



## Jython : Sekvenčné dátové typy

Doposiaľ sme sa zaoberali iba jednoduchými dátovými typmi, ktoré neumožňovali pracovať s celou skupinou hodnôt. Často však potrebujeme pracovať práve s takými skupinami navzájom súvisiacich údajov. Na to slúžia sekvenčné dátové typy. Sekvenčné dátové typy sú *usporiadané*, *indexované*, poliam podobné *kolekcie*, ktoré sa vyskytujú tak v konštantnej, ako aj premenlivej forme. Medzi nekonštantné sekvenčné dátové typy patria *zoznamy*. Konštantné sekvenčné dátové typy sa nazývajú *tuple* (*n-tice*).

### Zoznamy (Lists)

Zoznam je objekt reprezentujúci nekonštantnú, sekvenčnú, heterogénnu kolekciu objektov. Pozrime sa podrobnejšie na významy jednotlivých slov v definícii:

#### Kolekcia

V Jythone zoznam môže obsahovať odkazy na skupinu objektov.

#### Heterogénne

V Jythone nemusí zoznam obsahovať položky rovnakého dátového typu. V jednom zozname môžeme súčasne zhromažďovať základné dátové typy, objekty tried definovaných používateľom, ďalšie zoznamy, tuple alebo asociatívne polia (hashe).

#### Sekvenčné

Jednotlivé prvky v zozname sú dosiahnuteľné prostredníctvom celočíselných indexov. Prvý prvok zoznamu má index 0, posledný prvok zoznamu má index rovný počtu prvkov - 1.

#### Nekonštantné

Obsah zoznamu nie je konštantný, môže sa počas vykonávania kódu meniť. Prvky do zoznamu môžeme dynamicky pridávať, odoberať, prípadne i jednotlivé prvky meniť. To všetko bez toho, aby sme vytvárali nový objekt reprezentujúci obsah zoznamu. Veľkosť, počet prvkov zoznamu sa môže časom meniť.

Zoznam môžeme vytvoriť tak, že vymenujeme jednotlivé položky zoznamu, navzájom oddelené čiarkou, a celý zoznam uzavrieme do hranatých zátvoriek:

```
[ výraz, výraz, ....]
```

Prázdny zoznam je reprezentovaný prázdny hranatými zátvorkami [] .

```
>>> slovo = 'Popokatepetl'
>>> zoznam = [1, 123, slovo, [], 'ahoj']
>>>
>>> zoznam
[1, 123, 'Popokatepetl', [], 'ahoj']
>>>
>>> inyZoznam = [123, zoznam, 'Jirka']
>>>
>>> inyZoznam
[123, [1, 123, 'Popokatepetl', [], 'ahoj'], 'Jirka']
>>>
>>> zasobnik = [] #prazdny zoznam
>>> zasobnik
[]
```

### Indexovanie a výrezy

Zoznamy možno obdobne ako reťazce indexovať a vytvárať výrezy:

```
>>> zoznam
[1, 123, 'Popokatepetl', [], 'ahoj']
>>>
>>> zoznam[0] #Prvy prvok zoznamu
1
>>> zoznam[-1] #Posledny prvok zoznamu
'ahoj'
>>> zoznam[:2] #Prve dva prvky zoznamu
[1, 123]
>>> zoznam[-2:] #Posledne dva prvky zoznamu
[[], 'ahoj']
>>>
>>> len(zoznam) #Pocet prvkov v zozname
5
```

Za pozornosť stojí nasledujúci výraz `zoznam[:]`, ktorý zo zoznamu vytvorí novú, tzv. *plytkú kópiu* (po anglicky *shallow copy*), t. j. nový objekt majúci rovnaký obsah ako pôvodný zoznam.

```
>>> zoznam = [123, 'Eva', 'Ona jedna']
>>> zoznam
[123, 'Eva', 'Tomas']
>>>
>>> kopia = zoznam[:]
#Vytvorim kopiu zoznamu
>>> kopia
[123, 'Eva', 'Tomas']
```

