Python : les dictionnaires

Présentation

Les **dictionnaires** permettent de mémoriser des informations en les associant à une **clé** qui est la plupart du temps une chaîne de caractères. Par exemple, un dictionnaire peut contenir un carnet d'adresses et on accède à chaque contact en précisant son nom.

Création d'un dictionnaire

Les parenthèses **()** délimitent les tuples, les crochets **[]** délimitent les listes et les accolades **{}** délimitent les dictionnaires.

Création d'un dictionnaire

Première méthode :

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

Deuxième méthode :

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

Ajout de valeurs

Pour **ajouter** une valeur, il faut préciser la **clé** et la valeur **associée** :

```
>>> carnet = {}
>>> carnet["nom"] = "Dupond"
>>> carnet["prenom"] = "Jean"
>>> carnet
{'nom': 'Dupond', 'prenom': 'Jean'}
>>>
```

- on indique entre crochets la clé à laquelle on souhaite accéder.
- Si la clé n'existe pas, elle est ajoutée au dictionnaire avec la valeur spécifiée après le signe =.
- Sinon, l'ancienne valeur à l'emplacement indiqué est remplacée par la nouvelle.

Accès à une valeur

Pour accéder à une valeur, on utilise la clé associée :

```
>>> carnet["nom"]
'Dupond*'
>>>
```

Si la **clé n'existe pas** dans le dictionnaire, cela génère une **erreur**. Il est possible de tester au préalable l'existence d'une clé de cette façon :

```
>>> "nom" in carnet
True
>>> "adresse" in carnet
False
```

On peut utilisé comme clé, des chaînes de caractères (comme précédemment), des entiers, des listes ou les toris à la fois selon les besoins.

Par exemple :

- utilisation d'entiers pour les clés :

```
>>> mon_dictionnaire = {}
>>> mon_dictionnaire[0] = "a"
>>> mon_dictionnaire[1] = "e"
>>> mon_dictionnaire[2] = "i"
>>> mon_dictionnaire[3] = "o"
>>> mon_dictionnaire[4] = "u"
>>> mon_dictionnaire[5] = "y"
>>> mon_dictionnaire
{0: 'a', 1: 'e', 2: 'i', 3: 'o', 4: 'u', 5: 'y'}
>>>
```

- utilisation de tuples pour les clés :

```
echiquier = {}
echiquier[('a', 1)] = "tour blanche" # En bas à gauche de l'échiquier
echiquier[('b', 1)] = "cavalier blanc" # À droite de la tour
```

Création de dictionnaires préremplis

note = {“maths”:12, “anglais”:15, “histoire”:13}

Supprimer des clés d'un dictionnaire

- avec le mot-clé **del** :

```
del carnet["nom"]
```

- avec la méthode de dictionnaire **pop** ; qui renvoie alors la valeur supprimée :

```
carnet.pop("nom")
```

Parcourir un dictionnaire

- parcourir la liste de **clés** :

```
>>> carnet
{'nom': 'Dupond', 'prenom': 'Jean', ('prenom', 2): 'Jean dominique'}
>>> for cle in carnet:
...     print(cle)
...
nom
prenom
('prenom', 2)
>>>
```

> Les clés ne s'affichent pas forcément dans l'ordre dans lequel on les a entrées car les dictionnaires n'ont pas de structure ordonnée.

- Parcours des **valeurs** du dictionnaire avec la méthode **values()** :

```
>>> for valeur in carnet.values():
...     print(valeur)
...
Dupond
Jean
Jean dominique
>>>
```

La méthode values() permet aussi de faire des tests d'existence d'une valeur dans un dictionnaire :

```
>>> if "Dupond" in carnet.values():
```

```
...     print("Le nom Dupond est déjà présent dans le carnet d'adresse.")
...
Le nom Dupond est déjà présent dans le carnet d'adresse.
>>>
```

Parcours des clés et valeurs simultanément

Pour obtenir à la fois la liste des clés et leur valeur, on utilise la méthode **items()** qui renvoie une liste de tuples contenant couples **clé : valeur**.

```
>>> for cle,valeur in carnet.items():
...     print("La clé {} contient la valeur {}.".format(cle, valeur))
...
La clé nom contient la valeur Dupond.
La clé prenom contient la valeur Jean.
La clé ('prenom', 2) contient la valeur Jean dominique.
>>>
```

Retour au cours : Les instructions du langage Python

- Cours : Les instructions du langage Python