava višestruko izvršavanje određenog bloka naredbi
- **Broj ponavljanja može biti unapred određen ?**

(naredba for)

Ili se može postaviti logički uslov za izvršavanje ponavljanja ?

(naredba while)

2.2. Naredba for

- Ponavljanje (iteracija) zadani broj puta:

for <promenljiva> in <skup vrednosti>:

<blok naredbi>

- **blok naredbi se izvršava za svaku pojedinačnu vrednost iz zadanog skupa vrednosti**
- Skup vrednosti može biti eksplicitno zadan nabrajanjem pojedinačnih vrednosti ili proizveden, npr. pomoću funkcije ?

range()

- funkcija range(...) omogućava generisanje skupa vrednosti za koje se vrši svako ponavljanje.

Ugrađena funkcija range()

• Funkcija range() generiše skup vrednosti celobrojnog tipa:
range(<početak>, <kraj- ali sam kraj nijeuključen>, <korak>)

funkcija range(n) čini kolekciju
1, 2, ..., n-1;

funkcija range(a, b) čini kolekciju
a, a+1, ..., b-1;

funkcija range(a, b, k) čini kolekciju
a, a+k, a+2k, ..., b -k,

Ugrađena funkcija range()

Funkcija range() kreira listu elemenata.

Podrazumeva se korak:1

Ako primi jednu vrednost, tj. ako je poziv funkcije range() sedećeg oblika:
range(10)

Kreira se lista od 10 elemenata, prvi element poprima vrednost ?

0,

dok zadnji element poprima vrednost ?

9.

Funkcija range() može primiti i dve vrednosti. Podrazumeva se korak:1

range(1,11) ?

skup vrednosti 1,2,..,10

range(0,10,3)

skup vrednosti 0,3,6,9

range(0,-10,-1)

skup vrednosti 0,-1,-2,..,-9

Kako ćemo napisati ovo u editoru?

for i in range(0,-10,-1): print(i)

Rezultat?

```
>>> for i in range(0,-10,-1): print(i)
...
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
>>>
```

range(20,1,-2)

generiše skup vrednosti 20,18,16,14,12,1 0, 8, 6, 4, 2

range(13, 5, 1)

range(1,1,1)

generiše prazan skup vrednosti

range(1,2,2)

1

range(0)

generiše prazan skup vrednosti

range(1,0)

generiše prazan skup vrednosti

range(0.0)

za decimalne argumente nije definisana:
Traceback (most recent call last):
File "<pyshell#5>", line 1, in <module>
range(0.0) TypeError: 'float' object cannot be interpreted as an integer

Range(10,10) ?

for i in range(10,10): print("Dobro jutro!")

?

for i in range(1,10): print("Dobro jutro!")

?

Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!

Znači za razliku od nekih drugih programskih jezika, kao što su C, C++, Java itd, u programskom jeziku Python, petlja for koristi se na drugačiji način.

Petlja for u Pythonu iterira kroz elemente zadane sekvence.

Zadana sekvenca može biti lista ili niz znakova.

Da li će ovo raditi:

```
x=1
for i in range(x,x+10): print("Dobro jutro!")
```

```
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
Dobro jutro!
```

Primer koji određuje je li neki proizvoljni broj spremljen u varijablu n, prost ili ne.

Prosti brojevi su svi prirodni brojevi uz uslove?:

- veći od broja 1 i
- koji su djeljivi bez ostatka samo s brojem 1
- deljivi sami sa sobom

(na primer: 2, 3, 5, 7, 11, 13, 17, ...)

16.10.2024.

Termin za konsultacije:

Na fakultetu: sredom od 13 do 14 časova

Online: ponedeljkom od 18 do 19 časova

Danas prezentacija 3 projekta:

11:00 – 11:15 - **Reddit:Saška Miljković**

11:15 – 11:30 - **PyCon: Stefan Umiljanović**

Mes	Pon	Uto	Sre	Čet	Pet	Sub	Ned	Aktivnosti studenata
IX	23	24	25	26	27	28	29	I nedelja nastave
X	30	1	2	3	4	5	6	II nedelja nastave
	7	8	9	10	11	12	13	III nedelja nastave
	14	15	16	17	18	19	20	IV nedelja nastave
	21	22	23	24	25	26	27	V nedelja nastave / Oktobarski ispitni rok
	28	29	30	31	1	2	3	VI nedelja nastave - kolokvijumska nedelja

Mes	Pon	Uto	Sre	Čet	Pet	Sub
XI	23	24	25	26	27	28
X	30	1	2	3	4	5
	7	8	9	10	11	12
	14	15	16	17	18	19
	21	22	23	24	25	26
	28	29	30	31	1	2
XI	4	5	6	7	8	9
	11	12	13	14	15	16
	18	19	20	21	22	23

