CC BY NC SA

CHAPITRE 19

Types contruits - les dictionnaires

Il n'y a que dans le dictionnaire que réussite vient avant travail

Pierre Fornerod

Thème : types construits

Contenus	Capacités attendues	Commentaires
Dictionnaires par clés et valeurs	Construire une entrée de dictionnaire. Itérer sur les éléments d'un dictionnaire.	Il est possible de présenter les données EXIF d'une image sous la forme d'un enregistrement. En Python, les p-uplets nommés sont implémentés par des dictionnaires. Utiliser les méthodes keys(), values () et items ().

I. Structure d'un dictionnaire

Déf. 1 Un dictionnaire est un ensemble non-ordonné d'associations clé/valeur. À la différence des tableaux et des tuples, on n'accède pas à un élément par son indice (d'où l'adjectif de « non-ordonné »).

Propriété 1

- Un dictionnaire est un type non ordonné.
- Un dictionnaire est un type mutable.
- Chaque élément d'un dictionnaire est composé d'une clé et d'une valeur. Chaque élément est séparé par une virgule.
- La structure est alors : `dico = {clé1:valeur1, clé2:valeur2}`
- Le type d'un dictionnaire est `dict`

Exemple 1 (Deux exemples de dictionnaires)

```
>>> animaux = {"nom" : "singe", "poids" : 70, "taille":1.75}
>>> placard = {"chemise" : 3, "pantalon" : 6, "tee-shirt" : 7}
>>> type(placard)
dict
```

Propriété 2

- Les clés d'un dictionnaire sont uniques. Ce sont donc des objets non mutables (de type `int`, `float`, `str`, `tuple`).
- Les valeurs d'un dictionnaires peuvent être multiples et mutables ou non mutables (tous les types sont acceptés).

Exemple 2 (Dictionnaire avec des valeurs simples et des valeurs multiples)

```
>>> simpson = {"père" : "Homer" , "mère" : "Marge", "fils" : "Bart", "fille" : "Lisa"}
>>> dressing = {"chemise" : ["M", "Bleu", "tiroir du bas"],
    "pantalon" : ["M", "Vert", "tiroir du milieu"]}
```

II. Accéder à un élément du dictionnaire

Pour accéder à la valeur correspondant à la clé, il suffit de placer cette clé entre crochets [ ] : `dico[clé]`.

Exemple 3 (Accès aux valeurs du dictionnaire *Simpson*)

```
>>> simpson["père"]
'Homer'
>>> simpson["mère"]
'Marge'
>>> for cle in simpson :
        print(simpson[cle])
Homer
Marge
Bart
Lisa
```

III. Création et modification d'un dictionnaire

III. 1. Création d'un dictionnaire

III. 1. a. Création par instanciation

C'est la méthode la plus simple. Elle consiste à créer le dictionnaire en utilisant le symbole d'affectation « = », et en saisissant les éléments les uns après les autres.

Exemple 4 (Création du dictionnaire *Simpson* par instanciation)

```
>>> simpson = {"père" : "Homer" , "mère" : "Marge", "fils" : "Bart", "fille" : "Lisa"}
```

Activité 1.
Créez un dictionnaire appelé *annee* dans lequel la clé correspond au nom du mois, et la valeur sera celle du nombre de jours dans le mois. On considèrera une année non bissextile.

III. 1. b. Création par compréhension

Cette méthode ressemble à celle utilisée pour remplir les tableaux. Au lieu d'indiquer les crochets du tableau [ ], on utilise les accolades du dictionnaire { }.

Exemple 5 (Création du dictionnaire *carre* par compréhension)

```
>>> carre = {a: a**2 for a un range(0,11,1)}
```

III. 2. Ajouter des éléments à un dictionnaire

Pour ajouter un ensemble clé/valeur à un dictionnaire, on utilise l'instruction : `dico[clé] = valeur`

Exemple 6 (Ajout des clés "pseudo" et "mot de passe" au dictionnaire *user*)

```
>>> user["pseudo"] = "Bond"
>>> user["mot de passe"] = "007"
```

Propriété 3

Si la clé n'existe pas, elle est ajoutée au dictionnaire avec la valeur spécifiée par le signe « = ».
Si la clé existe déjà, l'ancienne valeur est remplacée par celle spécifiée par le signe « = ».

