Les dictionnaires

Dernière mise à jour le : 07/12/2023

Introduction

Prenons l'exemple d'un répertoire téléphonique. Nous pouvons la modéliser simplement comme un tableau (ou liste) contenant des tableaux de la forme `[nom, numéro]` :

```
>>> liste_tel = [["Paul", 5234],
                 ["Victor", 5186],
                 ["Rose", 5678],
                 ["Hélène", 5432]]
```

Si nous voulons appeler Rose, nous avons deux possibilités avec un tel tableau :

- soit il faut savoir que les informations la concernant sont dans le quatrième élément de la liste (ce qui ne semble pas très pratique et réaliste)

```
>>> liste_tel[2][1]  # il faut savoir que l'index de Rose est 2
5678
```

- soit nous cherchons dans le tableau en partant du premier élément de la liste jusqu'à ce que nous trouvions *Rose* (ce qui revient à feuilleter son répertoire) : cela nécessite d'utiliser une boucle pour parcourir le tableau.

```
>>> for element in liste_tel:
        if element[0] == 'Rose':
            print(element[1])

5678
```

Vous conviendrez que ce n'est pas pratique pour accéder à son numéro de téléphone. De même, la modification ou l'ajout d'une information nécessiterait de devoir feuilleter tout le répertoire. Il semblerait plus pratique d'associer un numéro à un nom, c'est ce que les dictionnaires vont permettre !

Les dictionnaires en Python

Un **dictionnaire** est un ensemble **non ordonné** de paires (clé, valeur). On peut accéder à une valeur à partir de sa clé, et cet accès est très rapide.

Création d'un dictionnaire

Le répertoire téléphonique précédent peut se mémoriser dans le dictionnaire `repertoire` suivant :

```
repertoire = {'Paul': 5234, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}
```

Analyse :

- Vous aurez noté que les dictionnaires Python se représentent entre accolades `{}`.
- Les différentes paires sont séparées par des virgules et sont de la forme `clé: valeur`. Dans cet exemple :
  - la valeur `5234` est associée à la clé `'Paul'`
  - la valeur `5186` est associée à la clé `'Victor'`
  - etc.

Dans ce cas, on dit qu'on a créé le dictionnaire *par extension*.

On dit qu'un dictionnaire n'est pas ordonné car l'ordre des paires (clé, valeur) n'a aucune importance. Ainsi, les deux dictionnaires `d1` et `d2` suivants sont considérés comme égaux.

```
>>> d1 = {'a': 2, 'b': 5}
>>> d2 = {'b': 5, 'a': 2}
>>> d1 == d2
True
```

En Python, un dictionnaire est un objet de type `dict` :

```
>>> type(repertoire)
dict
```

Exercice 1

Écrivez les instructions permettant de répondre aux questions suivantes.

1. Créer un dictionnaire appelé `notes` qui contient les paires (matières, moyenne) de vos trois spécialités.
2. Afficher ensuite ce dictionnaire.

Remarques importantes :

- Une **clé** peut être de type alphabétique, numérique, ou même de type construit sous certaines conditions.
- Les **valeurs** pourront être de tout type sans exclusion.

Par exemple, le dictionnaire suivant est tout à fait correct (essayez de repérer les clés et les valeurs associées, ainsi que leurs types respectifs) :

```
{1: 'deux', 'trois': [4, 5, 6], 'sept': {8: 9, 10: 'onze'}, 12: 13}
```

Accès, modification, ajout, suppression

En Python, le dictionnaire est un objet **mutable**, autrement dit, on peut le modifier (comme nous allons le voir dans le paragraphe suivant)

L'**accès** à une valeur d'un dictionnaire se fait par sa clé :

```
>>> repertoire = {'Paul': 5234, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}
>>> repertoire['Rose']
5678
```

```
>>> fruits = {'poires': 5, 'bananes': 7, 'abricots' : 12}
>>> fruits['abricots']
12
```

Analyse : Pour accéder à une valeur d'un dictionnaire, on utilise les crochets et on y indique à l'intérieur la clé correspondante.

> Vous noterez que c'est beaucoup plus pratique que ce qu'on a vu en introduction.