PRVI NAČIN: Da li je 7 prost broj.py

```
n = 7                          #N dobija vrednost 7
brojjeprost = True             #Promenljiva-pretpostavka:broj prost
for x in range(2, n):          #X ide od 2 do 6 a ne treba i 7

   if n % x == 0:              #Ako je ostatak od deljenja jednak 0 broj nije prost
      brojjeprost = False          #Broj je prost kada je ostatak deljenja 0
      print(x, " nema ostatka ",brojjeprost)
   else:
      print(x, " ima ostatka ",brojjeprost)   #kraj for petlje

if brojjeprost == True:          #Ako je ostatak od deljenja različit od 0
   print('Broj', n, 'je prost!')   #Broj je prost
else:
   print('Broj', n, 'nije prost!') #Broj nije prost
```

7 pa promenuti u 8

Ostatak deljenja 7/2 je 1
“Broj je prost”
Ostatak deljenja 7/3 je 1
“Broj je prost”
Ostatak deljenja 7/4 je 3
“Broj je prost”
Ostatak deljenja 7/5 je 2
“Broj je prost”
Ostatak deljenja 7/6 je 1
“Broj je prost”

- Ostatak deljenja 8/2 je 0
- “Broj nije prost”
- Ostatak deljenja 8/3 je 2
- “Broj nije prost”
- Ostatak deljenja 8/4 je 0
- “Broj je prost”
- Ostatak deljenja 8/5 je 3
- Broj nije prost
- Ostatak deljenja 8/6 je 2
- Broj nije prost
- Ostatak deljenja 8/7 je 2
- Broj nije prost

2.3.Naredbe za prekid ponavljanja

• U Python-u postoje dva načina prekida ponavljanja, potpuno ili delimično, pomoću naredbi **break i continue**:

– naredba **break** *prekida ?*

izvršavanje petlje, a izvršavanje programa nastavlja se od prve sledeće naredbe iza petlje

– naredba **continue** ?

okončava izvršavanje jednog *koraka* petlje i program nastavlja izvršavanje petlje, ali od sledeće iteracije

Ove naredbe su slične naredbi skoka ili bezuslovnog grananja (*goto*), ali se koriste samo u okviru petlji, tako da su manje problematične

U opštem slučaju se u realizaciji algoritama ***izbegava upotreba naredbi skoka (prekida),*** *iako se ponekad može dobiti kraći i razumljiviji kod programa*

Naredba break nam služi da?

izađemo iz for ili while petlje.

Kako petlje mogu biti jedna unutar druge, treba obratiti pažnju na to da **naredba break izlazi samo iz petlje unutar koje je pozvana**.

Ova naredba se dosta često koristi kod sprečavanja beskonačnih petlji.

Zašto nam ovo treba u prethodnom programu?

U nastavku je poboljšani primer programskog kôda koji određuje ako je broj prost ili ne.

Za razliku od prethodnog primera ovog istog zadatka, u ovom primeru jednom kada se pronađe da je neki broj n djeljiv s nekim drugim brojem, daljnja analiza deljivosti broja n s ostalim brojevima prestaje.

Unutrašnja for petlja u if telu sadržava ključnu riječ **break**

Break nam u ovom slučaju omogućava da izađemo iz unutrašnje for petlje.

DRUGI NAČIN: SA BREAK: Da li je 7 prost broj

PRVO:Ubaciti break na sajd 14 posle else

```
n = 7                                          #n dobija vrednost
for x in range(2, n):                          #x ide od 2 do n-1
    print(x)                                   #štampaj vrednost x
    if n % x == 0:                             #Da li je ostatak od deljenja jednak 0
        print(x,"nema ostatka","Broj nije prost")   #DA,Broj nije prost
        ind=0                                  #indikator za Broj nije prost
        break                                  #ovo je dovoljno, izlaz iz FOR petlje
    else:
        ind=1                                  #NE, indikator za Broj je prost

if ind== 1:
    print("Prost")
else:
    print("Nije prost")
```

Probati sada sa n=8

Treći način

```
n=7
provera = True                  #promenljiva je ISTINA-broj nije prost
for x in range(2, n):               #x ide od 2 do n
   if n % x == 0:           #Da li  je ostatak deljenja jednak 0
       provera = False      #DA, promenljiva je NEISTINA-prost broj
   print(x)
if provera == True:         #Petlja za ispis
     print('Broj', n, 'je prost!')
else:
    print('Broj', n, 'nije prost!')
```

U čemu je razlika od prvog načina?

Nema ELSE

Probati i sa 8

Ako se želi ispisati lista prostih brojeva iz intervala [2, 9], prvo se ?

pomoću petlje izgeneriše popis svih brojeva, neovisno o tome je li generirani broj prost ili ne.

Primer koji ispisuje sve brojeve iz intervala [2, 9].

```
for n in range(2, 9):
    print(n)
```

Rezultat:

2
3
4
5
6
7
8

Ova petlja ispisuje sve brojeve iz intervala [2, 9].

Ako se žele ispisati samo prosti brojevi, tada je potrebno ?

funkciju print(n) zameniti implementacijom koja otkriva ako neki broj jest ili nije prost.

```
for n in range(2, 9):
    provera = True          #promenljiva je ISTINA-broj nije prost
    for x in range(2, n):   #x ide od 2 do n
        if n % x == 0:      #Da li  je ostatak deljenja jednak 0
            provera = False     #DA, promenljiva je NEISTINA-prost broj
        print(x)
    if provera == True:         #Petlja za ispis
        print('Broj', n, 'je prost!')
    else:
        print('Broj', n, 'nije prost!')
```

REZULTAT:

Broj 2 je prost!

