

Chapitre (S1) 4 Dictionnaires

1	Définition	Objectifs
2	Parcours d'un dictionnaire	● Découvrir et savoir utiliser un dictionnaire.
3	Opérations élémentaires sur un dictionnaire	
4	Copie d'un dictionnaire	

1 DÉFINITION

Nous avons découvert dans un chapitre précédent les listes : ce sont des objets ordonnés indexés par une liste d'entiers successifs : 0,1,2,... Par exemple, si `L = [125, 'a', 'toto', 158]`, alors :

- `L[0] = 125`,
- `L[1] = 'a'`,
- `L[2] = 'toto'`
- `L[3] = 158`.

Un dictionnaire permet aussi de regrouper des valeurs, mais contrairement aux listes, les indices peuvent ne pas être des entiers. Ces indices sont appelés des **clés**, les éléments correspondants sont appelés des **valeurs**. La création d'un dictionnaire peut se faire en fournissant entre accolades une liste de termes de la forme clé : valeur. Par exemple l'instruction :

```
>>> D = {'abricot' : 5 , 'ananas' : 2 , 'pomme' : 8}
```

crée un dictionnaire stocké dans la variable `D` et contenant les trois valeurs 5, 2 et 8, qui sont indicées respectivement par les clés `'abricot'`, `'ananas'` et `'pomme'`. La valeur correspondant à une clé donnée s'obtient comme pour les listes :

```
>>> D['ananas']
2
```

Il est bien sûr possible de modifier la valeur associée à une clé :

```
>>> D['abricot'] = 10
```

On peut ajouter une nouvelle clé avec la valeur associée de la manière suivante :

```
>>> D['cerise'] = 15
```

Notons que dans cet exemple les clés sont des chaînes de caractères, mais elles auraient pu être d'un autre type (les clés pouvant même ne pas être toutes du même type), à l'exception des listes.

La seule contrainte est que les noms choisis pour les clés soient distincts deux à deux (sans quoi il y aurait ambiguïté sur la valeur associée à cette clé).

Exercice 1 [Sol 1] Créer un dictionnaire dont les clés sont les 26 lettres minuscules de l'alphabet et la valeur associée à chaque clé est sa place dans l'alphabet :

$$D = \{ 'a':1 , 'b':2 , \dots \}$$

Pour ne pas avoir à rentrer manuellement tous les éléments du dictionnaire, on pourra utiliser le codage ASCII (prononcer ASKI) et notamment la fonction `chr`. Par exemple, la commande `chr(97)` renvoie la lettre minuscule `'a'`, `chr(98)` renvoie la lettre minuscule `'b'`. Le lecteur curieux pourra faire une recherche dans son moteur de recherche préféré en tapant « Codage ASCII ».

2 PARCOURS D'UN DICTIONNAIRE

Il est possible de parcourir les données stockées dans un dictionnaire en itérant

- soit sur les clés :

```
>>> for c in D.keys() :
...     print(c)
...
abricot
ananas
pomme
```

- soit sur les valeurs :

```
>>> for v in D.values():
...     print(v)
...
5
2
8
```

- soit sur les couples clé,valeur :

```
>>> for c,v in D.items():
...     print(c)
...     print(v)
...
abricot
5
ananas
2
pomme
8
```

- Notons enfin que l'on peut boucler sur les **clés** avec la syntaxe simplifiée :

```
>>> for c in D:
...     print(c)
...
abricot
ananas
pomme
```

Attention
Il est important de noter que les éléments d'un dictionnaire n'ont pas à être ordonnés.

Cela ne fait pas partie des caractéristiques exigées d'un dictionnaire. Donc, pour le parcours ou l'affichage des clés ou des valeurs, il n'y a aucun ordre prévisible!

Exercice 2 [Sol 2] On considère deux dictionnaires D1 et D2, et on suppose que les clés de D1 sont différentes des clés de D2. Créer un dictionnaire D obtenu en conca-

ténant les deux dictionnaires D1 et D2. Expliquer quelle difficulté on rencontrerait si des clés étaient communes à D1 et D2.

3 OPÉRATIONS ÉLÉMENTAIRES SUR UN DICTIONNAIRE

- Créer un dictionnaire vide :

```
>>> D = {}
>>> type(D)
<class 'dict'>
```

- Connaître la longueur d'un dictionnaire :

```
>>> D = {'toto':2 , 'tata':3 , 'titi':5}
>>> len(D)
3
```

- Supprimer un élément du dictionnaire :

```
>>> D = {'toto':2 , 'tata':3 , 'titi':5}
>>> del(D['tata'])
```

- Supprimer et renvoyer un élément du dictionnaire :

```
>>> D = {'toto':2 , 'tata':3 , 'titi':5}
>>> D.pop('titi')
5
```

- Savoir si une clé ou une valeur appartiennent à un dictionnaire :

```
>>> D = {'toto':2 , 'tata':3 , 'titi':5}
>>> 'tata' in D    # Est-ce que 'tata' est une clé de D ?
True
>>> 'tata' in D.keys() # Même question
True
>>> 7 in D.values() # Est-ce que 7 est une valeur de D ?
False
```

Exercice 3 [Sol 3] Créer une fonction `CompteLettres` à qui on fournit une chaîne de caractères et qui renvoie un dictionnaire dont les clés sont les lettres présentes dans la chaîne, et pour chaque clé, la valeur est le nombre de fois où la lettre apparaît. Par exemple, `CompteLettres('abracadabra')` renverrait le dictionnaire

```
D = { 'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1 }
```

Exercice 4 [Sol 4] On considère dans cet exercice deux dictionnaires D1 et D2 dont les valeurs sont toutes des nombres réels. Améliorer l'exercice 2 pour pouvoir concaténer les deux dictionnaires D1 et D2. Lorsqu'une clé figure dans chacun des deux dictionnaires, on fera la somme des deux valeurs.

