



# Jython: Logické výrazy alebo Čo je pravda a čo lož (pokračovanie)

V tejto časti dokončíme prehľad jednoduchých príkazov a následne sa zameriame na štruktúrované príkazy.

## Break

Príkaz `break` ukončí vykonávanie cyklu, v ktorom je použitý, a odovzdá riadenie na riadok nasledujúci po tomto cykle:

```
>>> for i in range(0, 9):
...     if i > 4:
...         break
...     print "i:", i
...
i: 0
i: 1
i: 2
i: 3
i: 4
```

## Continue

Príkaz `continue` ukončí vykonávanie tela cyklu a odovzdá riadenie na začiatok cyklu na ďalšie kolo iterácie:

```
>>> for i in range(0, 5):
...     if i == 2:
...         print 'Dvojku vynechame'
...         continue
...     print "i:", i
...
i: 0
i: 1
Dvojku vynechame
i: 3
i: 4
```

## Del

Príkaz `del` vymaže premennú:

```
>>> a = [1, 2, 3]
>>> print a
[1, 2, 3]
>>>
>>> del a
>>> print a
Traceback (innermost last):
  File "<console>", line 1, in ?
NameError: a
>>>
>>> a = [1,2,3,4]
>>> del a[2]
>>> print a
[1, 2, 4]
```

## Pass

Príkaz `pass` nespôsobuje žiadnu akciu. Používa sa v prípadoch, keď syntax jazyka vyžaduje, aby zdrojový kód obsahoval nejaký príkaz, ale súčasne nie je žiaduce, aby sa čokoľvek vykonalo.

```
>>> #definícia prazdnej funkcie, ktora vobec
... nic nerobi, len je
...
>>> def dummy():
...     pass
...
>>>
>>> #definícia prazdnej triedy
>>> class EmptyClass:
...     pass
...
>>>
```

## Print

Príkaz `print` vyhodnotí výraz, ktorý dostane ako parameter, konvertuje ho na reťazec a zapíše ho na štandardný výstup.

```
print [ vyraz]
```

Výstup možno tiež presmerovať do inštancie dátového typu Jython `file`.

```
print >> instancijaTypuSubor, vyraz
>>> print "Hello world!!!"
Hello world!!!
```

## Return

Príkaz `return` ukončí vykonávanie metódy alebo funkcie a vráti hodnotu vyhodnoteného výrazu, ktorý možno zadať ako parameter. Pokiaľ nijaký výraz v príkaze `return` nie je uvedený, vráti sa `None`. Syntaxe:

```
return [ vyraz]
>>> def mojeFce():
...     return 'vrati tento retazec'
...
>>> mojeFce()
'vrati tento retazec'
```

## Štruktúrované príkazy

Štruktúrované príkazy majú tú vlastnosť, že logický riadok môže zaberáť i viac fyzických riadkov. Väčšinou ide o jazykové konštrukcie, ako sú cykly či vetvenie.

## Blok

Jython má neštandardný prístup k definovaniu začiatku a konca bloku kódu. Neexistuje žiadny explicitne definovaný znak (ako napr. zložený zátvorky v Java), určený na tento účel.

Bloky kódu (funkcie, vetvenie `if`, cykly `for` ...) sú definované ich zarovnaním. To znamená, že všetky medzery (presnejšie povedané, biele znaky, ako medzery, tabulátory atď.) sú pri zápise kódu podstatné a majú svoj význam. Musia sa používať konzistentne.

Jedným z prínosov tohto spôsobu zápisu programového kódu je, že všetky programy v Jythone vyzerajú podobne. Spôsoby zarovnania nie sú už vecou štýlu programátora, ale sú vynútené definíciou jazyka.

Pravidlá pre zarovnávanie programového kódu sú nasledujúce:

- Prvý riadok súboru sa musí začínať v najľavejšom stĺpci.
- Každý príkaz v bloku sa musí začínať v tom istom stĺpci.
- Každé z kľúčových slov (`class`, `def`, `for/else`, `if/elif/else`, `try/except/else`, `try/finally`, `while/else`) označuje začiatok nového bloku.
- Nový blok musí byť zarovnaný doprava aspoň o jednu medzeru oproti koncu predchádzajúceho bloku. Štandardne sa zarovnáva doprava o štyri medzery alebo o jeden tabulátor.

Uvedené pravidlá môžu znieť nezrozumiteľne, nasledujúci príklad tieto pravidlá názorne osvetlí:

```
for x in (1,2,3):
    print x          #zarovnanie označujúce
                    #začiatok nového bloku
                    for y in (10, 20, 30):
                        if y > 10:          #zarovnanie označujúce
                                                #začiatok nového bloku
                                                    print y          #zarovnanie označujúce
                                                        #začiatok nového bloku
                                                            print x*y          #zarovnanie doľava označujúce
                                                                koniec predchádzajúceho bloku
                                                                    #ďalší príkaz by mohol pokračovať
                                                                    #ako súčasť vnútorného cyklu
                                                                    #alebo tu ako súčasť vonkajšieho cyklu,
                                                                    #alebo tu ako samostatný príkaz, nepatriaci
                                                                    #do tela cyklu
```

Po riadku obsahujúcom kľúčové slovo interpret príkazov občas z dôvodu konzistencie vyžaduje nový zarovnaný riadok. V prípade, že nechceme, aby sa čokoľvek na tomto vykonalo, možno použiť príkaz `pass`.

```
def prazdnaFunkcia():
    pass
```

## Podmienený príkaz

Podmienený príkaz má v Jythone syntax:

```
if logicky_vyraz:
    blok
elif logicky_vyraz:
    blok
else:
    blok
```

Klauzula `elif` je nepovinná a môže sa v príkaze vyskytovať hocikolko ráz. Klauzula `else` je takisto nepovinná.

Mechanizmus určujúci, ku ktorému príkazu `if` sa viaže koncové `else`, závisí od zarovnania. V ďalej uvedenom príklade sa `else` viaže k `if` na najvyššej úrovni:

```
>>> x, y, z = 10, 20, 30
>>> if x == 3:
...     if y == 20:
...         if z == 35:
...             pass
...     else:
...         print x
```

## Cykly

Jython obsahuje dve konštrukcie pre cykly. Všeobecný cyklus `while` a cyklus `for`, určený na spracúvanie sekvencií.

## While cyklus

Cyklus `while` má nasledujúcu syntax:

```
while logicky_vyraz:
    blok
else:
    blok
```

Telo cyklu bude vykonávané tak dlho, kým bude mať `logicky_vyraz` hodnotu *pravda*. Vnútorň blok cyklu môže obsahovať príkaz `continue`. Ten odovzdá riadenie opäť na začiatok cyklu. Do vnútorného bloku cyklu možno vložiť i príkaz `break`, ktorý ukončí vykonávanie vnútorného tela cyklu.

Klauzula `else` je nepovinná. Blok nasledujúci za týmto kľúčovým slovom sa vykonáva práve raz, pokiaľ sa cyklus skončí normálne, t. j. pokiaľ bude mať `logicky_vyraz` hodnotu *nepravda*. Ak je cyklus ukončený príkazom `break`, blok za príkazom `else` sa nevykoná.

V Jythone nemožno napísať konštrukciu, akú poznáte hoci z Javy:

```
while((c = char()) == -1) {
    System.out.println(c);
}
```

Je to z toho dôvodu, že priradenie nevracia hodnotu ako v Jave. V Jythone miesto toho použijeme obrat:

```
while 1:
    priradovaci príkaz
    if test:
        break
    zbytok bloku
```



■ ŠTEFAN HAVLÍČEK,  
Sales engineer, InterSystems B. V.