2

Broj 3 je prost!

2

3

Broj 4 nije prost!

2

3

4

Broj 5 je prost!

2

3

4

5

Broj 6 nije prost!

2

3

4

5

6

Broj 7 je prost!

2

3

4

5

6

7

Broj 8 nije prost!

Kako izbaciti medju rezultate?

Staviti komentar na liniji print(x)

```
for n in range(2, 9):
    provera = True          #promenljiva je ISTINA-broj nije prost
    for x in range(2, n):   #x ide od 2 do n
        if n % x == 0:      #Da li  je ostatak deljenja jednak 0
            provera = False     #DA, promenljiva je NEISTINA-prost
     broj
        #print(x)
    if provera == True:         #Petlja za ispis
         print('Broj', n, 'je prost!')
    else:
         print('Broj', n, 'nije prost!')
```

Rezulatat:

Broj 2 je prost!

Broj 3 je prost!

Broj 4 nije prost!

Broj 5 je prost!

Broj 6 nije prost!

Broj 7 je prost!

Broj 8 nije prost!

```
for n in range(2, 9):
    brojjeprost = True
    for x in range(2,n):
        if n % x == 0:
            brojjeprost = False
            break
    if brojjeprost == True:
        print('Broj', n, 'je prost!')
    else:
        print('Broj', n, 'nije prost!')
```

Rezultat:

```
Broj 2 je prost!
Broj 3 je prost!
Broj 4 nije prost!
Broj 5 je prost!
Broj 6 nije prost!
Broj 7 je prost!
Broj 8 nije prost!
```

Naredba continue

Naredba continue skače na novu iteraciju petlje.

Za razliku od naredbe break, **naredba continue ne prekida izvođenje programa.**

Sledeći kôd koji ispituje je li broj u datoj listi paran ili neparan

Provera parnosti

```
for n in range(2, 6):                    #n ide od ? do ?
    if n % 2 == 0:                       #Da li je ?
        print("Parni broj:", n)
        continue                         #skače na ?
    print("Neparni broj:", n)
```

Šta bi se desilo da izostavimo naredbu : continue?

Rezultat:

Parni broj: 2
Neparni broj: 2
Neparni broj: 3
Parni broj: 4
Neparni broj: 4
Neparni broj: 5

Rezultat:

Parni broj: 2
Neparni broj: 3
Parni broj: 4
Neparni broj: 5

Identično je sa else:

```
for n in range(2, 6):
    if n % 2 == 0:
        print("Parni broj:", n)
    else:
        print("Neparni broj:", n)
```

```
#n ide od 2 do 5
#Da li je n deljivo sa 2
#DA. Broj je paran
#NE.Broj je neparan
```

```
Parni broj: 2
Neparni broj: 3
Parni broj: 4
Neparni broj: 5
```

2.4. Naredba while

- Osnovna sintaksa naredbe je:

while <uslov>:

<blok naredbi s modifikacijom uslova>

– da bi se blok naredbi izvršio bar jednom , *uslov* petlje treba da bude istinit (True) pre izvršavanja naredbe

– modifikacija uslova treba da omogući okončanje petlje, za šta je potrebno da izraz u uslovu petlje promeni vrednost istinitosti u False

Primeri upotrebe naredbe while

- Primer upotrebe je petlja koja menja brojač s korakom 3, sve dok je vrednost brojača veća od nule (PRIMER SA PROŠLOG ČASA)

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

```
10
7
4
1
```

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

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

Petlja ne bi mogla normalno da okonča

Kako napisati Dobar dan 10 x korišćenjem naredbe WHILE?

Ovo treba da bude rezultat:

Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan

```
x = 0                    #X dobija vrednost 0
while x < 10:            #ide se dok X ne bude 10
    print(" Dobar dan")     #štampa
    x = x + 1               #inkrement X
```

```
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
Dobar dan
>>>
```

FOR koliko je trebalo naredbi?
2
A za WHILE ?
4 naredbe

```
for n in range(1,10):
    print("Dobar dan")
```

Šta je rezultat ove naredbe?

while True: print("Hello ")

Beskonačno ispisivanje…

Napisati program za sumu prvih 5 prirodnih brojeva

```
zbir = broj = 0                    #obe promenljive dobijaju vrednost 0
while broj < 6:                #petlja dok je broj manji od 20
    broj = broj + 1                #broj se inkrementira
    zbir = zbir + broj             #Zbir se sabora sa brojem

print("Vrednost promenljive broj je", broj)
print("Zbir je", zbir)
```

Primer 1: Upotreba prekida izvršavanja petlje

Sledeći program dodaje promenljivoj "zbir" cele brojeve 1..20 I prekida rad kad suma bude veća ili jednaka 100:

```
zbir = broj = 0                    #obe promenljive dobijaju vrednost 0
while broj < 20:                   #petlja dok je broj manji od 20
    broj = broj + 1                #broj se inkrementira
    zbir = zbir + broj             #Zbir se sabira sa brojem
    if zbir >= 100:                #ako je Zbir manji od 100 kraj
        break
print("Vrednost promenljive broj je", broj)
print("Zbir je", zbir)
```

