PYTHON – Dictionnaires

Introduction

- Les dictionnaires sont des objets pouvant en contenir d'autres, comme les listes.
- Cependant, au lieu d'héberger des informations dans un ordre précis, ils **associent chaque objet contenu à une clé** (la plupart du temps, une chaîne de caractères)
- Donc pour accéder aux objets contenus dans le dictionnaire, on n'utilise pas des indices mais des clés, qui peuvent être de bien des types distincts

Créer un dictionnaire

première méthode d'instanciation d'un dictionnaire vide :

```
mon_dictionnaire = dict()
```

deuxième méthode d'instanciation d'un dictionnaire vide :

```
mon_dictionnaire = {}
```

Si on résume :

- les crochets délimitent les listes
- les accolades {} délimitent les dictionnaires.

La classe sur laquelle se construit un dictionnaire est **dict**.

```
mon_dictionnaire = {}
print(type(mon_dictionnaire))
print(mon_dictionnaire)
```

```
<class 'dict'>
{}
```

Ajouter des éléments dans un dictionnaire

Pour ajouter des clés et valeurs dans notre dictionnaire vide :

```
mon_dictionnaire = {}
mon_dictionnaire["pseudo"] = "Jean"
mon_dictionnaire["mot de passe"] = "***"
print(mon_dictionnaire)
```

```
{'mot de passe': '***', 'pseudo': 'Jean'}
```

Nous indiquons entre crochets la clé à laquelle nous souhaitons accéder (pseudo par exemple)

Si la clé n'existe pas, elle est ajoutée au dictionnaire avec la valeur spécifiée après le signe = (Jean par exemple)

Concernant les clés, on peut utiliser quasiment tous les types comme clés.

Au lieu d'utiliser des chaînes de caractères, vous pouvez utiliser des entiers :

```
mon_dictionnaire = {}
mon_dictionnaire[0] = "a"
mon_dictionnaire[1] = "e"
mon_dictionnaire[2] = "i"
mon_dictionnaire[3] = "o"
mon_dictionnaire[4] = "u"
print(mon_dictionnaire)
```

```
{0: 'a', 1: 'e', 2: 'i', 3: 'o', 4: 'u'}
```

On peut aussi créer des dictionnaires déjà remplis :

```
panier = {"pomme":3, "poire":6, "fraise":7}
```

On précise :

- la clé entre guillemets ,
- le signe deux points « : »
- la valeur correspondante.

On sépare les différents couples clé : valeur par une virgule.

C'est d'ailleurs comme cela que Python vous affiche un dictionnaire quand vous le lui demandez.

Modifier des éléments dans un dictionnaire

Un dictionnaire **ne peut pas contenir deux clés identiques** (mais rien n'empêche d'avoir deux valeurs identiques).

L'ancienne valeur à l'emplacement indiqué est remplacée par la nouvelle :

```
mon_dictionnaire = {}
mon_dictionnaire["pseudo"] = "Jean"
mon_dictionnaire["mot de passe"] = "***"
mon_dictionnaire["pseudo"] = "Arthur"
print(mon_dictionnaire)
```

```
'mot de passe': '***', 'pseudo': 'Arthur'}
```

La valeur 'Jean‘ pointée par la clé 'pseudo‘ a été remplacée, à la ligne 4, par la valeur 'Arthur'.

Accéder à un élément dans un dictionnaire

Pour accéder à la valeur d'une clé précise :

```
mon_dictionnaire["mot de passe"]
print(mon_dictionnaire)
```

```
'***'
```

Si la clé n'existe pas dans le dictionnaire, une exception de type **KeyError** sera levée.

Supprimer des clés d'un dictionnaire

Vous avez deux possibilités :

- le mot-clé del;

```
panier = {"pomme":3, "poire":6, "fraise":7}
del panier["pomme"]
print(mon_dictionnaire)
```

```
{"poire":6, "fraise":7}
```

- la méthode de dictionnaire pop.

```
panier = {"pomme":3, "pantalon":6, "tee shirt":7}
print(panier.pop("pomme"))
```

```
3
```

La méthode pop supprime la clé précisée, mais elle renvoie en plus la valeur supprimée :

Parcourir un dictionnaire

Parcours des clés

Pour parcourir les clés d'un dictionnaire, on a deux solutions :

```
fruits = {"pommes":21, "melons":3, "poires":31}
for cle in fruits:
    print(cle)
```

```
fruits = {"pommes":21, "melons":3, "poires":31}
for cle in fruits.keys():
    print(cle)
```

```
melons
poires
pommes
```

On parcourt la liste des clés contenues dans le dictionnaire.

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

Parcours des valeurs

On peut parcourir les valeurs contenues dans un dictionnaire.

Pour ce faire, on utilise la méthode **values** (« valeurs » en anglais).

```
fruits = {"pommes":21, "melons":3, "poires":31}
for valeur in fruits.values():
    print(valeur)
```

```
3
31
21
```

Parcours des clés et valeurs simultanément

Pour avoir en même temps les indices et les objets d'un dictionnaire, on utilise la méthode **items**.

Elle renvoie une liste, contenant les couples clé : valeur, sous la forme d'un tuple :

```
fruits = {"pommes":21, "melons":3, "poires":31}
for cle, valeur in fruits.items():
    print("La clé ", cle , " contient la valeur ", valeur)
```

```
La clé melons contient la valeur 3.
La clé poires contient la valeur 31.
La clé pommes contient la valeur 21.
```

Vérifier l'existence d'un élément dans un dictionnaire

on utilise une condition :

```
fruits = {"pommes":21, "melons":3, "poires":31}
if 21 in fruits.values():
    print("Un des fruits se trouve dans la quantité 21.")
```

```
Un des fruits se trouve dans la quantité 21.
```