Les Dictionnaires

Université Virtuelle de Côte d'Ivoire

Table des matières

Objectifs

A la fin de cette leçon l'apprenant sera capable de :

- Décrire le concept de dictionnaire
- Réaliser des opérations sur les dictionnaires
- Écrire des scripts python avancés

Introduction

Un dictionnaire est un type de données extrêmement puissant et pratique. Il se rapproche des listes sur certains points mais, sur beaucoup d'autres, il en diffère totalement. Python utilise ce type pour représenter diverses fonctionnalités : on peut par exemple retrouver les attributs d'un objet grâce à un dictionnaire particulier.

Cette leçon présente succinctement les fonctionnalités d'un dictionnaire et les méthodes pour le parcourir.

Introduction à l'objet dictionnaire

I

Objectifs

- Décrire le concept de dictionnaire
- Faire des manipulations de base sur un dictionnaire

1. Dictionnaire

Les dictionnaires sont des objets pouvant en contenir d'autres. A la différence des listes et des tuples, les dictionnaires sont créés en associant à chaque objet une clé. Par exemple, un dictionnaire peut contenir un carnet de contact et on accède à chaque contact en précisant son nom.

Syntaxe : Création d'un dictionnaire

un dictionnaire contient des éléments constitués d'une paire (clé, valeur). Chaque élément du dictionnaire est séparé de l'autre par une virgule.

```
1 dictionnaire = dict()  #permet de créer un dictionnaire vide
2 diction = {}
3 diction[clé] = valeur  #permet d'ajouter un élément au dictionnaire
4
5 dictionnaire = {clé1:valeur1, ..., clén:valeurn} #permet de créer un
  dictionnaire et de l'initialiser avec n élément
```

Exemple

```
>>> repertoire = {}
>>> type(repertoire)
<class 'dict'>
>>> repertoire["Moustapha"] = '07 95 24 15'
>>>
>>> repertoire["Kouadio"] = '23 50 50 93'
>>>
>>> print(repertoire)
{'Moustapha': '07 95 24 15', 'Kouadio': '23 50 50 93'}
>>>
>>> print(repertoire["Moustapha"]
... )
07 95 24 15
>>> print(repertoire["Kouadio"])
23 50 50 93
>>>
```

Création et édition d'un dictionnaire sous la console python 3.7

2. La paire clé-valeur

En créant un dictionnaire, les clés doivent être des objets *immuables* et les valeurs peuvent contenir n'importe quel objet. Les seuls objets *mutables* que nous avons vu sont : les listes et maintenant les dictionnaires

Exemple

```
>>> coq_a_lane = {1:"John Jay", "rue":"VGE", 2:True}
>>> print(coq_a_lane[1])
John Jay
>>> print(coq_a_lane)
{1: 'John Jay', 'rue': 'VGE', 2: True}
>>>
```

manipulation d'un dictionnaire sous la console de python 3.7

3. supprimer les clés d'un dictionnaire

Il existe deux possibilités:

- le mot-clé *del* qui fonctionne pareillement avec les listes
- la méthode de dictionnaire *pop*

En supprimant la clé, la valeur associée à la clé est automatiquement supprimée.

Exemple

```
1 grande_famille = {"homme":18, "femme":6, "enfant":30}
2 del grande_famille["homme"]
3 grande_famille.pop("homme")
```

Les deux méthodes permettent de supprimer la clé `homme`

4. accès à un élément du dictionnaire

La syntaxe `dictionnaire["cle"]` permet d'accéder à un élément du dictionnaire par sa clé. Si la clé n'existe pas dans le dictionnaire, un message d'erreur de type KeyError apparaitra.

Python met à disposition une méthode utile pour accéder à un élément du dictionnaire par sa clé : la méthode *get*. Cette méthode si elle ne retrouve pas la clé spécifié dans un dictionnaire *retourne la valeur none par defaut* plutôt qu'un message d'erreur. la syntaxe suivante montre l'utilisation de la méthode get :

```
dictionnaire.get(cle, msg)
```

Où *cle* est la clé dont on voudrait afficher la valeur et *msg* est le message à afficher si la clé n'existe pas dans le dictionnaire. Le paramètre *msg* est facultatif

Exemple

```
>>> print(repertoire)
{'Moustapha': '07 95 24 15', 'Kouadio': '23 50 50 93', 'Michel': '69 58 24 74'}
>>> print(repertoire.get('Michel'))
69 58 24 74
>>> print(repertoire.get("Roland", "Roland n'existe pas dans le repertoire"))
Roland n'existe pas dans le repertoire
>>>
>>> print(repertoire['Roland'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Roland'
>>>
```

Accès à un élément d'un dictionnaire

Remarque

Pour déterminer si une clé est dans un dictionnaire, les mots clés *in* et *not in* peuvent être utilisés

5. Exercice

[solution n°1 p.12]

Remplir les trous pour définir un dictionnaire valide avec deux éléments

>>> voiture = {"Mercedes" "Kompressor" "BMW" "X5"}

6. Exercice

[solution n°2 p.12]

Laquelle de ces valeurs ne peut être utilisée comme une clé de dictionnaire

- {2 : 4, 3 : 6}
- "un deux trois"
- True

7. Exercice

[solution n°3 p.12]

Quel est le résultat de ce script

```
1 primes = {1:2, 2:3, 4:7, 7:17}
2 print(primes[primes[4]])
```

8. Exercice

[solution n°4 p.12]

Quel est le résultat de ce script ?

```
1 var = {1:1, 2:1, 3:2, 4:3}
2 print(var.get(4, 0) + var.get(7, 5))
```

