Факултет за електротехника и
информарички технологии - Скопје

PYTHON
(2 дел)

http://docs.python.org/tut/tut.html

Изработил:
Билјана Милева - Бошкоска

Скопје
18.10.2006

Наредбата del

Ова е начин да се отстрани даден елемент од листата ако е даден неговиот индекс наместо неговата вредност. Оваа наредба се разликува од наредбата pop() која враќа вредност. Наредбата del може да се искористи и за отсранување и на делови од листата или пак сосема да ја избреше листата . На пример:

```
>>> a = [-1, 1, 66.25, 333, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 333, 1234.5]
>>> del a[2:4]
>>> a
[1, 66.25, 1234.5]
>>> del a[:]
>>> a
[]
```

Наредбата del може да се искористи и за бришење на варијабли:

```
>>> del a
```

Листите и стринговите имаат многу заеднички својства, како на пример индексирањето и операциите на slicing. Тие се два примери на *секвенцијални типови на податоци*. Со оглед на тоа дека Pythone е јазик кој е во развој, може да се додадат и други секвенцијални типови на податоци. Друг вид на стандарден секвенцијален тип на податоци е: *торката*.

Торката се состои од повеќе варијабли одделени со запирка, како на пример:

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```

Како што се гледа од примерот, торките на излез се претставени во загради за да може правилно да се интерпретираат вгнездените торки; при внесување, може но не мора да се користат загради, иако често пати заградите се неопходни (ако торката е дел од поголем исказ).
Торките имаат повеќе намени. На пример: (x, y) координатни парови, податоци за вработен во база на податоци, итн. Торките, како и стринговите, се непроменливи: не е можно да се изврши доделување на определени делови од елементите на торката (иако тој ефект може да се симулира со slicing и конкатенација). Исто така можно е да се креираат торки кои содржат променливи објекти, како што се листите.

Специјален проблем е креирање на торка која содржи 0 или1: синтаксата има некоја додатна смисла доделена на овие вредности. Празните торки се конструираат со празни загради. Торката со еден елемент се креира на тој начин што после вредноста се става запирка (не е доволно да се стави единечна вредност во загради). Изгледа грдо, но е ефективно. На пример:

```
>>> empty = ()
>>> singleton = 'hello',    # <-- note trailing comma
>>> len(empty)
0
>>> len(singleton)
1
>>> singleton
('hello',)
```

Исказот t = 12345, 54321, 'hello!' е пример на *пакување на торки*: вредностите 12345, 54321 и 'hello!' се спакувани во торка. Обратната операција е исто така можна:

```
>>> x, y, z = t
```

Ова се нарекува, одпакување на секвенцата (*sequence unpacking)*.

Python вклучува и типови на податоци наречени *множества.* множество е неподредена колекција на елементи без дупликати. Множествата се поддржани со математички операции како унија, пресек, разлика и симерична разлика.

Еве една кратка демонстрација:

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> fruit = set(basket)               # create a set without duplicates
>>> fruit
set(['orange', 'pear', 'apple', 'banana'])
>>> 'orange' in fruit                 # fast membership testing
True
>>> 'crabgrass' in fruit
False

>>> # Demonstrate set operations on unique letters from two words
...
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                                 # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b                             # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b                             # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b                             # letters in both a and b
set(['a', 'c'])
>>> a ^ b                             # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

Кога се прават јамки во речници, клучниот и кореспондирачки елемент може да се извлече со користење на метдот iteritems.

```
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.iteritems():
...     print k, v
...
gallahad the pure
robin the brave
```

Кога се прават јамки низ низи, индексот на позицијата и соодветната вредност може да се извлечат со користење на функцијата enumerate().

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):
...     print i, v
...
0 tic
1 tac
2 toe
```

Кога се прават јамки низ две или повеќе низи истовремено, влезовите може да се спарат со функцијата zip().

```
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...     print 'What is your %s?  It is %s.' % (q, a)
...
What is your name?  It is lancelot.
What is your quest?  It is the holy grail.
What is your favorite color?  It is blue.
```

Кога се прават јамки низ низа но обратно, почнувајќи од крајот, прво специфицирајте ја низата во насока нанапред а потоа повикајте ја функцијата reversed().

```
>>> for i in reversed(xrange(1,10,2)):
...     print i
...
9
7
5
3
1
```

Кога се прават јамки низ низа на сортиран начлин, се користи функцијата sorted() која враќа нова сортирана листа, а притоа изворната листа ја остава непроменета.

```
>>> basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
>>> for f in sorted(set(basket)):
...     print f
...
apple
banana
orange
pear
```