Operatori

Operatori nam omogućavaju razne operacije poput kalkulacije brojeva, spajanja stringova, pridruživanja vrednosti promenljivama, upoređivanja vrednosti i slično.

U programiranju, operator je simbol sa specijalnim značenjem koji se koristi za određenu operaciju. Oni se ponašaju slično funkcijama u smislu da uzimaju ulazni parametar i daju izlazni rezultat, ali se po sintaksi dosta razlikuju od njih. Na primer, u izrazu 1 + 1, operator (+) dodaje levom broju desni broj.

Python podržava operatore iz sledećih kategorija:

- aritmetički operatori;
- relacioni operatori (operatori upoređivanja);
- logički operatori;
- operatori pridruživanja;
- operatori nad bitovima (bitwise operatori);
- ternarni operatori (operatori uslovljavanja).

Aritmetički operatori

Aritmetički operatori koje Python podržava su:

```
+ - * / // % **
```

Aritmetički operatori uzimaju numeričke vrednosti kao operande i vraćaju jednu numeričku vrednost. Na primer, znak plus (+) dodaje levom broju broj sa desne strane operatora. Na sledećoj slici su rezultati korišćenja svakog od ovih operatora:

Primer aritmetičkih operatora		
Kod	**Rezultat**	**Opis**
print(100 + 30)	130	sabiranje dva broja
print(100 - 30)	70	oduzimanje dva broja
print(100 * 30)	3000	množenje dva broja
print(100 / 30)	3.33333333333335	deljenje dva broja sa ostatkom
print(100 // 30)	3	deljenje dva broja bez ostatka (zaokružiće se manji broj)
print(38 % 14)	10	ostatak pri deljenju (moduo operator)
print(2 ** 3)	8	stepenovanje, gde je desni operand eksponent

Tabela 8.1. Aritmetički operatori

Relacioni operatori (operatori upoređivanja)

Python podržava sledeće relacione operatore:

```
== != > < >= <=
```

Copyright © Link group

Ovi operatori nam omogućavaju upoređivanje dva objekta. Ova operacija vraća logičke vrednosti True ili False. Objekti koji se upoređuju ne moraju biti istog tipa. U sledećoj tabeli vidimo i primenu operatora (==) i rezultate njihove primene. Dakle, ako su oba operanda potpuno jednaka, vratiće se vrednost True; u suprotnom, vratiće se vrednost False.

Primeri relacionih operatora		
Kod	**Rezultat**	**Opis**

`print(1 == 1)`
`print(1 == 2)`
`print(1 == 0.999)`
`print(True == None)`
`print(True == True)`
	`True`
	`False`
	`False`
	`False`
	`True`
		Operator jednakosti. Upoređuju se dve vrednosti. Samo ako su iste, dakle istog tipa i istih vrednosti, rezultat će biti True. Primera radi, 1 nije isto što i 0.99, pre svega jer je reč o dva različita tipa podataka.

`print(1 != 1)` `print(1 != 2)`	`False` `True`	Operator nejednakosti. Ako date vrednosti nisu jednake, vratiće se vrednost True. Dakle, ovaj operator funkcioniše suprotno od operatora jednakosti.

`print(1 > 1)`
`print(1 > 2)`
`print(1 < 0.999)`
`print(True < True)`
`print(True > None)`
	`False`
	`False`
	`False`
	`False`
	`Type Error`
		Operatori veće od (>) i manje od (<). Ako je levi operand veći od desnog (>), odnosno manji od desnog (<), vratiće nam se True vrednost u program.

`print(1 >= 1)`
`print(1 >= 2)`
`print(1 <= 0.999)`
`print(True <= True)`
`print(True >= None)`
	`True`
	`False`
	`False`
	`True`
	`Type Error`
		Operatori veće ili jednako (>=) i manje ili jednako (<=). Ako je levi operand veći ili jednak od desnog (>=), odnosno manji ili jednak od desnog (<=), vratiće nam se True vrednost u program.