Skutočne ide o kópiu zoznamu, možno to vidieť z nasledujúceho výpisu:

```
>>> zoznam[0] = 'Monika' #Zmenim hodnotu
1. prvku
>>> zoznam
['Monika', 'Eva', 'Tomas']
>>> kopia
#Tu k zmene nedoslo
[123, 'Eva', 'Tomas']
```

To, že ide o plytkú kópiu, je viditeľné z nasledujúceho:

```
>>> zoznam = [[1], [2], [3]]
>>> kopia = zoznam[:]
>>> kopia
[[1], [2], [3]]
>>>
>>> zoznam[0][0] = 111
>>> zoznam
[[111], [2], [3]]
>>> kopia
[[111], [2], [3]]
```

Prí zmene obsahu objektu vnoreného do zoznamu `zoznam` sa táto zmena prejaví i v kópii zoznamu `kopia`.

Oproti tomu priradený `zoznam2 = zoznam` priradí premenné `zoznam2` iba odkaz na existujúci objekt pôvodného zoznamu:

```
>>>
>>> inyZoznam = zoznam
>>>
>>> Zoznam
[123, 'Eva', 'Tomas']
>>> inyZoznam
[123, 'Eva', 'Tomas']
>>>
>>> zoznam[0] = 'Monika'
>>> zoznam
['Monika', 'Eva', 'Tomas']
>>> inyZoznam
['Monika', 'Eva', 'Tomas']
```

Hodnota zoznamu je skutočne iba *referencia na objekt*. Obdobne ako v Jave možno tieto od-

kazy uchovávať na viacerých miestach, vo viacerých premenných, ale stále pôjde o ten istý objekt, ten istý zoznam. Nasledujúci príklad to pekne demonštruje:

```
>>> klietka = []
>>> kamery = [klietka] * 4
#Jedna klietka, 4 kamery
>>>
>>> kamery
[[[]], [], [], []]
>>> klec.append('macka')
#Vloz macku do klietky
>>>
>>> kamery
[['macka'], ['macka'], ['macka'], ['macka']]
```

Získali sme 4 rôzne pohľady na tú istú macku v jednej klietke.

Prí vnáraní zoznamov platia pre indexáciu pravidlá zrejme z príkladu:

```
>>> riadok1 = [11,12,13]
>>> riadok2 = [21,22]
>>> matica = [riadok1, riadok2]
>>>
>>> matica
[[11, 12, 13], [21, 22]]
>>> matica[0][2] #Prvy riadok, tretii prvok
13
>>> matica[1][1] #Druhy riadok, druhy prvok
22
```

Priradovať hodnoty možno nielen jednotlivým prvkom zoznamu, ale i výrezom:

```
>>> zasobnik = ['a', 'b', 'c', 'd', 'e']
>>> zasobnik[0] = 'ahoj'
>>>
>>> zasobnik
['ahoj', 'b', 'c', 'd', 'e']
>>>
>>> zasobnik[3:4] = ['koniec']
>>>
>>> zasobnik
['ahoj', 'b', 'c', 'koniec', 'e']
>>>
```

ale pozor:

```
>>> zasobnik = [0,1,2,3]
>>> zasobnik[2:3] = 'abcde'
>>>
>>> zasobnik
[0, 1, 'a', 'b', 'c', 'd', 'e', 3]
```

Na zoznamy sa dajú aplikovať aj operátory \* (hviezdička) a + (plus):

```
>>> s1 = ['a', 'b', 'c']
>>> s2 = [1, 2, 3]
>>>
>>> s1 + s2
['a', 'b', 'c', 1, 2, 3]
>>>
>>> s1 * 2
['a', 'b', 'c', 'a', 'b', 'c']
```

Test, či prvok `x` je členom zoznamu `s`, možno uskutočniť pomocou operátora `in`:

```
>>> s = ['jedna', 2, 'tre', 'yottsui']
>>>
>>> x = 'tre'
>>>
>>> x in s
1
>>>
>>> 'styri' in s
0
```



ŠTEFAN HAVLÍČEK,  
Sales Engineer, InterSystems B.V.