Les dictionnaires en Python : Guide complet

Un dictionnaire en Python est une structure de données puissante qui permet de stocker des paires clé-valeur. Contrairement aux listes ou tuples, les éléments d'un dictionnaire ne sont pas ordonnés et sont accessibles via leurs clés uniques, plutôt que par un indice.

1. Création d'un dictionnaire

Voici comment créer un dictionnaire en Python :

```
# Dictionnaire vide
mon_dictionnaire = {}

# Dictionnaire avec des données
mon_dictionnaire = {
    "nom": "Alice",
    "âge": 25,
    "ville": "Paris"
}
```

2. Accéder aux valeurs

Pour accéder à une valeur, utilisez sa clé :

```
print(mon_dictionnaire["nom"])  # Affiche : Alice
```

Pour éviter une erreur si la clé n'existe pas, utilisez la méthode `get` :

```
print(mon_dictionnaire.get("pays", "Clé non trouvée"))  # Affiche : Clé non trouvée
```

3. Ajouter ou modifier des éléments

Vous pouvez ajouter une nouvelle paire clé-valeur ou modifier une clé existante :

```
# Ajouter une nouvelle clé
mon_dictionnaire["pays"] = "France"
```

```
# Modifier une clé existante
mon_dictionnaire["âge"] = 26
```

4. Supprimer des éléments

Pour supprimer une clé, utilisez `del` ou la méthode pop :

```
# Supprimer avec del
del mon_dictionnaire["ville"]

# Supprimer avec pop
age = mon_dictionnaire.pop("âge")
print(age)  # Affiche : 26
```

5. Parcourir un dictionnaire

Vous pouvez parcourir les clés, les valeurs ou les deux :

```
# Parcourir les clés
for clé in mon_dictionnaire:
    print(clé)

# Parcourir les valeurs
for valeur in mon_dictionnaire.values():
    print(valeur)

# Parcourir les paires clé-valeur
for clé, valeur in mon_dictionnaire.items():
    print(f"{clé}: {valeur}")
```

6. Méthodes utiles

Voici quelques méthodes courantes pour manipuler les dictionnaires :

- `keys()`: Retourne toutes les clés.
- `values()`: Retourne toutes les valeurs.
- `items()`: Retourne toutes les paires clé-valeur.

- `clear()`: Vide le dictionnaire.
- `copy()`: Crée une copie du dictionnaire.

Exemple :

```
clés = mon_dictionnaire.keys()
valeurs = mon_dictionnaire.values()
paires = mon_dictionnaire.items()
```

7. Dictionnaires imbriqués

Les dictionnaires peuvent contenir d'autres dictionnaires :

```
dictionnaire_complexe = {
    "personne1": {"nom": "Alice", "âge": 25},
    "personne2": {"nom": "Bob", "âge": 30}
}

print(dictionnaire_complexe["personne1"]["nom"])  # Affiche : Alice
```

8. Avantages des dictionnaires

- Accès rapide aux données grâce aux clés.
- Flexible pour stocker des données complexes.
- Idéal pour des associations clé-valeur.

Avec ces bases, vous êtes prêt à exploiter tout le potentiel des dictionnaires en Python ! 😊