Operátory

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1

Základní pojmy

- Operace
  - je to, co se provede, např. operace sčítání sečte dvě čísla
- Operand
  - objekty, s nimiž se operace provádějí
- Operátor
  - znak (nebo skupina znaků), které oznamují, jaká operace se má v daném místě provést

- Příklad

```
10 + a
```

Arita operátorů

- označuje, kolik operandů se při provádění dané operace zpracovává
- máme:
  - nulární
    - nemají žádný operand,
    - konstanty,
  - unární
    - mají jen jeden operand,
    - např.: minus, negace nebo závorky,
  - binární
    - mají dva operandy,
    - většina operátorů: aritmetické, porovnání, ...
  - ternární
    - mají tři operátory,
    - podmínkový operátor, řezy
  - kvadrální operátor
    - potřebuje čtyři operandy
    - řezy

Priorita operátorů

- udává pořadí, ve kterém se operace provádějí
- pořadí lze pozměnit použitím kulatých závorek (závorky je ovšem dobré používat i tam, kde nejsou nutné, protože zvyšují přehlednost kódu)

3*6+12/4-2	(3*6)+12/(4-2)
3*6+12/4-2	(3*6)+12/(4-2)
18+12/4-2	18+12/(4-2)
18+3-2	18+12/2
21-2	18+6
19	24

Asociativita binárních operátorů

- oznamuje, jak překladač postupuje v okamžiku, kdy za sebou následuje několik stejných (nebo ekvivalentních) operátorů, tedy jak jednotlivé operandy sdružuje do dvojic,
- udává „směr“, ve kterém se vyhodnocují operace se stejnou prioritou (zleva nebo zprava)
- lze opět ovlivnit použitím závorek
- příklad:
  - operátor # je asociovaný zleva, potom výraz:

    ```
    a # b # c # d
    ```

    se vyhodnocuje stejně, jako bychom jej přepsali do tvaru

    ```
    ((a # b) # c) # d
    ```

  - operátor # je asociovaný zprava, potom výraz:

    ```
    a # b # c # d
    ```

    se vyhodnocuje stejně, jako bychom jej přepsali do tvaru

    ```
    a # (b # (c # d))
    ```

Aritmetické operátory

Aritmetické operátory

- umožňují kombinovat jednoduché výrazy a tím vytvářet výrazy složitější,
  - `-` mění znaménko operandu (unární operátor)
  - `+` součet dvou operandů
  - `-` rozdíl dvou operandů
  - `*` násobení dvou operandů
  - `/` dělení dvou operandů
  - `%` operace modulo (zbytek po celočíselném dělení prvního operandu druhým)
  - `**` mocnina
  - `//` výsledek dělení prvního operandu druhým zaokrouhlený směrem dolů (floor division)

Příklad

- Vyzkoušejme si nové operátory.

```
>>> 5-2
3
>>> 5*2
10
>>> 5//2
2
>>> 5%2
1
```

- Všimneme si, že pomocí operátoru - můžeme vytvořit záporné číslo.

```
>>> 0-1
-1
```

Příklad

- U operátorů // a % dochází k chybě dělení nulou v případě, že se druhý podvýraz vyhodnotí na nulu.

```
>>> 4%0
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
>>> 4//0
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
```

- Zapište výrazy 1+2*3 a (1+2)*3:

```
>>> 1+2*3
7
>>> (1+2)*3
9
>>>
```

Příklad

- Napište výraz:

```
>>> 5**2
25
>>>
```

- Napište výraz:

```
>>> -1**2
-1
```

- ** má vyšší prioritu než -, proto se výraz vyhodnotí jako tento: -1(1**2). Pokud chceme umocnit -1, musíme použít operátor závorek:

```
>>> (-1)**2
1
>>>
```

- Napište výraz:

```
>>> 2**2**3
256
```

- ** je asociativní zprava, takže se první vyhodnotí pravý operátor **. Pokud chceme první vyhodnotit levý operátor, musíme použít závorky:

```
>>> (2**2)**3
64
>>>
```

Operátory porovnání a logické operátory

Logické hodnoty

- Součástí jazyka Python je Booleovský datový typ, který reprezentuje
- logickou pravdu (True) a logickou nepravdu (False).

