Définition et création

Le dictionnaire Python est une structure de donnée régie par l'association clé - valeur. Il est l'équivalent du tableau associatif en PHP, ou de la Map en javascript. Il peut avoir une syntaxe proche du Json lorsque ses clés sont des strings ce qui le rend très adapté pour le web.

Le dictionnaire est un objet **ordonné** et muable qui possède des clés qui doivent être uniques.

```
mon_dict = {
    "nom"   : "Bouchard",
    "prenom": "Gérard"
}
```

Accès aux valeurs d'un dictionnaire

2 méthodes s'offrent à nous. Nous pouvons utiliser l'opérateur d'accès `[]`, ou la méthode `get()`.

`dict[clé]`

Accède à la valeur du dictionnaire situé à la clé donnée. Soulève une erreur si la clé n'existe pas.

```
from datetime import date

mon_dict = {
    "nom"           : "Bouchard",
    "prenom"        : "Gérard",
    "admin"         : True,
    "date_naissance": date(1982, 5, 26),
    "nombre_adonnes": 123627,
    "enfants"       : [
        {
            "nom"   : "Bouchard",
            "prenom": "Timothée"
        }
    ]
}

print(mon_dict["admin"])                # True
print(mon_dict["enfants"][0]["prenom"]) # Timothée
```

`dict.get(clé, message)` Accès à la valeur du dictionnaire à la clé donnée. Renvoi `None` si la clé n'existe pas. Si une string est initialisée en second paramètre, renvoi cette dernière en cas de clé introuvable.

```
mon_dict = {
    0: "titi",
    1: "tata",
    2: "toto"
}

print(mon_dict.get(2, "La clé demandée n'existe pas !"))
print(mon_dict.get(3, "La clé demandée n'existe pas !"))
```

>  Conseil d'utilisation
>
> Utiliser `dict.get(clé)` au lieu de `dict[clé]` permet de boucler sur une liste de dictionnaires aux données inconsistantes sans risquer de planter la boucle si une clé n'existe pas dans l'un d'entre eux.

Modifier une valeur d'un dictionnaire

On utilise la même syntaxe que pour accéder à la valeur, c'est à dire via l'opérateur `[clé]`. Comme les dictionnaires sont des objets muables, la modification des clé sont persistantes.

```
mon_dict = {
    "a": True,
    "b": True,
    "c": False
}

mon_dict["a"] = False

print(mon_dict)
```

Ajout et suppression d'une clé

Ajouter une clé

Procéder aussi simplement que lorsqu'on veut modifier la valeur d'une clé préexistente. Ici, la clé n'existe pas mais sera créeer à la volée.

```
mon_dict = {
    "test": [1, 2, 3],
    "dev" : [4, 5, 6]
}

mon_dict["prod"] = [7, 8, 9]
print(mon_dict)
```

⚠️ Vous ne pouvez pas utiliser la méthode `dict.get(clé)` pour modifier la valeur d'une clé ou pour ajouter une nouvelle clé.

Supprimer une clé

Pour supprimer une clé, on utilise le mot clé `del`.

```
mon_dict = {
    1: {"nom": "Chahiri", "prenom": "Hamza"},
    2: {"nom": "Martin", "prenom": "Etienne"}
}

if 2 in mon_dict: # contrôler la présence de la clé pour éviter de soulever une erreur
    del mon_dict[2]

print(mon_dict)
```

Boucler sur un dictionnaire

1. Il est possible de boucler sur un dictionnaire directement avec une boucle `for`, et cela va, par défaut, **boucler sur les clés**.

```
mon_dict = {
    "a": True,
    "b": True,
    "c": False
}

for cle in mon_dict: # boucle sur les clés
    print(cle) # affiche la clé
    print(mon_dict[cle]) # affiche le contenu de la clé
```

2. L'objet dictionnaire propose deux méthodes pour retourner une liste de ses clés, et une liste de ses valeur : `dict.keys()` et `dict.values()`.

```
mon_dict = {
    "a": True,
    "b": True,
    "c": False
}

for key in mon_dict.keys(): # boucle sur les clés
    print(key)

for value in mon_dict.values(): # boucle sur les résultats
    print(value)
```

3. Enfin, la méthode `dict.items()` retourne une liste de tuple contenant la clé et la valeur pour chaque élément.

```
mon_dict = {
    "a": True,
    "b": True,
    "c": False
}

for item in mon_dict.items(): # récupère les données en tuples (key, value)
    print(item)

for key, value in mon_dict.items(): # Séparation de la clé et la valeur en 2
variables
    print(f"la clé {key} a pour valeur {value}")
```

Trier un dictionnaire

On peut utiliser les méthodes `sort()` et `sorted()` en passer en paramètre la liste de tuples obtenue à partir de la méthode `dict.items()`, et utiliser le paramètre `key` pour utiliser une fonction `lambda` de tri (nous verrons les fonctions lambda dans le chapitre consacré aux fonctions). Enfin, la liste de tuple triée obtenue peux être reconvertie en dictionnaire.

```
mon_dict = {
    "stat_1": 23423,
    "stat_2": 432,
    "stat_3": 7823,
    "stat_4": 212310,
    "stat_5": 70,
    "stat_6": 6439267
}

# Tri par défaut par ordre croissant
# x[1]: on trie par les valeurs car dans la liste de tuples obtenue avec
dict.items(), les valeur sont à l'indice 1 de chaque tuple
sorted_dict = sorted(mon_dict.items(), key=lambda x: x[1])
print(dict(sorted_dict))
```

Exercice: "17. Les dictionnaires"