Dictionnaire - Résumé

2025-06-24

1. Définition

- Un **dictionnaire**, parfois appelé *table d'association* est une structure de donnée gardant en mémoire des informations de la forme `(clé, valeur)`.
- Un dictionnaire permet d'accéder rapidement à une `valeur` à partir de sa `clé`.

2. Python

En Python les dictionnaires sont implantés par le type `dict`.

Rappels

```
>>> contact = {"nom": "Duchmol", "prenom": "Robert",
...            "age": 122, "taille": 122, "masse": 122} # accolades !
>>> contact["nom"] # accéder à une valeur par sa clé.
'Duchmol'
>>> contact["prenom"] = "Roberto" # les dict sont mutables
>>> contact["tel"] = "0011223344" # insérer une nouvelle paire
>>> contact
{'nom': 'Duchmol', 'prenom': 'Roberto', 'age': 122,
 'taille': 122, 'masse': 122, "tel": "0011223344"}
>>> "adresse" in contact # La clé est-elle dans le dict ?
False
>>> "nom" in contact
True
>>> contact[0] # pas d'indice !!!
Traceback (most recent call last):
  File "<python-input-6>", line 1, in <module>
    contact[0]
    ~~~~~~~^^^
KeyError: 0
>>> len(contact) # mais une longueur
6
```

Trois manières d'itérer sur les dictionnaires

```
>>> for cle in contact: # sur les clés
...     cle
...
'nom'
'prenom'
'age'
'taille'
'masse'
'tel'
>>> for cle in contact.keys(): # identique au précédent...
...     cle, contact[cle]
...
('nom', 'Duchmol')
('prenom', 'Roberto')
('age', 122)
('taille', 122)
('masse', 122)
('tel', "0011223344")
>>> for valeur in contact.values(): # sur les valeurs
...     valeur
...
'Duchmol'
'Roberto'
122
122
122
"0011223344"
>>> for cle, valeur in contact.items(): # sur les paires (cle, valeur)
...     f"{cle} -> {valeur}"
...
'nom -> Duchmol'
'prenom -> Roberto'
'age -> 122'
'taille -> 122'
'masse -> 122'
'tel -> 0011223344'
>>>
```

3. Interface

Opinion personnelle, ce n'est pas toujours clair dans la littérature

1. Créer un dictionnaire vide.
2. Mesurer un dictionnaire : combien d'entrées comporte-t-il ?
3. Ajouter un couple `(clé, valeur)`.
4. Accéder à une `valeur` depuis sa `clé`.
5. Retirer un couple `(clé, valeur)`.
6. Répondre à la question : ce dictionnaire comporte-t-il une entrée avec cette `clé` ?

On peut ajouter d'autres manipulations, en particulier :

1. Itérer sur les `clés`, sur les `valeurs`, sur les couples `(clés, valeurs)` ;
2. Vider un dictionnaire ;
3. Comparer deux dictionnaires (`d1 == d2`) ;
4. Créer un dictionnaire avec des valeurs existantes `d = {'nom': 'Minvussa', 'prenom': 'Gerard'}` ;
5. Faire une copie du dictionnaire etc.

4. Fonction de hachage

On nomme **fonction de hachage**, de l'anglais *hash function* (*hash* : pagaille, désordre, recouper et mélanger) une fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une empreinte numérique servant à identifier rapidement la donnée initiale, au même titre qu'une signature pour identifier une personne. Les fonctions de hachage sont utilisées en informatique et en cryptographie notamment pour reconnaître rapidement des fichiers ou des mots de passe.

5. Implantation des dictionnaires avec une table de hachage

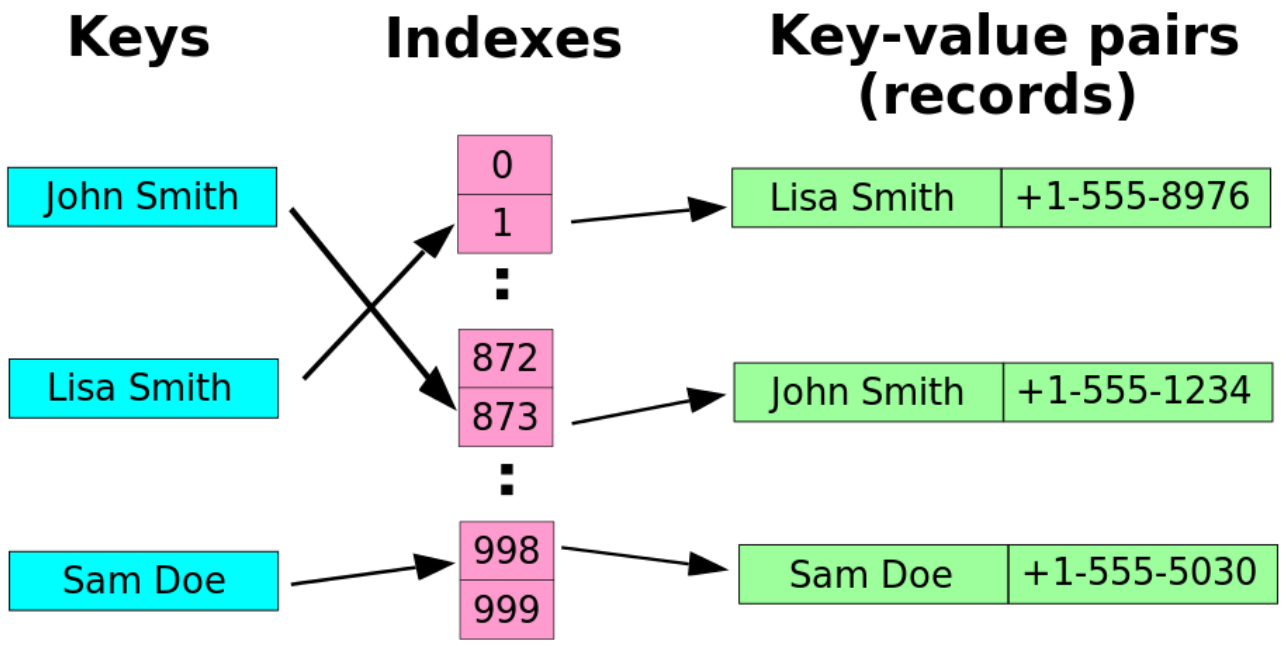

Figure 1: hachage

Le principe est d'utiliser une fonction de hachage (voir plus haut) afin de localiser rapidement les paires `(clé, valeur)`

1. On crée un dictionnaire :

   On initialise un tableau avec un grand nombre d'éléments tous identiques `None` en Python, par exemple.

2. On ajoute le couple `(clé, valeur)` :

   On calcule le hash de la `clé` (en choisissant une fonction de hachage adaptée). Ce nombre devient l'`indice` où sera enregistré la paire.

Exemple

```
>>> contenu = [None] * 1024
>>> hachage('clé')
5
>>> contenu[5] = ('clé', 'valeur')
>>> contenu
[None, None, None, None, None, (clé, valeur), None, None, ...]
>>> contenu[ hachage(('clé')) ][ 1 ]
'valeur'
```

6. Exemple d'implantation

- Une implémentation simpla mais bof bof
- Une implémentation relativement complète et détaillée