Méthodes de parcours de dictionnaire

II

Objectifs

- Écrire des scripts pour parcourir les dictionnaires.
- Utiliser des méthodes pour manipuler les couples clé-valeur d'un dictionnaire
- Réaliser des scripts python avancés

1. Parcours par clé

Les dictionnaires contrairement aux listes ne possèdent pas de structure ordonnée. Ainsi le parcours du dictionnaire se fait parfois par le parcours des clés

Exemple

```
>>> print(coq_a_lane)
{1: 'John Jay', 'rue': 'VGE', 2: True}
>>> for cle in coq_a_lane:
...     print(cle)
...
1
rue
2
>>>
```

parcours du dictionnaire coq_a_lane par clé

Remarque

La méthode *keys()* appliquée à un dictionnaire retourne un objet qui contient la liste des clés du dictionnaire. Dans l'exemple précédent la syntaxe aurait été :

```
1 for cle in coq_a_lane.keys():
```

2. parcours par valeur

la méthode values()

La méthode values() appliquée à un dictionnaire retourne un objet contenant la liste des valeurs du dictionnaire

Exemple

```
>>> for valeur in coq_a_lane.values():
...     print(valeur)
...
John Jay
VGE
True
>>>
```

parcours d'un dictionnaire par valeur

3. Parcours par clé, valeur simultanément

La méthode items()

La méthode items() appliquée à un dictionnaire retourne une liste de tuples. Chaque tuple de cette liste est composé d'une paire clé, valeur du dictionnaire. Elle est semblable à la méthode enumerate() des listes

Exemple

```
>>> for cle, valeur in coq_a_lane.items():
...     print("la clé {} correspond à la valeur {}".format(cle, valeur))
...
la clé 1 correspond à la valeur John Jay
la clé rue correspond à la valeur VGE
la clé 2 correspond à la valeur True
>>>
```

parcours par clé, valeur d'un dictionnaire

4. Exercice

[solution n°5 p.12*]*

Compléter le script suivant pour réaliser un répertoire téléphonique

```
1. def repertoire(dictionnaire) :
...     cle = input("donnez le nom du contact : ")
...    if cle != 'Q' :
...         valeur = input("donnez le numéro du contact")
...     else :
...         return 'Q'
...     dictionnaire[                ] =
....
```

5. Exercice

[solution n°6 p.13]

En se basant sur le code précédent, donnez le résultat de ce script

```
>>> while repertoire(rep) != 'Q':
...     print("insertion")
...
donnez le nom du contact yves
donnez le numero du contact01 05 06 07
insertion
donnez le nom du contactvictoire
donnez le numero du contact45 78 47 76
insertion
donnez le nom du contactpapa
donnez le numero du contact07 98 70 02
insertion
donnez le nom du contactQ
>>> print(rep)
```

{' ' : ' ', ' ' : ' ', ' ' : ' '
'}

6. Exercice

[solution n°7 p.13]

Écrire le script pour parcourir le dictionnaire *rep* afin d'afficher la liste de tous les contacts

for , in rep.items() :

... print("le numéro {} est celui de {}".format(numero, nom))

Conclusion

Au terme de cette leçon, l'apprenant a su l'utilité du concept de dictionnaire dans la mise en place de programme avancé tel que la mise en place d'annuaire téléphonique. La prochaine leçon est destinée à l'enregistrement de données dans les fichiers.

Solutions des exercices

> **Solution** n°1

Exercice p. 7

Remplir les trous pour définir un dictionnaire valide avec deux éléments

>>> voiture = {"Mercedes" : "Kompressor", "BMW" : "X5"}

> **Solution** n°2

Exercice p. 7

Laquelle de ces valeurs ne peut être utilisée comme une clé de dictionnaire

- ◉ {2 : 4, 3 : 6}
- ○ "un deux trois"
- ○ True

La clé d'un dictionnaire doit être un objet immuable or un dictionnaire est mutable

> **Solution** n°3

Exercice p. 7

Quel est le résultat de ce script

```
primes = {1:2, 2:3, 4:7, 7:17}
print(primes[primes[4]])
```

17

> **Solution** n°4

Exercice p. 7

Quel est le résultat de ce script ?

```
var = {1:1, 2:1, 3:2, 4:3}
print(var.get(4, 0) + var.get(7, 5))
```

8

> **Solution** n°5

Exercice p. 9

Compléter le script suivant pour réaliser un répertoire téléphonique

1. def repertoire(dictionnaire) :

... cle = input("donnez le nom du contact : ")

... if cle != 'Q' :

... valeur = input("donnez le numéro du contact")

... else :

... return 'Q'

... dictionnaire[`cle`] = `valeur`

....

> **Solution** n°6

Exercice p. 10

En se basant sur le code précédent, donnez le résultat de ce script

```
>>> while repertoire(rep) != 'Q':
...     print("insertion")
...
donnez le nom du contact yves
donnez le numero du contact01 05 06 07
insertion
donnez le nom du contactvictoire
donnez le numero du contact45 78 47 76
insertion
donnez le nom du contactpapa
donnez le numero du contact07 98 70 02
insertion
donnez le nom du contactQ
>>> print(rep)
```

{'`yves`' : '`01 05 06 07`', '`victoire`' : '`45 78 47 76`', '`papa`' : '`07 98 70 02`'}

> **Solution** n°7

Exercice p. 10

Écrire le script pour parcourir le dictionnaire *rep* afin d'afficher la liste de tous les contacts

for `nom`, `numero` in rep.items() :

... print("le numéro {} est celui de {}".format(numero, nom))