Tabela 8.2. Relacioni operatori

Upišite kod u radno okruženje i proverite rezultat:

Radno okruženje

Operacije upoređivanja imaju niži prioritet u odnosu na aritmetičke operacije ili operatore sa bitovima.

Vežba

Napisati program koji deli cele brojeve i generiše celobrojni rezultat (bez decimala).

Rešenje

```
a = 4
b = 3
c = a // b
print(c)
```

Objašnjenje:

Postoji više načina za rešavanje ovog problema. Jedan od njih je eksplicitna konverzija tipova. U ovom slučaju smo iskoristil operator deljenja bez ostatka (//).

Copyright © Link group

Logički operatori

Python podržava sledeće logičke operatore:

```
and or not
```

Logički operatori se uglavnom koriste u uslovnim izrazima. Na primer, vratiće `True` ili `False` vrednosti u zavisnosti od vrednosti operanada. U narednom primeru, logički operator `and` testira da li oba operanda imaju istu vrednost:

Primer logičkih operatora		
Kod	**Rezultat**	**Opis**

`print(1==1 and False==False)`
 	
`print(True==True and 1==2)`
	`True`
	`False`
		Logičko i: Ako su oba operanda ili uslova tačna, vratiće se vrednost True. Ako je samo jedan od tih uslova netačan, vratiće se vrednost False.

`print(1==2 or False==True)`
 	
`print(True==True or 1==2)`
	`False`
	`True`
		Logičko ili: Ako je samo jedan od uslova tačan, vratiće se vrednost True. U suprotnom, tj. samo ako su oba uslova netačna, vratiće se netačna vrednost.

`print(not False)`
 	
`print(not None)`
 	
`print(not 5)`
	`True`
	`True`
	`False`
		Operator not (negacija) menja logičku vrednost prosleđenog operanda u suprotnu od njegove. Ovom operatoru se prosleđuje samo jedan operand, što se razlikuje od ostalih operatora, gde su potrebna dva operanda.

Tabela 8.3. Logički operatori

Operatori pridruživanja

Operatori pridruživanja u Pythonu su:

```
= += -= *= /= %= **= //=
```

Koriste se za pridruživanje vrednosti promenljivama. Osnovni operator pridruživanja je znak jednako (=), koji dodeljuje vrednost sa svoje desne strane levoj strani.

Primer operatora pridruživanja (dodele)		
Primer	**Operator**	**Opis**
a = 1 + 2 print(a)	=	Dodaje vrednost desne strane levom operandu. U ovom slučaju promenljiva a će imati vrednost 3.
a = 10 print(a) a += 10 print(a) b=10 b-= 5	+=, -=	Operator koji sabira (+=) ili oduzima (-=) desni operand sa levim / od levog i rezultat pridružuje levom operandu. Nakon upotrebe += operatora nad promeljivom a=10, rezultat promenljive a je 20, a nakon upotrebe -= nad promenljivom b=10 rezultat je 5. Isti efekat bismo postigli kada bismo napisali a = a+10 ili, u

Copyright © Link group

print(b)		drugom slučaju, b= b- 5.
a = 10 print(a) a *= 2 print(a) b=10 b**= 2 print(b)	*=, **=	Operator (*=) množi desni operand sa levim i vrednost smešta u levi. Operator (**=) stepenuje levi operand desnim i rezultat smešta u levi operand. Nakon upotrebe *= operatora nad promenljivom a=10, rezultat promenljive je 20, a nakon upotrebe **= nad promenljivom b=10 vrednost je 100. Isti efekat bismo postigli kada bismo napisali a = a *2 ili, u drugom slučaju b= b **2.
a = 10 print(a) a /= 2 print(a) b= 10 b//= 2 print(b)	/=, //=	Operator (/=) deli levi operand desnim i levom operandu dodeljuje rezultat. Ako je levi operand tipa int, a rezultat ovakvog deljenja ima ostatak, levi operand će ostati int. Operator (//=) deli levi operand desnim i i upisuje rezultat levom operandu. Pošto // deljenje zaokružuje na manji broj, to je i ovde slučaj, pa rezultat izraza a //= 2 iznosi 5. Isto bismo postigli kada bismo napisali a = a // 2
a = 10 a %= 2 print(a)	%=	Operator (%=) deli levi operand desnim i ostatak tog deljenja smešta u levi operand. Isti efekat bismo postigli kada bismo napisali a = a % 2.