Exercice 5 [Sol 5] On considère un dictionnaires D dont les éléments sont de la forme NOM : NOTE, où NOM est le nom d'un élève et NOTE est sa moyenne générale (sur 20).

1. Créer deux dictionnaires D1 et D2 où D1 (respectivement D2) est la partie de D dont les clés sont les noms des élèves ayant une note <10 (respectivement ≥ 10).
2. Calculer la moyenne des notes à ce devoir.
3. Calculer la note maximal.
4. Un professeur peu scrupuleux décide de doubler les notes des élèves dont le nom commence par un 'A'. Modifier le dictionnaire pour qu'il corresponde à cette pratique. *Si la note doublée dépasse 20, on indiquera la note de 20.*

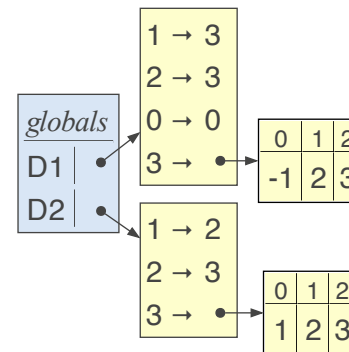
```
>>> D1 = {1 : 2, 2: 3, 3:[1, 2, 3]}
>>> from copy import deepcopy
>>> D2 = deepcopy(D1)
>>> D1[0] = 0
>>> D1[3][0] = 0
>>> D1[1] = 3
>>> D1[3][0] = -1
>>> D1
{1: 3, 2: 3, 3: [-1, 2, 3], 0: 0}
>>> D2
{1: 2, 2: 3, 3: [1, 2, 3]}
```

4

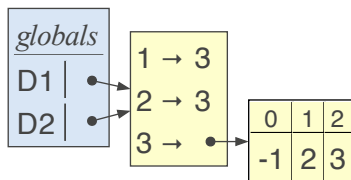
COPIE D'UN DICTIONNAIRE

Le problème est similaire à celui rencontré déjà pour les listes :

```
>>> D1 = {1 : 2, 2: 3, 3:[1, 2, 3]}
>>> D2 = D1
>>> D1[1] = 3
>>> D1[3][0] = -1
>>> D1
{1: 3, 2: 3, 3: [-1, 2, 3]}
>>> D2
{1: 3, 2: 3, 3: [-1, 2, 3]}
```



Cette fois-ci les deux dictionnaires sont bien indépendants.



Les modifications faites sur D1 sont donc répercutées sur D2.

Comme pour les listes, nous utiliserons la méthode ci-dessous, fonctionnelle pour n'importe quel dictionnaire.

Solution 1

■ ■ Un premier dictionnaire

```
D = {}
for k in range(1, 27):
    D[chr(96+k)] = k
>>> D
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8, \
↳ 'i': 9, 'j': 10, 'k': 11, 'l': 12, 'm': 13, 'n': 14, 'o': 15, \
↳ 'p': 16, 'q': 17, 'r': 18, 's': 19, 't': 20, 'u': 21, 'v': 22, \
↳ 'w': 23, 'x': 24, 'y': 25, 'z': 26}
```

Solution 2

■ ■ Concaténation de deux dictionnaires

```
D = {}
for c in D1:
    D[c] = D1[c]
for c in D2:
    D[c] = D2[c]
```

Les clés d'un dictionnaire devant être deux à deux distinctes, il aurait été problématique de rencontrer deux clés identiques, l'une dans D1 et l'autre dans D2.

Solution 3

■ ■ Comptage de lettres

```
def ComptageLettres(ch:str)->dict:
    D = {}
    for c in ch:
        if c in D:
            D[c] = D[c]+1
        else:
            D[c] = 1
    return D
```

Solution 4

■ ■ Concaténation de deux dictionnaires

```
D = {}
for c in D1:
    D[c] = D1[c]
for c in D2:
    if c in D:
        D[c] = D[c]+D2[c]
    else:
        D[c] = D2[c]
```

Solution 5

■ ■ Partition d'un dictionnaire

```
D1 = {}
D2 = {}
for nom in D:
    if D[nom] < 10:
        D1[nom] = D[nom]
    else:
        D2[nom] = D[nom]
```

■ ■ Moyenne

```
S = 0
for nom in D:
    note = D[nom]
    S += note
moy = S/len(L)
```

■ ■ Note max

```
max = 0
for nom in D:
    note = D[nom]
    if note > max:
        max = note
```

■ ■ Truquage

```
for nom in D:
    if nom[0] == 'A':
        D[nom] = min(2*D[nom], 20)
```