Tuples

Chapitre 10

Python for Informatics: Exploring Information
www.pythonlearn.com

Les tuples sont comme les listes

- Les Tuples sont un autre type de séquence qui fonctionnent plus ou moins comme une liste - ils ont des éléments dont l'index commence à 0

```
>>> x = ('Glenn', 'Sally', 'Joseph')
>>> print x[2]Joseph
>>> y = ( 1, 9, 2 )
>>> print y
(1, 9, 2)
>>> print max(y)
9
```

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

mais... les tuples sont “immutables”

- Contrairement aux listes, une fois qu’un tuple est créé, son contenu ne peut être modifié - comme une chaîne

```
>>> 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
>>>
```

Les choses à ne pas faire avec les tuples

```
>>> 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'
>>>
```

Une histoire de deux séquences

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

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

Les tuples sont plus efficaces

- Python n'ayant pas à construire des structures de tuple pour être modifiable, ils sont plus simples et plus efficaces en termes d'utilisation de la mémoire et de performances que les listes
- Donc dans notre programme lorsque nous utilisons des “variables temporaires”, nous préférons les tuples aux listes

Tuples et affectation

- Nous pouvons également mettre un tuple sur le côté gauche d'une instruction d'assignation
- Nous pouvons même omettre les parenthèses

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

Tuples et dictionnaires

- La méthode items() dans les dictionnaires retourne une liste de (clé, valeur) n-uplets

```
>>> 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
[('csev', 2), ('cwen', 4)]
```

Les tuples sont comparables

- Les opérateurs de comparaison fonctionnent avec les tuples et les autres séquences. Si le premier élément est égal, Python continue à l'élément suivant et ainsi de suite, jusqu'à ce qu'il trouve les éléments qui diffèrent.

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

Trier des listes de tuples

- Nous pouvons profiter de la possibilité de trier une liste de tuples pour obtenir une version triée d'un dictionnaire
- Tout d'abord, nous trions le dictionnaire par la clé à l'aide de la méthode items()

```
>>> d = {'a':10, 'b':1, 'c':22}
>>> t = d.items()
>>> t
[('a', 10), ('c', 22), ('b', 1)]
>>> t.sort()
>>> t
[('a', 10), ('b', 1), ('c', 22)]
```

Utilisation de sorted()

Nous pouvons le faire plus directement à l'aide de la fonction intégrée de tri qui prend une séquence en tant que paramètre et retourne une séquence triée

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

Trier par valeurs au lieu de la clé

- Si nous pouvons construire une liste de tuples de la forme (valeur, clé) nous pouvons trier par valeur
- Nous le faisons avec une boucle for qui crée une liste de tuples

```
>>> 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.sort(reverse=True)
>>> print tmp
[(22, 'c'), (10, 'a'), (1, 'b')]
```

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

lst = list()
for key, val in counts.items():
    lst.append( (val, key) )
lst.sort(reverse=True)
for val, key in lst[:10] :
    print key, val
```

Les 10 mots les plus courants

Version encore plus courte

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

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

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

La technique de ”List comprehension” crée une liste dynamique. Dans ce cas, nous faisons une liste de tuples inversées, puis nous la trions.

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

Résumé

- Syntaxe de Tuple
- Immutabilité
- Comparabilité
- Tri

- Tuples dans les instructions d'assignation
- Tri des dictionnaires par clé ou valeur

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Translation: Frederic Foiry

...