Rezultat izvršavanja programa je:

Vrednost promenljive broj je 14

Zbir je 105

Bez selekcije i naredbe break:

WHILE zbir prvih 20 brojeva.py

```
zbir = broj = 0
while broj < 20:
    broj = broj + 1
    zbir = zbir + broj

print("Vrednost promenljive broj je", broj)
print("Zbir je", zbir)
```

1. Stavljamo # (komentare) kod naredbi IF i BREAK (peti i šesti red pprethodnog programa)

Rezultat:

Vrednost promenljive broj je 20

Zbir je 210

2. Stavljamo # (komentar) kod naredbe BREAK i uvučemo prvi sledeći print za dve identacije

3. Stavljamo # (komentar) kod naredbe BREAK i uvučemo prvi i drugi print za dve identacije

PROBATI TAČKE 2 I 3

Rezultat tačke 2:

Vrednost promenljive broj je 14
Vrednost promenljive broj je 15
Vrednost promenljive broj je 16
Vrednost promenljive broj je 17
Vrednost promenljive broj je 18
Vrednost promenljive broj je 19
Vrednost promenljive broj je 20
Zbir je 210

Rezultat tačke 3:

Vrednost promenljive broj je 14
Zbir je 105
Vrednost promenljive broj je 15
Zbir je 120
Vrednost promenljive broj je 16
Zbir je 136
Vrednost promenljive broj je 17
Zbir je 153
Vrednost promenljive broj je 18
Zbir je 171
Vrednost promenljive broj je 19
Zbir je 190
Vrednost promenljive broj je 20
Zbir je 210

Primer 2: Upotreba prekida iteracije petlje

• Sledeći program sabira sve cele brojeve od 1 do 20 izuzev brojeva 10 i 11:

```
zbir = broj = 0
while broj < 20:
    broj = broj + 1
    if broj == 10 or broj == 11:                #Da li je broj jednak 10 ili 11
        continue                                # DA.Skok na kraj tekuće iteracije
    zbir = zbir + broj
print("Vrednost promenljive broj je", broj)
print("Zbir je", zbir)
```

• Rezultat izvršavanja programa je:

Vrednost promenljive broj je 20

Zbir je 189

PROBATI:

1. and umesto or

```
 zbir = broj = 0
while broj < 20:
    broj = broj + 1
    if broj == 10 and broj == 11:
        continue
    zbir = zbir + broj
print("Vrednost prom.broj je", broj)
print("Zbir je", zbir)
```

Šta se dobija?

Vrednost prom.broj je 20

Zbir je 210

2. bez CONTINUE

```
zbir = broj = 0
while broj < 20:
    broj = broj + 1
    if broj == 10 or broj == 11:
        #continue
        zbir = zbir + broj
print("Vrednost promenljive broj
print("Zbir je", zbir)
```

Šta se dobija?

Vrednost promenljive broj je 20

Zbir je 21

Sabere brojeve 10 i 11 a prolazi k

Mes	Pon	Uto	Sre	Čet	Pet	Sub	Ned	Aktivnosti studenata
IX	23	24	25	26	27	28	29	I nedelja nastave
X	30	1	2	3	4	5	6	II nedelja nastave
	7	8	9	10	11	12	13	III nedelja nastave
	14	15	16	17	18	19	20	IV nedelja nastave
	21	22	23	24	25	26	27	V nedelja nastave / Oktobarski ispitni rok
	28	29	30	31	1	2	3	VI nedelja nastave - kolokvijumska nedelja

Mes	Pon	Uto	Sre	Čet	Pet	Sub
XI	23	24	25	26	27	28
X	30	1	2	3	4	5
	7	8	9	10	11	12
	14	15	16	17	18	19
	21	22	23	24	25	26
	28	29	30	31	1	2
XI	4	5	6	7	8	9
	11	12	13	14	15	16
	18	19	20	21	22	23

3.čas

A bez CONTINUE i IF ?

```
zbir = broj = 0
while broj < 20:
    broj = broj + 1
    #if broj == 10 or broj == 11:
        #continue
    zbir = zbir + broj
print("Vrednost promenljive broj je", broj)
print("Zbir je", zbir)
```

Vrednost promenljive broj je 20

Zbir je 210

Ilustracija: Implementacija programa pomoću naredbe skoka

- Šta je najmanji delilac (ND)?

To je najmanji broj sa kojim se može podeliti zadat broj a da nije broj 1

Koji je ND za 25?

5

Algoritam: počne se sa proverom da li deljiv sa 2 pa ako nije deljiv, proba da li je deljiv sa 3, pa ako nije deljiv proba sa 4 pa sa 5…. A ako je recimo deljiv sa 4 prekida se i to je ND.