Tabela 8.4. Operatori pridruživanja

Upišite kod u radno okruženje i proverite rezultat:

Radno okruženje

Pitanje

Šta će izraz print(`False == 0`) ispisati?

- Izbaciće grešku
- False
- **True**

Objašnjenje:

Daće vrednost `True`. U Pythonu, logički tipovi kao što su `True/False` su potklase klase `int()`, dakle potklase celog broja, pa tako `False` zapravo ima vrednost 0 dok `True` ima vrednost 1.

Operatori nad bitovima (binarni operatori)

Da bi se razumele operacije nad bitovima, treba reći šta su to zapravo bitovi.

Copyright © Link group

U računarstvu, sve se vrti oko nula i jedinica, tačnije binarnog koda. Ove dve vrednosti su jedine koje procesor može razumeti i na tom principu je bazirano današnje računarstvo. Tako, jedan bit može biti ili 0 ili 1 (False ili True stanja). Na ovaj način, nulama i jedinicama, predstavljeni su i svi podaci u računaru, od brojeva i stringova do filmova, muzike itd. Kako sa jednim bitom možemo predstaviti samo dva broja, sa dva bita možemo predstaviti četiri broja – od 0 do 3 (00, 01, 10, 11). Kako ne bismo ručno računali koliko vrednosti sa koliko bitova možemo napisati, postoji formula koja glasi 2^{broj_bitova} . Ako imamo tri bita, koristeći formulu 2^3, dobićemo da sa ta tri bita možemo ispisati 8 cifara (od 0 do 7). Ovo nas dovodi do sledeće, veće jedinice mere i to je bajt. Po definiciji, jedan bajt sadrži osam bitova (00000000). Sa osam bitova, moguće je predstaviti 256 brojeva ili drugih vrednosti. Za prebacivanje vrednosti u binarni kod, možemo koristiti ugrađenu funkciju bin(). A za pretvaranje binarnog koda u dekadni, možemo iskoristiti naredbu `int(bin(), 2)`. Parametar 2 funkciji int() dodeljujemo jer je baza binarnog sistema dva (baratamo samo sa dve cifre). Na primer, baza dekadnog sistema je 10 (od 0 do 9).

Operatori nad bitovima su:

```
| ^ & << >> ~
```

Oni obavljaju operacije bit po bit, što se najbolje može videti na primeru:

```
i = 4, j = 8
```

Ako su njihove binarne vrednosti:

```
i = 0100
j = 1000
```

onda će rezultati binarnih operacija biti:

Operator	Ime operatora	Pojašnjenje	Primer
&	Binarno „i"	Ako su oba bita 1 – daje 1, u suprotnom 0.	Izraz: i & j Rezultat: 0 (0000)
|	Binarno „ili"	Ako je jedan od bitova 1 – daje 1, u suprotnom 0.	Izraz: i | j Rezultat: 12 (1100)
^	Binarno ekskluzivno „ili"	Ako su oba bita ista, rezultat je 0, u suprotnom 1.	Izraz: i ^ j Rezultat: 12 (1100)
~	Komplement operator (binarno ne, negacija)	Ako je bit 1, pretvara ga u 0 i suprotno – ako je bit 0, pretvara ga u 1.	Izraz: ~i Rezultat: -5
<<	Binarno pomeranje ulevo	Levi operand se pomera (bit po bit) ulevo za onoliko koliko iznosi desni operand.	Izraz: i << 2 Rezultat: 16 (10000)
>>	Binarno pomeranje udesno	Levi operand se pomera (bit po bit) udesno za onoliko koliko iznosi desni operand.	Izraz: i >> 1 Rezultat: 2 (0010)

Tabela 8.5. Rezultati binarnih operacija na primeru i i j

Copyright © Link group

Analiza birnarne i operacije

Neka su sada promenljive i =5 i j = 7, a u binarnom zapisu zapravo: i = 101, j = 111.

