Dictionnaires

S. Vialette

IGM-LabInfo, Université Paris-Est

21 janvier 2008

Dictionnaires

Un des types de données fondamentaux de Python est le dictionnaire, qui défini une relation 1 à 1 entre des clés et des valeurs.

- Un dictionnaire Python est similaire à une instance de la classe `Hashtable` en Java.
- En Python, un dictionnaire est comme une table de hachage en Perl.

Dictionnaire

```
>>> d = {"server" : "monge.univ-mlv.fr", "database" : "sql-base"}
>>> d
{'database': 'sql-base', 'server': 'monge.univ-mlv.fr'}
>>> type(d)
<type 'dict'>
>>> d["server"]
'monge.univ-mlv.fr'
>>> d["database"]
'sql-base'
>>> d["owner"]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'owner'
>>>
```

Dictionnaire

```
>>> # dictionnaire vide
>>> d1 = {}
>>> d1
{}
>>> type(d1)
<type 'dict'>
>>> # dictionnaire vide
>>> d2 = dict()
>>> d2
{}
>>> type(d2)
<type 'dict'>
>>>
>>> d1 == d2
True
>>> len(d1), len(d2)
(0, 0)
>>>
```

Dictionnaire

À partir de Python 2.3, il est possible de créer facilement des dictionnaire dont tous les éléments ont la même valeur.

```
>>> d = dict().fromkeys(("x", "y"), 5)
>>> # ou encore {}.fromkeys(("x", "y"), 5)
>>> d
{'y': 5, 'x': 5}
>>>
>>> # Mais attention ...
>>> d = {}
>>> d.fromkeys(("x", "y"), 5)
{'y': 5, 'x': 5}
>>> d
{}
>>>
>>> # La valeur par defaut est None
>>> d = {}.fromkeys(("x", "y"))
>>> d
{'y': None, 'x': None}
```

Dictionnaire

```
>>> d = {"x" : [1, 2, 3], "y" : [4, 5, 6]}
>>> d
{'y': [4, 5, 6], 'x': [1, 2, 3]}
>>> d["x"]
[1, 2, 3]
>>> type(d["x"])
<type 'list'>
>>>
>>> for i in d["x"]:
...     print i, "␣",
...
1 ␣ 2 ␣ 3 ␣
>>>
>>> print ",".join([str(i) for i in d["x"]])
1,2,3
>>>
```

Dictionnaire

Il n'est pas nécessaire que toutes les valeurs soient d'un même type.

```
>>> d = {"x" : [1, 2], "y" : (3, 4), "z" : 5}
>>> d
{'y': (3, 4), 'x': [1, 2], 'z': 5}
>>>
>>> d["x"], type(d["x"])
([1, 2], <type 'list'>)
>>>
>>> d["y"], type(d["y"])
((3, 4), <type 'tuple'>)
>>>
>>> d["z"], type(d["z"])
(5, <type 'int'>)
>>>
```

Dictionnaire

La méthode `keys` retourne la liste de toutes les clés.

```
>>> d = {"server" : "monge.univ-mlv.fr", "database" : "sql-base"}
>>> d
{'database': 'sql-base', 'server': 'monge.univ-mlv.fr'}
>>> d.keys()
['database', 'server']
>>> type(d.keys())
<type 'list'>
>>>
>>> for key in d.keys():
...     print "key=%s value=%s" % (key, d[key])
...
key=database value=sql-base
key=server value=monge.univ-mlv.fr
>>> # remarquer ici que l'ordre des cles different de celui
>>> # donne lors de la definition de d
```

Dictionnaire

À partir de Python 2.2, il n'est même plus nécessaire d'utiliser la méthode `keys` dans les boucles.

```
>>> d = {"server" : "monge.univ-mlv.fr", "database" : "sql-base"}
>>> d
{'database': 'sql-base', 'server': 'monge.univ-mlv.fr'}
>>>
>>> for key in d: # equivalent a "for key in d.keys():"
...     print "key=%s value=%s" % (key, d[key])
...
key=database value=sql-base
key=server value=monge.univ-mlv.fr
>>>
```

Modifier un dictionnaire

```
>>> d = {"server" : "monge", "database" : "sql-base"}
>>> d
{'database': 'sql-base', 'server': 'monge'}
>>>
>>> d["port"] = 1234              # Ajout d'un element
>>> d
{'database': 'sql-base', 'port': 1234, 'server': 'monge'}
>>> d["server"] = "localhost" # Modification d'un element
>>> d
{'database': 'sql-base', 'port': 1234, 'server': 'localhost'}
>>>
```

