Chap. 2 Types structurés Thème 2

Dictionnaires

Sommaire du chapitre 2

Un **dictionnaire** est une table associative qui fait correspondre des **clés** à des **valeurs** de type quelconque. En d'autres termes, il s'agit d'une collection non ordonnée d'éléments.

Remarques

Les clés doivent être d'un type immuable, donc soit des nombres, soit des chaînes de caractères, soit des tuples à condition que ceux-ci ne contiennent que des éléments eux-mêmes immuables. Dans la majorité des cas, les clés sont des chaînes de caractères, ce qui explique le nom de *dictionnaire*, qui associe à chaque mot sa définition.

I Création d'un dictionnaire

En langage Python, les dictionnaires sont notés entre **accolades**. Chaque élément est constitué d'une clé associée à une valeur selon la syntaxe `cle : valeur`.

Exemples

L'instruction `annuaire = {10: 'Etienne', 20: 'Tom', 30: 'Pauline'}` crée le dictionnaire `annuaire`.

Remarques

Un dictionnaire vide est noté `{ }`.

II Accès aux éléments du dictionnaire

Les éléments du dictionnaire ne sont pas indexés. Il est donc impossible d'afficher son n-ième élément. On accède à une valeur grâce à la **clé** qui lui est associée.

C'est la raison pour laquelle les clés d'un dictionnaire doivent être toutes différentes.

Exemples

`annuaire[0]` renvoie ainsi une erreur de clé « `KeyError` » tandis que `annuaire[10]` renvoie `'Etienne'` car le nombre `10` est la clé du dictionnaire à laquelle est associée la valeur `'Etienne'`.

Remarques

— Si l'on tente d'accéder à une valeur avec une clé erronée ou inexistante, on obtient une erreur :

```
>>> annuaire[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 0
```

— On peut tester la présence ou l'absence d'une clé avec l'opérateur `in` :

```
1 if 10 in annuaire : print(annuaire[10])
2 if 20 not in annuaire : print("Il manque l'entrée 20 !")
```

III Énumération des clés d'un dictionnaire

On peut **énumérer** les clés d'un dictionnaire de la même façon qu'un tableau :

```
1 for k in annuaire :      # k vaut successivement les clés du
      dictionnaire
2     print(k + ": " + annuaire[k])
```

IV Modification du dictionnaire

Les dictionnaires sont des types **muables**. Il est possible de modifier les valeurs du dictionnaire, de créer ou de supprimer une entrée du dictionnaire.

Exemples

L'instruction `del annuaire[10]` supprimera l'entrée `10 : 'Etienne'` du dictionnaire `annuaire`.

L'instruction `annuaire[30] = 'Hamza'` remplace la valeur `'Pauline'` par la valeur `'Hamza'` ou crée une nouvelle entrée dans le dictionnaire si la clé `30` n'existait pas.

V Méthodes spécifiques du dictionnaire

En Python, on peut récupérer le contenu d'un dictionnaire avec les méthodes `keys` (clés), `values` (valeurs) et `items` (paires clés/valeurs) :

Exemples

`annuaire.keys()` renvoie la collection itérable de toutes les clés du dictionnaire `annuaire`, c'est-à-dire retourne la liste `[10, 20, 30]`.

`annuaire.values()` renvoie la collection itérable de toutes les valeurs du dictionnaire `annuaire`, c'est-à-dire retourne la liste `['Etienne, 'Tom', 'Hamza']`.

`annuaire.items()` renvoie la collection de tous les objets du dictionnaire `annuaire`, c'est-à-dire retourne la liste de tuples `[(10, 'Etienne), (20, 'Tom'), (30,'Hamza')]`.

Les listes obtenues par les méthodes `keys()`, `values()` et `items()` peuvent être énumérées avec une boucle `for ... in` :

```
1 for cle, valeur in annuaire.items() :
2     print(cle + ": " + valeur)
```