- Příklad:

```
a = True
b = False
```

- Ve své podstatě se jedná o celočíselný datový typ. Hodnota True je v jazyce Python chápána jako hodnota 1 a hodnota False jako hodnota 0.

Logické operátory

- logický součin, logický součet, negace

```
a and b, a or b, not a
```

a	b	a and b
0	0	0
1	0	0
0	1	0
1	1	1

a	b	a or b
0	0	0
1	0	1
0	1	1
1	1	1

a	not a
0	1
1	0

Příklad

- Napište výrazy:

```
>>> True or False
True
>>> True and False
False
>>> not True
False
```

- Nejvyšší prioritu má not, poté and a nakonec or. Proto platí:

```
>>> not False and not True or True
True
>>> ((not False) and (not True)) or True
True
```

Operátory porovnání

- (je menší, je větší, je menší nebo rovno,
  je větší nebo rovno, rovná se, nerovná se)

```
a<1, 2>b, 2<=3, a>=b, a==1, b!=2
```

- výsledkem logická hodnota True nebo False.
- Příklad:

```
>>> a=1
>>> b=2
>>> c=3
>>> a<b
True
>>> c==a
False
>>> b!=c
True
>>> a<b<c
True
```

Příklad

- Napište tyto výrazy:

```
>>> 1 < 0
False
>>> 2 < 2
False
>>> 2 <= 2
True
>>> 3 > 2
True
>>> 4 >= 5
False
```

- Pozor na operátor ==. Neplést s operací =.

```
>>> 10==10
True
>>> 10=10
  File "<stdin>", line 1
SyntaxError: can't assign to literal
>>>
```

Líné vyhodnocování

- V logických výrazech se vyhodnocuje pouze část výrazu (podle asociativity) nutná pro získání výsledku.
- Příklad:

```
>>> a=0
>>> a == 0 or 20 % a == 0
True
>>> 20 % a == 0 or a == 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero
>>>
```

Rozhodovací příkaz

- Máme rozhodnout, zda číslo a je menší než číslo b:

```
# vstup
a = 1
b = 2
# porovnání
result = a < b
# výstup
print(result)
```

Další operátory

Operátor přiřazení (mroží operátor)

- operátor přiřazení

```
p := výraz
```

- do proměnné p se přiřadí hodnota výrazu
- výsledkem operace je hodnota výrazu
- Příklad:

```
>>> (a:=(b:=(c:=10)))
10
>>> a,b,c
```

- totéž lze udělat i pomocí speciálního tvaru příkazu přiřazení

```
(10, 10, 10)
>>> a=b=c=10
>>> a,b,c
(10, 10, 10)
```

Operátor přiřazení (mroží operátor)

- Příklad:

```
>>> (a:=10+(b:=10))
20
>>> a,b
(20, 10)
```

- při použití příkazu přiřazení se objeví chyba:

```
>>> a=10+(b=10)
  File "<stdin>", line 1
    a=10+(b=10)
```

Podmíněný výraz

- podmíněný výraz má tvar

```
výraz1 if podmínka else výraz2
```

  - pokud je podmínka splněna, pak se vyhodnotí výraz1, v opačném případě výraz2
- příklad:

```
>>> a=10
>>> b=20
>>> vysledek="Číslo A je menší než číslo B" if a<b else "Číslo A není menší než číslo B."
>>> vysledek
'Číslo A je menší než číslo B'
```

Standardní vstup

Vstup z klávesnice

- Pro čtení z klávesnice slouží funkce

```
input(text)
```

  - text – text, který se zobrazí před čekáním na čtení z klávesnice.
- Příklad:

```
jmeno=input("Zadej své jméno:")
print(f"Tvoje jméno je {jmeno}.")
```

- Z klávesnice se vždy načítá řetězec!
- Příklad:

```
cislo=input("Zadej číslo:")
print(f"Druhá mocnina čísla je {cislo**2}.")
```

  - chyba -> Musíme vstup převést na jiný datový typ!

Převod na jiný datový typ

- Používají se funkce

```
str(výraz)
int(výraz)
float(výraz)
```

- Příklad:

```
cislo=input("Zadej číslo:")
cislo=int(cislo)
print(f"Druhá mocnina čísla je {cislo**2}.")
```