Krotki

Rozdział 10

Python dla wszystkich
www.py4e.pl

Krotki są jak listy

Krotki to kolejny typ sekwencji, które działają podobnie do list – mają elementy indeksowane od 0

```
>>> x = ('Gienek', 'Staszek', 'Józek')
>>> print(x[2])
Józek
>>> y = ( 1, 9, 2 )
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

```
>>> for iter in y:
...     print(iter)
...
1
9
2
>>>
```

ale... krotki są "niezmienne"

Inaczej niż listy, raz stworzona krotka nie może zmienić swojej zawartości – podobnie jak ciąg znaków

```
>>> x = [9, 8, 7]
>>> x[2] = 6
>>> print(x)
>>>[9, 8, 6]
>>>
```

```
>>> y = 'ABC'
>>> y[2] = 'D'
Traceback:'str'
object does
not support item
Assignment
>>>
```

```
>>> z = (5, 4, 3)
>>> z[2] = 0
Traceback:'tuple'
object does
not support item
Assignment
>>>
```

Czego nie robić z krotkami

```
>>> x = (3, 2, 1)
>>> x.sort()
Traceback:
AttributeError: 'tuple' object has no attribute 'sort'
>>> x.append(5)
Traceback:
AttributeError: 'tuple' object has no attribute 'append'
>>> x.reverse()
Traceback:
AttributeError: 'tuple' object has no attribute 'reverse'
>>>
```

O dwóch takich... sekwencjach

```
>>> l = list()
>>> dir(l)
[(...), 'append', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']

>>> t = tuple()
>>> dir(t)
[(...), 'count', 'index']
```

Krotki są bardziej wydajne

- Ponieważ Python nie musi tworzyć struktury krotek tak, aby były zmienne, są one prostsze i działają oraz wykorzystują pamięć wydajniej niż listy.
- Dlatego w programach, w których tworzymy "tymczasowe zmienne", od list wolimy krotki.

Krotki i operacja przypisania

- Możemy także umieścić krotkę po lewej stronie operacji przypisania
- Możemy nawet opuścić nawiasy

```
>>> (x, y) = (4, 'fred')
>>> print(y)
fred
>>> (a, b) = (99, 98)
>>> print(a)
99
```

Krotki i słowniki

Metoda items() w przypadku słownika zwraca listę par (klucz, wartość), czyli krotek

```
>>> d = dict()
>>> d['csev'] = 2
>>> d['cwen'] = 4
>>> for (k,v) in d.items():
...     print(k, v)
...
csev 2
cwen 4
>>> tups = d.items()
>>> print(tups)
dict_items([('csev', 2), ('cwen', 4)])
```

Krotki są porównywalne

Operatory porównania działają z krotkami i innymi sekwencjami. Jeśli pierwsze elementy są równe, Python przechodzi dalej, aż znajdzie takie elementy, które się różnią.

```
>>> (0, 1, 2) < (5, 1, 2)
True
>>> (0, 1, 2000000) < (0, 3, 4)
True
>>> ( 'Jones', 'Sally' ) < ('Jones', 'Sam')
True
>>> ( 'Jones', 'Sally') > ('Adams', 'Sam')
True
```

Sortowanie list krotek

- Możemy wykorzystać możliwość sortowania listy krotek, żeby uzyskać posortowaną wersję słownika
- Najpierw sortujemy słownik według klucza, używając metody items() i funkcji sorted()

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> d.items()
dict_items([('a', 10), ('c', 22), ('b', 1)])
>>> sorted(d.items())
[('a', 10), ('b', 1), ('c', 22)]
```

Używanie sorted()

Możemy zrobić to jeszcze bardziej bezpośrednio, korzystając z wbudowanej funkcji sorted(), która przyjmuje sekwencję jako parametr i zwraca posortowaną sekwencję

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> t = sorted(d.items())
>>> t
[('a', 10), ('b', 1), ('c', 22)]
>>> for k, v in sorted(d.items()):
...     print(k, v)
...
a 10
b 1
c 22
```

Sortowanie wg. wartości zamiast kluczy

- Gdybyśmy mogli stworzyć listę krotek w formie (wartość, klucz), moglibyśmy sortować według wartości
- Osiągniemy to pętlą for, która tworzy listę krotek

```
>>> c = {'a':10, 'b':1, 'c':22}
>>> tmp = list()
>>> for k, v in c.items() :
...     tmp.append( (v, k) )
...
>>> print(tmp)
[(10, 'a'), (22, 'c'), (1, 'b')]
>>> tmp = sorted(tmp, reverse=True)
>>> print(tmp)
[(22, 'c'), (10, 'a'), (1, 'b')]
```

10 najczęstszych słów

```
fhand = open('romeo.txt')
counts = {}
for line in fhand:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0 ) + 1

lst = []
for key, val in counts.items():
    newtup = (val, key)
    lst.append(newtup)

lst = sorted(lst, reverse=True)

for val, key in lst[:10] :
    print(key, val)
```

Jeszcze krótsza wersja

```
>>> c = {'a':10, 'b':1, 'c':22}

>>> print( sorted( [ (v,k) for k,v in c.items() ] ) )

[(1, 'b'), (10, 'a'), (22, 'c')]
```

Lista składana jest tworzona dynamicznie. W tym przypadku tworzymy listę odwróconych krotek i sortujemy ją.

http://wiki.python.org/moin/HowTo/Sorting

Podsumowanie

- Składnia krotek
- Niezmienność
- Porównywalność
- Sortowanie
- Krotki w instrukcjach przypisania
- Sortowanie słowników po kluczach lub wartościach

UMSI

Podziękowania dla współpracowników