Program koji pronalazi najmanji delilac različit od 1 za zadani celi broj n≥2 može se napisati korišćenjem naredbe *break*:

```
n = int(input("Unesi celi broj >= 2: "))    #Unos sa tastature
delilac = 2
while delilac <= n:                         #Ide se do unešenog broja N
   if n % delilac == 0:                     #Da li je deljiv N sa 2
      print("KRAJ: rezultat deljenja sa brojem:",delilac, "  je",n/delilac,"Ostatak=",n% delilac)
      break                                 #DA.Kraj
   print("rezultat deljenja sa brojem:",delilac, " je",int(n/delilac),"Ostatak=",n% delilac)
   delilac = delilac + 1                    #NE. Uvećaj za 1
print("Najmanji delilac broja", n, "je", delilac)
```

Izvršenje:

Unesi celi broj >= 2: 25

rezultat deljenja sa brojem: 2 je 12 Ostatak= 1

rezultat deljenja sa brojem: 3 je 8 Ostatak= 1

rezultat deljenja sa brojem: 4 je 6 Ostatak= 1

KRAJ: rezultat deljenja sa brojem: 5 je 5.0
Ostatak= 0

Najmanji delilac broja 25 je 5

Unesi celi broj >= 2: 25

Najmanji delilac broja 25 je 5

Unesi celi broj >= 2: 49

Najmanji delilac broja 49 je 7

Unesi celi broj >= 2: 999

Najmanji delilac broja 999 je 3

- Isti algoritam može se realizovati *bez* korišćenja naredbe skoka, dodavanjem jedne logičke promenljive (nadjen) i proverom njene vrednosti:

```
n = int(input("Unesi celi broj >= 2: "))
nadjen = False
delilac = 2
while delilac <= n and not nadjen:
    if n % delilac == 0:
        nadjen = True
    else:
        delilac = delilac + 1
print("ND broja", n, "je", delilac)
```

```
Unesi celi broj >= 2: 25
ND broja 25 je 5
```

POJAŠNJENJE KODA:

nadjen = False #Postavlja se logička promenljiva nadjen na False. Ona označava da ND još nije pronađen.

while delilac <= n and not nadjen: #Ova petlja će se ponavljati sve dok je delilac manji ili jednak broju n i dok nije pronađen ND (nadjen je False)

```
n = int(input("Unesi celi broj >= 2: "))
nadjen = 1
delilac = 2
while delilac <= n and nadjen == 1:
    if n % delilac == 0:
        nadjen = 2
    else:
        delilac = delilac + 1
print("Najmanji delilac broja", n, "je", delilac)
```

POJAŠNJENJE KODA:

nadjen = 1 Promenljiva nadjen je inicijalizovana sa vrednošću 1.

Ova promenljiva služi za praćenje da li je pronađen delilac.

while delilac <= n and nadjen == 1: Ova petlja će se izvršavati sve dok je delilac manji ili jednak n i dok je nadjen jednako 1. Dakle, petlja se nastavlja sve dok nije pronađen delilac (nadjen == 2 kada se pronađe).

Probati i sa nadjen = „a“

Praktične preporuke za izradu programa

1. Za uspešno razumevanje problema, dobro je pročitati njegov opis više puta

2. Pre programiranja treba osmisliti algoritam rešenja.

Nakon toga se algoritam može pretvoriti u program

3. Problemi se mogu rešiti na različite načine. Treba ispitati više mogućnosti i usvojiti najbolje rešenje

3. OBRADA IZUZETAKA

1. Pojam izuzetka
2. Upravljanje obradom izuzetaka

3.1 Pojam izuzetka

U svakom programu mogu se pojaviti nepredviđene greške ili izuzeci (*exceptions*), kao što su npr. greške izvršavanja koje nastaju zbog pogrešnog unosa podataka

U takvom slučaju, sistem dojavljuje poruku o grešci i terminira program:

Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'x

- Ovakva poruka je obično nerazumljiva korisniku programa

Recimo u prethodni program za ND uneti slovo umesto broja ili

```
c=input("unesite ceo broj:")
print(c)
d=int(c)
print(d)
```

```
unesite ceo broj:22
22
22
>>>
```

Unećemo slovo umesto broja:

```
========== RESTART: C:/Users/Milan/Documents/test.py ==========
unesite ceo broj:x
x
Traceback (most recent call last):
  File "C:/Users/Milan/Documents/test.py", line 3, in <module>
    d=int(c)
ValueError: invalid literal for int() with base 10: 'x'
>>>
```

Program je stao je ne može int od slova/stringa

3.2 Upravljanje obradom izuzetaka

• Posebna naredba za upravljanje tokom izvršavanja programa omogućava **definisanje grupe naredbi koje će se izvršiti u slučaju pojave nekog izuzetka** oblika:

try:

<naredbe koje mogu da izazovu pojavu izuzetka>

except:

<naredbe za obradu izuzetka>

[else:

<naredbe za slučaj da se izuzetak ne dogodi>] ovo ne mora...

