Logické operace a operace s bity

Logické operace

Každý výraz v jazyce Python může být zároveň i logickým výrazem. Jazyk Python nemá žádný typ určený pro logické proměnné. Nulová hodnota typu celé číslo je nepravda (**false**), jakákoli jiná je pravda (**true**). Ale i výrazy jiných typů mohou být vyhodnoceny jako logické výrazy.

Relace

V jazyce Python jsou **relační operátory:** **<** (menší než), **>** (větší než), **==** (rovno), **!=** (nerovno – lze psát i <>, ale nedoporučuje se to); dále **<=** (menší nebo rovno) a **>=** (větší nebo rovno).

Podobně jako v matematice lze psát `a<b<c`, což znamená `a<b and b<c` (v prvním případě se b vyhodnotí pouze jednou, v druhém dvakrát). Python dovoluje i výrazy jako `a<b>c`.

Čísla před porovnáním se nejprve převedou na společný typ. Řetězce se porovnávají lexikograficky (tedy podobně jako ve slovníku).

Příklady relací (tučně jsou označeny výsledky):

```
>>> 4==5
0
>>> 3==3
1
>>> 2<5
1
>>> 7>-4
1
>>> 3!=-2
1
```

Logické operátory

Mezi logické operátory patří **`and`** (logický součin), **`or`** (logický součet), **`not`** (negace) a dále **`in`** a **`not in`** (test výskytu či nevýskytu v sekvenci).

Operátory **`and`** a **`or`** pracují odlišně od jiných jazyků. Jejich výsledek nemusí být 1 nebo 0, ale může mít obecně jakoukoli hodnotu a typ.

- Při vyhodnocování **`x and y`** se nejprve vyhodnotí x. Je-li výraz x false, je vrácena jeho hodnota, jinak se vyhodnotí výraz y a vrátí se výsledek výrazu y:

  ```
  >>> 2 and 3
  3
  >>> 0 and -1
  0
  ```

- Při vyhodnocování výrazu **`x or y`** se nejprve vyhodnotí výraz x. Je-li jeho hodnota true, je vrácena jeho hodnota, jinak se vyhodnotí výraz y a vrátí se výsledek výrazu y:

  ```
  >>> 4 or 2
  4
  >>> 0 or -6
  -6
  ```

Toto poněkud zvláštní chování lze využít například pro dosazení implicitní hodnoty jako v následujícím příkladu.

```
>>> vstup = raw_input('Zadej text: ')                    (1)
Zadej text: abcd
>>> print 'Zadali jste ' + (vstup or 'prazdny retezec')
Zadali jste abcd
>>> vstup = raw_input('Zadej text: ')                    (2)
Zadej text:
>>> print 'Zadali jste ' + (vstup or 'prazdny retezec')
Zadali jste prazdny retezec
```

V případě (1) byl zadán vstupní text řetězec, proto počítač tento řetězec opíše. V případě (2) nebyl vstupní řetězec zadán, proto počítač vypsal, že byl zadán prázdný řetězec.

Operátory, které testují zda objekt na levé straně je či není prvkem sekvence napravo jsou operátory **`in`** a **`not in`.**

Příklady:

```
>>> 'w' in 'windows'
1
>>> 0 in (-1, 1, 2, 3)
0
```

Operace s bity

Operace s bity se používají především v řídicích aplikacích, při práci se zvuky a v počítačové grafice. V následujícím textu je vedle implementace operací s bity v jazyku Python popsána i nezbytná teorie. Pro jednoduchost budou operace vysvětlovány na operandech v délce jednoho bajtu.

Logický součet (OR)

Pravdivostní tabulka:

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Příklad:

0000 1011 (11)
0000 0101 (5)
0000 1111 (15)

Operátorem v jazyku Python je **`|`.**

Příklady v jazyku Python:

```
>>> 0x3 | 0x2
3
>>> 0x2 | 0x1
3
```

Logický součin (AND)

Pravdivostní tabulka:

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Příklad:

0000 1011 (11)
0000 0101 (5)
0000 0001 (1)

Operátorem v jazyku Python je **&.**

Příklady v jazyku Python:

```
>> 0x1 & 0x2
0
>>> 0x7 & 0x5
5
```

Exkluzivní nebo (XOR)

Pravdivostní tabulka:

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Příklad:

0000 1011 (11)
0000 0101 (5)
0000 1110 (14)

Operátorem v jazyku Python je **^.**

Příklady v jazyku Python:

```
>>> 2 ^ 3
1
>>> -1 ^ 0
-1
```

Negace (NOT)

Pravdivostní tabulka:

X	NOT X
0	1
1	0

Příklad:

0000 0101 (5)
1111 1010 (-6)

Operátorem v jazyku Python je **~.**

Příklady v jazyku Python:

```
>>> ~2
-3
>>> ~0
-1
>>> ~-1
0
```

Bitový posun doleva

Bitový posun doleva znamená, že bity se posouvají doleva a do uvolněných míst zprava se doplňují nuly. Od bitového posunu musíme odlišit **posun aritmetický**, který předpokládá, že v nejvyšším bitu (zcela vlevo) je znaménko, takže tento bit se při posunu nevyužívá.

Bitový posun doleva lze s výhodou užít pro násobení čísla násobky dvojky, podobně jako v desítkové soustavě se výhodně násobí číslo násobky desítky pouhým doplňováním nul zprava.

Příklad v jazyku Python:

```
>>> 2 << 3
16
```

Bitový posun doprava

Bitový posun doleva znamená, že bity se posouvají doprava a do uvolněných míst zleva se doplňují nuly. Bit zcela napravo se pak ztrácejí. Od bitového posunu musíme opět odlišit **posun aritmetický.** Bitový posun doprava lze s výhodou užít pro dělení čísla násobky dvojky.

Příklady v jazyku Python:

```
>>> 2 >>3
0
>>> 74 >> 1
37
```

Operace s bity lze aplikovat pouze na celá čísla a dlouhá celá čísla. Bitové operátory předpokládají, že celá čísla jsou reprezentována v dvojkovém doplňkovém tvaru. Pro dlouhá celá čísla fungují bitové operátory, jako kdyby byl znaménkový bit posunut nekonečně doleva.

Na závěr je nutno podotknout, že bitové operace sice Python podporuje, nicméně nepoužijí se příliš často, protože Python není vhodný jazyk pro přímou práci s hardwarem počítače.

Literatura:

[1] Rubeš, J.: Nebojte se programovat, ComputerMedia, Bedihošť 2001

[2] Lutz, M., Ascher, D.: Naučte se Python, Grada, Praha 2003

[3] Beazley, D. M.: Python, Neocortex, Praha 2002

[4] Python Reference Manual

[5] Švec, J.: Létající circus, Python tutoriál, 2003