Le dictionnaire étant un objet *mutable* on peut **modifier** la valeur associée à une clé ou **ajouter** une nouvelle association et afficher le dictionnaire modifié.

```
>>> repertoire = {'Paul': 5234, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}
>>> repertoire['Rose'] = 4921  # clé existante donc modification de la valeur
>>> repertoire
{'Paul': 5234, 'Victor': 5186, 'Rose': 4921, 'Hélène': 5432}
```

```
>>> repertoire['Louane'] = 4118  # nouvelle clé donc ajout d'une nouvelle association
>>> repertoire
{'Paul': 5234, 'Victor': 5186, 'Rose': 4921, 'Hélène': 5432, 'Louane': 4118}
```

Si on essaie d'accéder à une clé qui n'existe pas, une erreur de type `KeyError` (*erreur de clé*) est levée :

```
>>> repertoire['Kylian']
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
KeyError: 'Kylian'
```

Pour **supprimer** une association d'un dictionnaire on peut utilise le mot clé `del`.

```
>>> repertoire
{'Paul': 5234, 'Victor': 5186, 'Rose': 4921, 'Hélène': 5432, 'Louane': 4118}
>>> del repertoire['Paul']
>>> repertoire
{'Victor': 5186, 'Rose': 4921, 'Hélène': 5432, 'Louane': 4118}
```

Taille d'un dictionnaire

La fonction `len` renvoie la taille d'un dictionnaire.

```
>>> repertoire = {'Paul': 5234, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}
>>> len(repertoire)
4
```

```
>>> fruits = {'poires': 5, 'bananes': 7, 'abricots' : 12}
>>> len(fruits)
3
```

Exercice 2

On reprend le dictionnaire `notes` de l'exercice 1. Écrivez les instructions permettant de répondre aux questions suivantes.

1. Afficher la moyenne de NSI.
2. Modifier la moyenne de NSI, qui a gagné 2 points. Afficher le dictionnaire.
3. Ajouter la matière Anglais avec sa moyenne. Afficher le dictionnaire.
4. Afficher la taille du dictionnaire.
5. Supprimer une des trois spécialités et afficher le dictionnaire.

Autres manières de créer un dictionnaire

Pour créer un dictionnaire vide on peut procéder ainsi :

```
>>> d1 = {}  # création d'un dictionnaire vide
>>> d1
{}
```

```
>>> d2 = dict()  # création d'un dictionnaire vide (autre méthode)
>>> d2
{}
```

On peut aussi créer un dictionnaire par compréhension :

```
>>> d3 = {k: k**2 for k in range(1, 10)}  # création d'un dictionnaire par compréhension
>>> d3
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

Analyse : on a ici créé un dictionnaire dont les clés sont égales à `k` et leurs valeurs associées égales à `k**2`, pour `k` allant de 1 à 9.

Il est même possible de créer un dictionnaire à partir d'une liste de couples en le convertissant avec la fonction `dict()`.

```
>>> liste = [('cle1', 'valeur1'),('cle2', 'valeur2')]
>>> d4 = dict(liste)
>>> d4
{'cle1': 'valeur1', 'cle2': 'valeur2'}
```

```
>>> liste_tel = [["Paul", 5234], ["Victor", 5186], ["Rose", 5678], ["Hélène", 5432]]
>>> d5 = dict(liste_tel)
>>> d5
{'Paul': 5234, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}
```

Les itérateurs pour les dictionnaires

Il est possible de parcourir un dictionnaire de trois manières :

- parcourir l'ensemble des **clés** avec la méthode `keys()` ;

- parcourir l'ensemble des **valeurs** avec la méthode `values()` ;
- parcourir l'ensemble des **paires clés-valeurs** avec la méthode `items()`.

```
>>> repertoire = {'Paul': 5234, 'Victor': 5186, 'Rose': 5678, 'Hélène': 5432}

>>> for prenom in repertoire.keys():
        print(prenom)

Paul
Victor
Rose
Hélène
```

```
>>> for num in repertoire.values():
        print(num)

5234
5186
5678
5432
```

```
>>> for prenom, num in repertoire.items():
        print(prenom, '->', num)

Paul -> 5234
Victor -> 5186
Rose -> 5678
Hélène -> 5432
```

On peut aussi interroger l'appartenance d'une valeur ou d'une clé grâce au mot clé `in`.

```
>>> 'John' in repertoire.keys()
False
>>> 'Rose' in repertoire.keys()
True
>>> 'Victor' not in repertoire.keys()
False
>>> 5186 in repertoire.values()
True
```

Exercice 3

On considère le dictionnaire `fruits` suivant. Écrivez les instructions permettant de répondre aux questions suivantes.

```
fruits = {'poires': 5, 'pommes': 11, 'bananes': 7, 'abricots' : 12}
```

1. Afficher tous les fruits du dictionnaire.
2. Afficher toutes les quantités du dictionnaire.
3. Écriver un programme permettant d'obtenir l'affichage suivant.

```
Il reste 5 poires
Il reste 11 pommes
Il reste 7 bananes
Il reste 12 abricots
```

Références :

- Ressources pédagogiques du DIU EIL, Université de Nantes, C. DECLERQ.

Germain BECKER & Sébastien POINT, Lycée Mounier, ANGERS