Exemple 7 (La valeur de la clé "mot de passe" est remplacée par "Licence to kill")

```
>>> user["pseudo"] = "Bond"
>>> user["mot de passe"] = "Licence to kill""
```

Activité 2.
Modifiez le dictionnaire *annee* pour tenir compte d'une année bisextile

III. 3. Supprimer des éléments à un dictionnaire

Pour supprimer un ensemble clé/valeur à un dictionnaire, on utilise l'instruction `del dico[clé]`.

Exemple 8 (Supprimer la "mot de passe" au dictionnaire *user*)

```
>>> del user["mot de passe"]
```

IV. Les méthodes .keys(), .values() et .items()

La méthode `.keys()` permet de générer l'ensemble de la liste des clés d'un dictionnaire. C'est une méthode fréquemment utilisée pour parcourir un dictionnaire.

Exemple 9 (Exemple d'utilisation de la méthode .keys() sur dictionnaire *user*)

```
>>> for cle in user.keys() :
        print(cle)
pseudo
mot de passe
```

La méthode `.values()` permet de générer l'ensemble des valeurs d'un dictionnaire. C'est une méthode fréquemment utilisée pour parcourir un dictionnaire.

Exemple 10 (Exemples d'utilisation de la méthode .values() sur dictionnaire *user*)

```
>>> for valeur in user.values() :
        print(valeur)
Bond
007
```

La méthode `.items()` permet de retrouver la liste des ensembles {clé + valeur} d'un dictionnaire.

Exemple 11 (Exemples d'utilisation de la méthode .items() sur dictionnaire *user*)

```
>>> for couple in user.items() :
        print(couple)
('pseudo', 'Bond')
('mot de passe', '007')
```

L'instruction `in` permet de tester si un élément est une clé, une valeur ou un ensemble clé/valeur.

Exemple 12 (Tests d'appartenance au dictionnaire *user*)

```
>>> "pseudo" in user.keys()
True
>>> "Bond" in user.keys()
False
>>> "Bond" in user.values()
True
>>> "pseudo" in user.values()
False
>>> ('pseudo', 'Bond') in user.items()
True
>>> ('pseudo', 'mot de passe') in user.values()
False
```

Je retiens

- ▷ Un dictionnaire est un ensemble non-ordonné d'associations clé/valeur.
- ▷ Chaque élément d'un dictionnaire est composé d'une clé et d'une valeur (clé : valeur). Chaque élément est séparé par une virgule. L'ensemble du dictionnaire est séparé par { }.
- ▷ Chaque clé d'un dictionnaire est unique et non mutable. Seules les valeurs sont mutables.
- ▷ Pour accéder à un élément du dictionnaire, on utilise l'expression `dico[cle]`.
- ▷ La méthode `.keys()` permet d'itérer sur les clés d'un dictionnaire.
- ▷ La méthode `.values()` permet d'itérer sur les vlaleurs d'un dictionnaire.
- ▷ La méthode `.items()` permet d'itérer sur les ensembles clé/valeur d'un dictionnaire.

V. Exercices

Exercice 1 (Vrai - Faux).
Soit d un dictionnaire vide. L'instruction `v = d["un"]` provoque une erreur :

- ☐ Vrai
- ☐ Faux

Exercice 2 (Vrai - Faux).
Soit d un dictionnaire dont les clés sont des noms de pays, et les valeurs sont leur capitale. Si la France n'est pas dans le dictionnaire, l'instruction `d["France"] = "Paris"` provoque une erreur :

- ☐ Vrai
- ☐ Faux

Exercice 3 (Vrai - Faux).
Soit d un dictionnaire vide. L'instruction `d[[10, 15] = 5]` provoque une erreur :

- ☐ Vrai
- ☐ Faux

Exercice 4 (QCM).
On considère le dictionnaire `d : {"if":"si", "yes":"oui", "no":"non"}` et le morceau de code suivant :

```
1 for c in d.keys():
2     print(c)
```

Qu'obtient-on dans l'interpréteur ?

- ☐ L'affichage de if, yes et no.
- ☐ L'affichage de si, oui et non.
- ☐ L'affichage des couples ('if', 'si), ('yes', 'oui'), ('no', 'non').
- ☐ Une erreur.

Exercice 5.
On considère le dictionnaire créé par l'instruction suivante : `>>> d = {"nom": "Dupuis", "prenom": "Jacque", "age": 30}`.
Indiquez le code Python permettant de répondre aux questions suivantes :

1. Corrigez l'erreur dans le prénom. La bonne valeur est "Jacques".
2. Affichez la liste des clés du dictionnaire.
3. Affichez la liste des valeurs du dictionnaire.
4. Affichez la liste des paires clé/valeur du dictionnaire.
5. Écrivez la phrase : "Jacques Dupuis a 30 ans" en utilisant le dictionnaire.

