

Un Dictionnaire en Python

Un **dictionnaire** en Python est une structure de données extrêmement puissante et flexible, utilisée pour stocker des collections de données sous forme de paires **clé-valeur**.

Qu'est-ce qu'un Dictionnaire ?

Un dictionnaire est défini par les propriétés suivantes :

1. **Non-ordonné (pour Python < 3.7) ou Ordonné (pour Python ≥ 3.7) :** L'ordre des éléments est maintenant préservé dans les versions récentes.
2. **Modifiable (Mutable) :** Vous pouvez ajouter, supprimer ou modifier des paires clé-valeur après la création du dictionnaire.
3. **Indexé par des Clés :** Contrairement aux listes qui sont indexées par des nombres entiers (0, 1, 2, ...), les dictionnaires sont indexés par des **clés**.
 - **Clés :** Doivent être uniques et **immuables** (chaînes de caractères, nombres, ou tuples).
 - **Valeurs :** Peuvent être de n'importe quel type de données (chaînes, nombres, listes, fonctions, ou même d'autres dictionnaires).

Syntaxe

Les dictionnaires sont délimités par des **accolades** ({ }) et les paires clé-valeur sont séparées par deux-points (:).

Python

```
mon_dictionnaire = {  
    "clé_1": "valeur_A",  
    "clé_2": 150,  
    "clé_3": [1, 2, 3]  
}
```

Comment Utiliser un Dictionnaire

1. Créer un Dictionnaire

Vous pouvez créer un dictionnaire vide ou pré-rempli :

Python

```
# Dictionnaire vide
utilisateur_vide = {}

# Dictionnaire pré-rempli (exemple d'un livre)
livre = {
    "titre": "L'Étranger",
    "auteur": "Albert Camus",
    "année": 1942,
    "pages": 160
}

print(livre)
```

2. Accéder aux Éléments

Vous accédez aux valeurs en utilisant la **clé** entre crochets (`[]`) :

Python

```
titre_livre = livre["titre"]
print(f"Le titre est : {titre_livre}")
# Sortie: Le titre est : L'Étranger

# **Attention :** Si la clé n'existe pas, cela provoque une
# erreur (KeyError).
# Pour éviter cela, utilisez la méthode .get() :
annee_publication = livre.get("année")
prix = livre.get("prix", "Clé non trouvée") # La seconde
# valeur est une valeur par défaut
print(annee_publication) # Sortie: 1942
print(prix)              # Sortie: Clé non trouvée
```

3. Modifier, Ajouter et Supprimer des Éléments

Modification / Ajout

Pour **ajouter** une nouvelle paire ou **modifier** une valeur existante, vous utilisez l'opérateur d'accès (`[]`) :

Python

```
# Ajouter une nouvelle paire clé-valeur
```

```

livre["prix"] = 9.99
print(livre)
# Sortie: {'titre': 'L'Étranger', ..., 'prix': 9.99}

# Modifier une valeur existante
livre["année"] = 1943
print(livre["année"])
# Sortie: 1943

```

Suppression

Vous pouvez utiliser le mot-clé `del` ou la méthode `.pop()` :

Python

```

# Supprimer une paire clé-valeur avec del
del livre["pages"]

# Supprimer une paire clé-valeur avec .pop() (retourne la
valeur supprimée)
prix_supprime = livre.pop("prix")
print(f"Prix supprimé : {prix_supprime}")

# Supprimer TOUTES les paires
# livre.clear()

```

4. Parcourir un Dictionnaire (Itération)

Vous pouvez parcourir les clés, les valeurs, ou les deux à l'aide de boucles `for`.

| Méthode | Itère sur... | Exemple d'itération |
|------------------------|---------------------------------|--|
| Par défaut | Les clés | <code>for cle in livre:</code> |
| <code>.keys()</code> | Les clés | <code>for cle in livre.keys():</code> |
| <code>.values()</code> | Les valeurs | <code>for valeur in livre.values():</code> |
| <code>.items()</code> | Les paires (clé, valeur) | <code>for cle, valeur in livre.items():</code> |

Python

```

# Itération sur les paires (clé, valeur)
for cle, valeur in livre.items():

```

```
print(f"{cle.capitalize()}: {valeur}")
```

5. Vérifier l'Existence

Utilisez le mot-clé `in` pour vérifier si une **clé** existe dans le dictionnaire :

Python

```
if "auteur" in livre:
    print("L'auteur est renseigné.")
# Sortie: L'auteur est renseigné.

if 1943 in livre.values():
    print("Cette année fait partie des valeurs.")
```



Résumé : Pourquoi utiliser des Dictionnaires ?

Les dictionnaires sont essentiels lorsque vous avez besoin d'associer des informations de manière logique.

- **Exemple** : Stocker des données sur un utilisateur ("nom": "Dupont", "âge": 30).
- **Avantage** : L'accès aux données est très rapide, quelle que soit la taille du dictionnaire, car Python utilise une technique appelée **hachage** pour trouver directement l'emplacement de la clé.



Exercice Pratique : Gestion de Bibliothèque

L'objectif est de créer une classe pour représenter un livre, puis d'utiliser un dictionnaire pour gérer une collection de ces livres (votre bibliothèque).

Partie 1 : La Classe **Livre** (POO)

Créez une classe nommée `Livre` qui représente un seul ouvrage.

1. **Constructeur (`__init__`)** : Doit accepter trois arguments lors de l'instanciation :
 - `titre` (chaîne de caractères)
 - `auteur` (chaîne de caractères)
 - `isbn` (chaîne de caractères unique)
2. **Attributs** : Stockez ces trois arguments comme attributs d'instance. Ajoutez un attribut `disponible` initialisé à `True`.
3. **Méthode `afficher_details()`** : Une méthode qui affiche de manière lisible le titre, l'auteur et l'ISBN du livre.
4. **Méthode `emprunter()`** : Une méthode qui change l'attribut `disponible` à `False` si le livre est actuellement `True`. Si non, affichez un message indiquant qu'il est déjà emprunté.
5. **Méthode `retourner()`** : Une méthode qui change l'attribut `disponible` à `True` si le livre est actuellement `False`.

Partie 2 : Le Dictionnaire **Bibliothèque**

Maintenant, créez un dictionnaire pour organiser votre collection de livres.

1. **Création du Dictionnaire** : Initialisez un dictionnaire vide nommé `bibliothèque`.
2. **Ajout d'Objets** :
 - Instanciez au moins deux objets de la classe `Livre`.
 - Ajoutez ces objets au dictionnaire `bibliothèque`. La **clé** du dictionnaire doit être l'**ISBN** du livre (la valeur unique), et la **valeur** doit être l'**objet `Livre`** lui-même.
3. **Vérification et Manipulation** :

- Utilisez l'ISBN d'un des livres comme clé pour **recupérer** l'objet dans le dictionnaire.
 - Appelez la méthode `emprunter()` sur cet objet récupéré.
 - Vérifiez l'état de l'objet dans le dictionnaire en appelant sa méthode `afficher_details()` et en affichant la valeur de son attribut `disponible`.
4. **Parcourir** : Parcourez le dictionnaire et affichez le titre et l'état de disponibilité (`disponible`) de **tous** les livres.

Aide Mémoire

- Utilisez la syntaxe `mon_dictionnaire[clé] = valeur` pour ajouter un élément.
- Utilisez la syntaxe `for clé, objet in mon_dictionnaire.items():` pour parcourir.