Supprimer des entrées dans un dictionnaire

```
>>> d = {"server" : "monge", "database" : "sql-base", "port" : 1234}
>>> d
{'database': 'sql-base', 'port': 1234, 'server': 'monge'}
>>>
>>> del d["port"] # Suppression d'une entree
>>> d
{'database': 'sql-base', 'server': 'monge'}
>>>
>>> d.clear()      # Suppression de toutes les entrees
>>> d              # Le dictionnaire existe toujours
{}
>>> del d          # Supprimer le dictionnaire lui meme
>>> d
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'd' is not defined
>>>
```

Les méthodes values et item

```
>>> d = {"server" : "monge", "database" : "sql-base", "port" : 1234}
>>>
>>> d.values()  # Toutes les valeurs
['sql-base', 1234, 'monge']
>>> d.items()   # Toutes les paires (cle, valeur)
[('database', 'sql-base'), ('port', 1234), ('server', 'monge')]
>>>
>>> ["%s=%s" % (k, v) for k, v in d.items()]
['database=sql-base', 'port=1234', 'server=monge']
>>>
```

Résumons :

- `keys` retourne la liste des clés.
- `values` retourne la liste des valeurs.
- `items` retourne la liste des paires (clé, valeur).

Appartenance (membership)

À l'instar des listes, il est possible de tester l'appartenance.

```
>>> d = {"server" : "monge", "database" : "sql-base", "port" : 1234}
>>> d.has_key("server")
True
>>> d.has_key("DNS")
False
>>> # Plus simplement
>>> "server" in d
True
>>> "DNS" in d
False
>>> # Ou avec negation
>>> "server" not in d
False
>>> "DNS" not in d
True
>>>
```

Un peu plus sur la création de dictionnaires

```
>>> dict(server = "monge", database = "sql-base")
{'database': 'sql-base', 'server': 'monge'}
>>> # Attention
>>> dict("server" = "monge", "database" = "sql-base")
  File "<stdin>", line 1
SyntaxError: keyword can't be an expression
>>>
>>> d1 = dict(server = "monge", database = "sql-base")
>>> d2 = dict(**d1)
>>> d2
{'server': 'monge', 'database': 'sql-base'}
>>>
```

Un peu plus sur la création de dictionnaires

```
>>> my_keys = ["server", "database", "port"]
>>> my_values = ["monge", "sql-base", 1234]
>>>
>>> # utilisons la fonction 'zip'
>>> zip(my_keys, my_values)
[('server', 'monge'), ('database', 'sql-base'), ('port', 1234)]
>>> # et donc ...
>>> dict(zip(my_keys, my_values))
{'database': 'sql-base', 'port': 1234, 'server': 'monge'}
>>>
>>> # Nous pouvons varier (presque) a l'infini ...
>>> dict([("abcd"[i], i) for i in range(len("abcd"))])
{'a': 0, 'c': 2, 'b': 1, 'd': 3}
>>>
```

Un peu plus sur l'accès aux éléments

```
>>> d =  dict(server = "monge", database = "sql-base")
>>> val = d.get('server')
>>> print val
monge
>>> val = d.get('port')
>>> print val
None
>>> val = d.get('port', 80)
>>> print val
80
>>> d
{'database': 'sql-base', 'server': 'monge'}
>>> d.setdefault('database', 'another-sql-base')
'sql-base'
>>> d
{'database': 'sql-base', 'server': 'monge'}
>>> d.setdefault('port', 1234)
1234
>>> d
{'database': 'sql-base', 'port': 1234, 'server': 'monge'}
```

Pour en savoir plus ...

```
>>> dir({})
['__class__', '__cmp__', '__contains__', '__delattr__',
 '__delitem__', '__doc__', '__eq__', '__ge__', '__getattribute__',
 '__getitem__', '__gt__', '__hash__', '__init__', '__iter__',
 '__le__', '__len__', '__lt__', '__ne__', '__new__', '__reduce__',
 '__reduce_ex__', '__repr__', '__setattr__', '__setitem__',
 '__str__', 'clear', 'copy', 'fromkeys', 'get', 'has_key',
 'items', 'iteritems', 'iterkeys', 'itervalues', 'keys', 'pop',
 'popitem', 'setdefault', 'update', 'values']
>>>
```