Exercice 6.
On souhaite créer un dictionnaire français - anglais. Pour cela, nous allons créer le dictionnaire *traduction*, dont chaque clé sera constitué d'un mot en français, et chaque valeur sera constituée de sa traduction en anglais.
Indiquez le code Python permettant de répondre aux questions suivantes :

La suite page suivante

Suite de l'exercice

1. Choisissez 5 mots de la langue française, et créez le dictionnaire *traduction*.
2. Ajoutez une entrée au dictionnaire de la question précédente.
3. Écrivez une fonction *anglais*() prenant *dict* en argument et qui affiche, ligne par ligne, tous les mots en anglais du dictionnaire.
4. Écrivez une fonction *francais*() prenant *dict* en argument et qui affiche, ligne par ligne, tous les mots en français du dictionnaire.
5. Écrivez une fonction *ajoute*() prenant *mot*1, *mot*2 et *dict* comme arguments (*mot*1 représente le mot en français, *mot*2 représente le mot en anglais, et *dico* représente le dictionnaire à modifier), et ajoute cet ensemble dans le dictionnaire uniquement si la clé n'est pas présente dans le dictionnaire.
6. Écrivez une fonction *supprime*() prenant *car* et *dict* comme arguments et qui supprime du dictionnaire toutes les entrées dont la clé commence par la lettre *car*.

Exercice 7.

On souhaite représenter le parc de vélo en libre accès à Rouen, Velo'R. Pour cela, on utilise le dictionnaire *engin* basé sur les associations clé/valeur suivantes :

- "id" qui correspond à l'identifiant unique du vélo (l'identifiant est un entier),
- "type" qui peut prendre la valeur « électrique » ou « classique »,
- "statut" qui peut prendre la valeur « en déplacement », « en panne » ou « dispo »,
- "station" qui correspond à la station où est garé le vélo. Dans le cas où le vélo est en déplacement, "station" correspond à celle où il a dernièrement stationné.

Indiquez le code Python permettant de répondre aux questions suivantes :

1. Créez le dictionnaire vide *engin*.
2. Entrez les ensembles clés/valeurs pour le vélo électrique en panne d'identifiant 121000, stationné au Palais de Justice.
3. Ce vélo est maintenant réparé. Modifiez le dictionnaire en conséquence.
4. Suite à la réparation, il est maintenant en circulation. Modifiez le dictionnaire en conséquence.
5. Quelle morceau de code permettrait de renvoyer `True` si le vélo 121000 est disponible, et False dans le cas contraire.

On souhaite maintenant gérer l'intégralité des vélos du service Vélo'R. Pour cela, on souhaite créer un tableau *veloR*, où chaque élément du tableau sera un dictionnaire. *veloR* contiendra ainsi : ["id" :xxx,"type" :"xxx","statut" :"xxx", "xxx" :"xxx", . . . , . . .].

On saisit la suite d'instructions suivante :

```
>>> veloR[0] = {"id":121000,"type":"électrique","statut":"en déplacement",
    "station":"Palais de Justice"}
>>> veloR[1] = {"id":121027,"type":"électrique","statut":"en déplacement",
    "station":"Palais de Justice"}
>>> veloR[2] = {"id":121042,"type":"classique","statut":"dispo", "station":"Gare SNCF"}
>>> veloR[3] = {"id":121003,"type":"électrique","statut":"en déplacement",
    "station":"Gare SNCF"}
>>> veloR[4] = {"id":121017,"type":"électrique","statut":"dispo",
    "station":"Palais de Justice"}
>>> veloR[5] = {"id":121025,"type":"classique","statut":"en panne",
    "station":"Palais de Justice"}
>>> veloR[6] = {"id":121007,"type":"électrique","statut":"dispo",
    "station":"Belges"}
>>> veloR[7] = {"id":121055,"type":"électrique","statut":"dispo",
    "station":"Palais de Justice"}
```

6. Que renvoie l'instruction `veloR[5]` ?
7. Quelle séquence d'instructions permettrait d'afficher l'identifiant de tous les vélos disponibles ?
8. Quelle séquence d'instructions permettrait d'afficher l'identifiant de tous les vélos en panne ?
9. Quelle séquence d'instructions permettrait d'afficher l'identifiant de tous les vélos en déplacement ?
10. Quelle séquence d'instructions permettrait d'afficher l'identifiant et le statut de tous les vélos stationnés à la station Palais de Justice ?