Na taj način program ne mora da okonča u slučaju pojave greške (izuzetka) , već može npr. **dozvoliti korisniku da ispravi uneseni podatak** ili eventualno programski ispraviti grešku

Primer: Unos podataka i upotreba naredbi za upravljanje tokom izvršavanja

```
while True:
    c = input('Unesite celi broj: ')
    print(c)
    try:
        c = int(c)          #Naredba koja može da izazove problem
        break               #sve OK. Dobar unos
    except:                 #Nije dobar unos
        print('Niste uneli celi broj, ponovite.')
```

- Primer izvršavanja programa

Unesite celi broj: y
Niste uneli ispravan celi broj, ponovite.
Unesite celi broj: 2.2
Niste uneli ispravan celi broj, ponovite.
Unesite celi broj: 2

Kako simulirati program za deljenje sa 0

```
b=input('Unesi deljenik: ')
c=input('Unesi delioc: ')
try:
    a=int(b)/int(c)
    print(a)
except :
    print('Greška!')
```

Kako omogućiti petlju

```
while True:
    b=input('Unesi deljenik : ')
    c=input('Unesi delioc: ')
    try:
        a=int(b)/int(c)
        print(a)
        break
    except :
        print('Greška!')
```

U Pythonu, **while True:** pokreće **beskonačnu petlju**. To znači da će se kod unutar te petlje izvršavati neprestano, dok se ne naiđe na uslov koji prekida petlju, obično uz pomoć naredbe **break**

```
Unesi deljenik: 5
Unesi delioc: 0
Greška!
Unesi deljenik: 5
Unesi delioc: 2
2.5
```

Vraća se na unos i delioca i deljenika kako ga vratiti na unos samo delioca?

```
b=input('Unesi deljenik: ')
while True:
    c=input('Unesi delioc: ')
    try:
        a=int(b)/int(c)
        print(a)
        break
    except :
        print('Greška!')
```

```
Unesi deljenik: 23
Unesi delioc: 0
Greška!
Unesi delioc: 4
5.75
```

Znači, izuzetak je uslov koji nastaje tokom izvršavanja programa. To je signal da se dogodilo nešto neočekivano.

Izuzeci imaju svoja, opisna imena. Na primer, ako pokušate da podelite broj sa nulom, dobićete izuzetak :

ZeroDivisionError.

Evo malog izvoda izuzetaka:

KeyError

Raised when a key is not found in a dictionary

MemoryErro

Raised when an operation runs out of memory

NameError

Raised when a variable is not found in local or global scope

Primer za **ZeroDivisionError**: plav tekst je dodat u onaj primer

```
while True:
    b=input('Unesi deljenik: ')
    c=input('Unesi delioc: ')
    try:
        a=int(b)/int(c)
        print(a)
        break
    except ZeroDivisionError:
        print('Greška! - delite sa nulom')
```

ValueError – loše uneta vrednost

```
while True:
    b=input('Unesi deljenik: ')
    c=input('Unesi delioc: ')
    try:
        a=int(b)/int(c)
        print(a)
        break
    except ZeroDivisionError:
        print('Greška! - delite sa nulom')
    except ValueError:
        print('Greška! - uneli ste slovo')
```

Primer:

```
Unesi deljenik: 23
Unesi delioc: 0
Greška! - delite sa nulom
Unesi deljenik: 23
Unesi delioc: a
Greška! - uneli ste slovo
Unesi deljenik: 23
Unesi delioc: 2
11.5
```

4. PRIMERI PROGRAMA

1. Euklidov algoritam
2. Igra pogađanja

Primer: 4.1. Euklidov algoritam

- Algoritam za računanje najmanjeg zajedničkog delioca dva pozitivna cela broja, NZD(*x*,*y*). ŠTA JE NZD?
- Narativni opis:

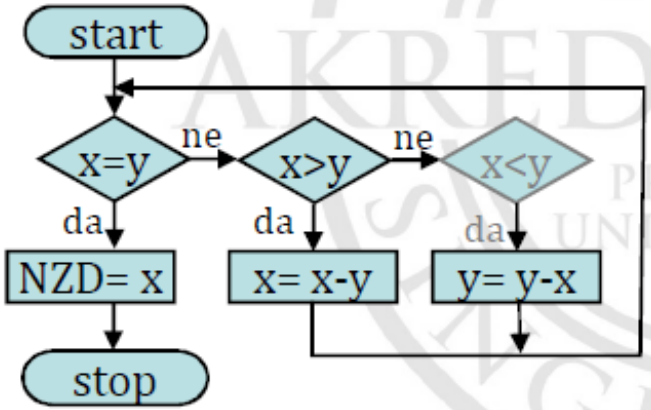
Sve dok su dva broja različita, oduzimaj manji broj od većeg

- Pseudokod (strukturirani tekst):

Sve dok je x≠y,

ako je x>y onda x = x-y,

ako je x<y, onda y = y-x



x=4 y=10
1.y=6
2.y=2
3.x=2
4.NZD=2

Za x=18 i y=15 NZD je 3

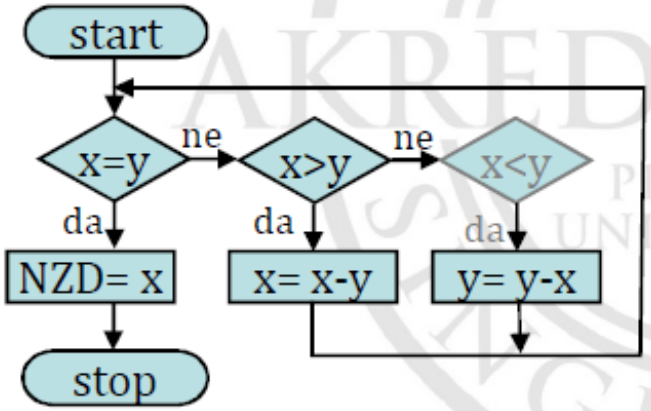
Probati…

4.1 Euklidov algoritam

koja vrsta ponavljanja je odgovarajuća (***sve dok je***)?