Nakon r = i & j operacije dobijamo rezultat 5 smešten u promenljivu r. Ako znamo da & operator vraća 1 ako su obe vrednosti 1, a 0 ako su obe vrednosti 0, u pozadini ovog izvršavanja se dešava sledeće:

1. Korak, potpisujemo ove dve vrednosti i računamo zdesna nalevo:

   i:10**1**
   j:11**1**
   r:...**1**

   U prvom koraku, na mestima prvih bitova u oba broja su jedinice i tako & vraća 1.

2. Korak:

   i:1**0**1
   j:1**1**1
   r:..01

   U drugom koraku, na mestu drugog bita prvog broja je 0, a na mestu drugog bita drugog broja je 1. U ovom slučaju naš operator vraća 0.

3. Korak:

   i:**1**01
   j:**1**11
   r:101

   U trećem koraku, na mestima prvih bitova u oba broja su jedinice i tako nam operator & vraća 1.

Rezultat ovog računanja je u binarnoj reprezentaciji i to: 101. Kada ovaj broj iz binarnog koda pretvorimo u dekadni naredbom int(r,2), dobićemo broj 5.

Opet, važno je napomenuti da operatori ove grupe funkcionišu po principu bit po bit datog broja ili operanda.

Ternarni operatori (operatori uslovljavanja)

Većina programskih jezika podržava korišćenje ternarnih operatora, koji nam omogućavaju da definišemo uslovni izraz u jednoj liniji. U Pythonu, definisanje takvog izraza se vrši na sledeći način:

```
n if M else p
```

Ovaj izraz će vratiti `True` vrednost, što znači da će prvo ispitati uslov M. Ako je tačan, vratiće se vrednost n, a u suprotnom će se vratiti vrednost p. Primer upotrebe ovih operatora se vidi u sledećoj tabeli:

Copyright © Link group

Primer ternarne operacije:

Radno okruženje

```
a=5
print("Positive number." if a >= 0 else "Negative number.")
```

Objašnjenje:

Pošto je broj a veći od nule odn. uslov a>=0 je tačan ispisaće se prvi string: `"Positive number."`

Prioriteti i asocijativnost operatora

Sada, nakon što smo se upoznali sa svim operatorima u Pythonu, treba napomenuti koji operatori imaju prvenstvo. Kao i u matematici, i u programskim jezicima je prioritet operatora veoma bitan i direktno utiče na željeni rezultat. Na primer, u Pythonu, kao i u matematici prvenstvo ima množenje nad sabiranjem. Takođe, kao i u matematici, prioritete možemo menjati tako što izraze smeštamo između običnih zagrada (). Prioriteti operatora su predstavljeni u tabeli 8.7, od najvišeg prioriteta ka najnižem.

Prioritet operatora	
Kod	**Rezultat**
()	Zagrade, odnosno grupisanje izraza
**	Eksponencijalni
~	Binarno ne
*, /, //, %	Množenje, deljenje, deljenje bez ostatka, deljenje po modulu
-, +	Oduzimanje, sabiranje
<<, >>	Binarno pomeranje ulevo i udesno
&	Binarno i
|, ^	Binarno, ili, binarno ekskluzivno ili
==, !=, >, >=, <, <=	Operatori upoređivanja (relacioni)
= += -= *= /= %= **= //=	Operatori pridruživanja (dodele)
not, or, and	Logički operatori

Tabela 8.6. Prioritet operatora

Sada kada znamo prioritet operatora, treba pomenuti slučaj kada se nađu operatori istog nivoa bez zagrada. Ova osobina operatora se naziva asocijativnost. U programskim jezicima se ovo rešava na dva načina: ili se izraz gleda sa leve strane na desnu ili obrnuto. Recimo da imamo primer a*b/c. I množenje i deljenje su operatori istog prioriteta i u ovom slučaju će se izvršiti prvo množenje pa deljenje, upravo zbog asocijativnosti. U Pythonu, sve grupe operatora osim dve imaju asocijativnost s leve strane ka desnoj. Dve grupe čija je asocijativnost zdesne nalevo su eksponencijalni operator (**) i operatori dodele.

