Les Dictionnaires en Python

💡 Définition et Création

Un **dictionnaire** est une collection de données **non ordonnée** et **modifiable**. Contrairement aux listes qui utilisent des index numériques, un dictionnaire stocke des données sous forme de paires **clé-valeur**. Chaque clé est unique et sert d'identifiant pour accéder à sa valeur. Un dictionnaire est reconnaissable par les accolades {} qui l'entourent.

```
root@python:~$
```

```
# Création d'un dictionnaire vide
mon_dictionnaire = {}
# Création d'un dictionnaire avec des éléments
personne = {"nom": "Alice", "age": 25, "ville": "Paris"}
# Obtenir le nombre d'éléments avec len()
nombre_elements = len(personne)
print(f"Le dictionnaire 'personne' contient {nombre_element
# Affiche "Le dictionnaire 'personne' contient 3 éléments."
```

⚙️ Modification du dictionnaire

Ajouter et modifier des éléments

Vous pouvez ajouter une nouvelle paire clé-valeur ou modifier la valeur d'une clé existante en utilisant la syntaxe de l'indexation.

```
root@python:~$
```

```
voiture = {"marque": "Ford", "modele": "Mustang"}
# Ajout d'un nouvel élément
voiture["annee"] = 1964
print(voiture) # {'marque': 'Ford', 'modele': 'Mustang', 'a
# Modification d'un élément existant
voiture["modele"] = "Focus"
print(voiture) # {'marque': 'Ford', 'modele': 'Focus', 'ann
```

Supprimer des éléments

pop(key) : supprime l'élément correspondant à la clé et renvoie sa valeur.

del : supprime l'élément avec la clé spécifiée.

```
root@python:~$
```

```
livre = {"titre": "1984", "auteur": "George Orwell", "annee
# Supprimer un élément avec pop()
annee_supprimee = livre.pop("annee")
print(f"Élément supprimé : {annee_supprimee}")
# Élément supprimé : 1949
print(livre) # {'titre': '1984', 'auteur': 'George Orwell'}
# Supprimer un élément avec del
del livre["auteur"]
print(livre) # {'titre': '1984'}
```

🤔 Copier un dictionnaire

⚠️ Si vous attribuez un dictionnaire à une autre variable avec le signe =, vous ne faites pas une copie ! Les deux variables pointent vers le même objet en mémoire. On appelle cela une référence.

Pour faire une vraie copie, vous devez utiliser la méthode **.copy()**.

```
root@python:~$
```

```
dict_a = {"a": 1, "b": 2}
dict_b = dict_a  # C'est une référence
dict_b["c"] = 3
print(dict_a) # Affiche {'a': 1, 'b': 2, 'c': 3} !
print(dict_b) # Affiche {'a': 1, 'b': 2, 'c': 3}
```

```
dict_c = dict_a.copy()  # Vraie copie avec .copy()
dict_c["d"] = 4
print(dict_a) # Affiche {'a': 1, 'b': 2, 'c': 3}
print(dict_c) # Affiche {'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

🧭 Accès aux éléments

Vous accédez à la valeur d'un dictionnaire en utilisant sa clé. ⚠️ Si la clé n'existe pas, une erreur est générée.

```
root@python:~$
```

```
etudiants = {
  "id_1": "Jean",
  "id_2": "Marie",
  "id_3": "Paul"
}
# Accès par la clé
print(etudiants["id_1"])   # Affiche "Jean"
# Utilisation de la méthode .get() avec une valeur par défa
print(etudiants.get("id_2", "Non trouvé"))  # Affiche "Mari
print(etudiants.get("id_4", "Non trouvé"))  # Affiche "Non
# Utilisation de la méthode .get() sans valeur par défaut
print(etudiants.get("id_2"))  # Affiche "Marie"
# Remarque : La méthode .get() ne génère pas d'erreur si la
```

📜 Parcourir un dictionnaire

La boucle for est l'outil principal pour parcourir un dictionnaire.

1. Par les clés :

C'est la méthode la plus simple et la plus courante.

```
root@python:~$
```

```
profil = {"nom": "Alex", "age": 30, "poste": "Designer"}
for cle in profil:
  print(cle)
Affiche :
nom
age
poste
```

2. Par les valeurs :

On utilise la méthode **.values()** pour parcourir uniquement les valeurs.

```
root@python:~$
```

```
profil = {"nom": "Alex", "age": 30, "poste": "Designer"}
for valeur in profil.values():
  print(valeur)
```

```
Affiche :
Alex
30
Designer
```

3. Par les paires clé-valeur (avec .items()) :

La méthode la plus élégante pour avoir à la fois la clé et la valeur.

```
root@python:~$
```

```
profil = {"nom": "Alex", "age": 30, "poste": "Designer"}
for cle, valeur in profil.items():
  print(f"La clé '{cle}' correspond à la valeur '{valeur}'.
```

```
Affiche :
La clé 'nom' correspond à la valeur 'Alex'.
La clé 'age' correspond à la valeur '30'.
La clé 'poste' correspond à la valeur 'Designer'.
```

🕵️ Vérifier la présence d'une clé

L'opérateur **in** est utilisé pour tester si une clé est présente dans un dictionnaire. Il renvoie un booléen : **True** si la clé est trouvée, et **False** sinon.

root@python:~$

```python
eleves = {"prenom": "Léo", "matiere": "Maths", "note": 15}
# Tester si "prenom" est dans le dictionnaire
if "prenom" in eleves:
    print("La clé 'prenom' existe dans le dictionnaire.")
# Affiche "La clé 'prenom' existe dans le dictionnaire."
# Tester si "age" n'est pas dans le dictionnaire
if "age" not in eleves:
    print("La clé 'age' n'est pas présente.")
# Affiche "La clé 'age' n'est pas présente."
```