Ponavljanje oduzimanjem definisano je prema uslovu pa je prirodno da se u Pythonu koristi while

```
# Euklidov algoritam računanja NZD
x = int(input("Unesi X "))        #unos X
y = int(input("Unesi Y "))        #unos Y
while x != y:                     #Dok je X različito od Y
    if x>y:                   #Da li je X veće od Y
        x = x-y                   #DA. X=X-Y
    else:
        y = y-x                   #NE. Y=Y-X
print("NZD =",x)
```



- Primer izvršavanja:

Unesi X 1116
Unesi Y 90
NZD = 18

4.2 Igra pogađanja

- Igra se sastoji u pogađanju "zamišljenog" (slučajno generisanog) celog broja u nekom intervalu
- Algoritam:

1. Program generiše slučajni celi broj u zadanom intervalu
2. Program traži od korisnika da pogodi koji je broj u pitanju
3. Ako se broj koji unese korisnik *razlikuje* od zamišljenog, korisniku se daje informacija da li je njegov broj *veći* ili *manji* od slučajnog broja
4. Korisnik mora da pogodi broj u konačnom broju pokušaja, inače program prekida rad uz odgovarajuću poruku

Pseudoslučajni broj generiše se funkcijom randrange(...) iz modula random

1. Program generiše slučajni celi broj u opsegu od 0 do zadanog broja

```
import random
MAX_BROJ = int(input("Unesi do kog broja se trazi slucajan broj :"))
broj = random.randint(1,MAX_BROJ+1);
print("slucajan broj je broj:", broj)
```

Rezultat slučajnog broja od 1 do 20 je sada 10
Proverićemo koji broj se dobija kada više puta ponovimo:

Uvek se dobija drugi broj

1. Proširenje dodajemo zeleno

```
import random
MAX_BROJ = int(input("Unesi do kog broja se trazi slucajan broj :"))
broj = random.randint(1,MAX_BROJ+1)
print(“slucajan broj je broj:”, broj)
MAX_POKUSAJA = 5                    # najveći broj pogađanja
# Uvodno obaveštenje
print("Pogađanje slučajnog broja od 1 do", MAX_BROJ, ".")
print("Dozvoljeno je", MAX_POKUSAJA,"pokušaja.")
brojPokusaja = 0 # inicijalizacija brojača
```

2. Program traži od korisnika da pogodi koji je broj u pitanju Ovo dodajemo na prethodno već prošireni kod

```
pokusaj= int(input('\nPogodite slučajni broj: '))
brojPokusaja = brojPokusaja + 1 # brojač pokušaja
print("Broj koji se pogadja je=",broj,"uneti broj=",pokusaj,"Broj pokusaja=",brojPokusaja)
```

Rezultat prvog dela programa:

```
Unesi do kog broja se trazi slucajan broj :20
slucajan broj je broj: 4
Pogađanje slučajnog broja od 1 do 20 .
Dozvoljeno je 5 pokušaja.

Pogodite slučajni broj: 2
Broj koji se pogadja je= 4 uneti broj= 2 Broj pokusaja= 1
```

Drugi deo programa:provera da li smo pogodili broj
ako jesmo štampa i kraj (break)
ako nismo i broj je manji od slučajnog:štampa
ako nismo i broj je veći od slučajnog: štampa
ovaj deo programa dodati ispod I dela programa

```
if pokusaj == broj:                    #Da li smo pogodili broj?
    print("Čestitamo, pogodili ste broj!")   #DA.Štampa
    print("Broj pokušaja:", brojPokusaja)   #Štampa broja pokušaja
    #break                             #Prekid programa jer je pogodjen
elif pokusaj < broj:                   #NE. Da li je naš broj manji ?
    print("Suviše mali broj.") # broj premali    #DA. Štampa..
else:
    print("Suviše veliki broj.") # broj prevelik  #NE.Štampa
```

Uz komandu break ćemo staviti komentar?

Ona važi za petlju

Petlju ćemo naknadno dodati:

```
if pokusaj == broj:                #Da li smo pogodili broj?
    print("Čestitamo, pogodili ste broj!")    #DA.Štampa
    print("Broj pokušaja:", brojPokusaja)    #Štampa broja pokušaja
    #break                                   #Prekid programa jer je pogodjen
  elif pokusaj < broj:                       #NE. Da li je naš broj manji ?
    print("Suviše mali broj.") # broj premali     #DA. Štampa..
  else:
    print("Suviše veliki broj.") # broj prevelik   #NE.Štampa
```

Pre programa sa prethodnog slajda kao i pre sledećih linija :

```
pokusaj= int(input('\nPogodite slučajni broj: '))
brojPokusaja = brojPokusaja + 1 # brojač pokušaja
print("Broj koji se pogadja je=",broj,"uneti broj=",pokusaj,"Broj
    pokusaja=",brojPokusaja)
```

Dodajemo:

```
while True:
```

I šta još?