Primeri prioriteta i asocijativnosti operatora

Copyright © Link group

Radno okruženje

```
#Neka su date sledece promenljive
nbr_1, nbr_2, nbr_3, nbr_4 = 1, 2, 3, 4

# Kako zagrade imaju najviši prioritet,
# sabraće se brojevi 1 i 2 pa tek onda izvršiti množenje sa trećim
print ((nbr_1 + nbr_2) * nbr_3)

# Kako eksponencijalni operator ** ima viši prioritet,
# prvo će se izvršiti stepenovanje broja 1 brojem 2
# pa izvršiti oduzimanje sa brojem 3.
print (nbr_1 ** nbr_2 - nbr_3)

# Operator negacije (~) ima viši prioritet, pa ćemo tako
# dobiti inverznu vrednost promenljive nbr_1 pa oduzeti od
# promenljive nbr_2
print (~nbr_1 - nbr_2)

# Kako je asocijativnost eksponencijalnog operatora
# s desne strane na levu, prvo će se izvršiti stepenovanje promenljive
# nbr_2 promenljivom nbr_3, pa će taj rezultat stepenovati
# promenljivu nbr_1. i tek na kraju će se taj rezultat dodeliti
# promenljivoj nbr_4, jer asocijativnost operatora dodele (=)
# takođe ide sa desne ka levoj strani.
nbr_4 = nbr_1 ** nbr_2 ** nbr_3
print(nbr_4)

# U ovom slučaju operatori su istog prioriteta.
# Pošto je asocijativnost operatora dodele (=) sa desne strane ka levoj,
# vrednost promenljive nbr_1 će se dodeliti nbr_2, a ta, nova vrednost
# promenljive nbr_2 će se dodeliti promenljivoj nbr_3
nbr_3 = nbr_2 = nbr_1
print(nbr_3)
```

Kao što vidite, rezultat je sledeći:

```
9
-2
-4
1
1
```

Rezime

- Python podržava sledeće operatore: aritmetičke, relacione, logičke, operatore pridruživanja, operatore nad bitovima i ternarne operatore.
- Aritmetički operatori (+,-,*, /, //, %, **) uzimaju dve vrednosti kao operande i bave se osnovnim aritmetičkim operacijama.

Copyright © Link group

- Relacioni operatori (operatori upoređivanja) (==, !=, >, <, >=, <=) omogućavaju nam upoređivanje dva operanda. Nakon korišćenja ove operacije u program nam se vraća ili True ili False vrednost.
- Logički operatori (and, or, not) koriste se u uslovnim izrazima za kombinaciju više uslova i takođe vraćaju True ili False.
- Operatori pridruživanja (=, =+, -=, *=, -=, %=, **=, //=) koriste se za pridruživanje vrednosti desnog operanda levom.
- Operatori nad bitovima (|, ^, &, <<, >>, ~) zapravo obavljaju operacije bit po bit nad zadatim operandima.
- Ternarni operatori (operatori uslovljavanja) omogućavaju nam definisanje uslovnih izraza u jednoj liniji koristeći if-else naredbe.
- Operatori u Pythonu imaju svoje prioritete.
- Ako se u izrazu nađu samo operatori istog nivoa prioriteta, redosled izvršenja će se gledati po njihovoj asocijativnosti, koja može biti ili sa leve strane ka desnoj ili sa desne ka levoj.

Copyright © Link group