Skidamo komentar sa komande BREAK

```
while True:
   pokusaj= int(input('Pogodite slučajni broj: '))
   brojPokusaja = brojPokusaja + 1 # brojač pokušaja
   print("Broj koji se pogadja je=",broj,"uneti broj=",pokusaj,"Broj
     pokusaja=",brojPokusaja)
if pokusaj == broj:                    #Da li smo pogodili broj?
    print("Čestitamo, pogodili ste broj!")   #DA.Štampa
    print("Broj pokušaja:", brojPokusaja)   #Štampa broja pokušaja
    break                          #Prekid programa jer je pogodjen
elif pokusaj < broj:                   #NE. Da li je naš broj manji ?
        print("Suviše mali broj.") # broj premali #DA. Štampa..
else:
        print("Suviše veliki broj.") # broj prevelik      #NE.Štampa
```

Zašto program ne radi?

Identacija

Posle petlje while svaki red uvući za po 4 znaka

Rezultat II dela programa:

Program za pogađanje slučajnog broja između 1 i 20 .
Dozvoljeno je 5 pokušaja.

Pogodite slučajni broj: 8
Broj koji se pogadja je= 11 uneti broj= 8 Broj pokusaja= 1
Suviše mali broj.

Pogodite slučajni broj: 9
Broj koji se pogadja je= 11 uneti broj= 9 Broj pokusaja= 2
Suviše mali broj.

Pogodite slučajni broj: 19
Broj koji se pogadja je= 11 uneti broj= 19 Broj pokusaja= 3
Suviše veliki broj.

Pogodite slučajni broj: 11
Broj koji se pogadja je= 11 uneti broj= 11 Broj pokusaja= 4
Čestitamo, pogodili ste broj!
Broj pokušaja: 4
>>>

Šta je ostalo da se uradi?

Kontrola da li smo pogadjali više od 5 puta

Dodaje se provera

```
if brojPokusaja == MAX_POKUSAJA :    # da li pogadjali 5 puta?
        print("Niste pogodili.. Hvala na ucestvovanju, Slucajan broj je bio:", broj)
        break
```

```
# Program za pogađanje slučajnog broja import random
MAX_POKUSAJA = 5 # najveći broj pogađanja
MAX_BROJ = 20 # najveći slučajni broj
# Uvodno obaveštenje
print("Program za pogađanje slučajnog broja između 1 i", MAX_BROJ, ".")
print("Dozvoljeno je", MAX_POKUSAJA,"pokušaja.")
# Generisanje slučajnog broja
import random
broj = random.randint(1,MAX_BROJ+1)
brojPokusaja = 0 # inicijalizacija brojača
while True: # beskonačna petlja
# Unos pogađanja
   pokusaj= int(input('\nPogodite slučajni broj: '))
   brojPokusaja = brojPokusaja + 1 # brojač pokušaja
   print("Broj koji se pogadja je=",broj,"uneti broj=",pokusaj,"Broj pokusaja=",brojPokusaja)
   # Ako je broj pogođen, čestitka i prekid petlje
   if pokusaj == broj:                    #Da li smo pogodili broj?
      print("Čestitamo, pogodili ste broj!") #DA.Štampa
      print("Broj pokušaja:", brojPokusaja) #Štampa broja pokušaja
      break                               #Prekid programa jer je pogodjen
   elif pokusaj < broj:                   #NE. Da li je nađ broj manji ?
      print("Suviše mali broj.") # broj premali      #DA. Štampa..
   else:
      print("Suviše veliki broj.") # broj prevelik     #NE.Štampa
   if brojPokusaja == MAX_POKUSAJA:     # da li pogadjali 5 puta?
      print("Niste pogodili.. Hvala na ucestvovanju, Slucajan broj je bio:", broj)
      break
```

Konacan program:

Underline: I deo
Italic: II deo
Bold: III deo

Rezultat programa:

```
Program za pogađanje slučajnog broja između 1 i 20 .
Dozvoljeno je 5 pokušaja.

Pogodite slučajni broj: 11
Broj koji se pogadja je= 6 uneti broj= 11 Broj pokusaja= 1
Suviše veliki broj.

Pogodite slučajni broj: 9
Broj koji se pogadja je= 6 uneti broj= 9 Broj pokusaja= 2
Suviše veliki broj.

Pogodite slučajni broj: 7
Broj koji se pogadja je= 6 uneti broj= 7 Broj pokusaja= 3
Suviše veliki broj.

Pogodite slučajni broj: 6
Broj koji se pogadja je= 6 uneti broj= 6 Broj pokusaja= 4
Čestitamo, pogodili ste broj!
Broj pokušaja: 4
>>>
```