

# Dictionnaires

## Fiche de synthèse

### ① DÉFINITION

**Définition 1 (Dictionnaire)** Un **dictionnaire** associe des **clés** à des **valeurs**. Il s'écrit entre accolades ; chaque **entrée** s'écrit **clé**: **valeur**, et les entrées sont séparées par des virgules.

```
eleve = {"nom": "Dupont", "prenom": "Marie",
        "age": 16}
vide = {}
```

Un dictionnaire n'a pas d'indices : `eleve[0]` provoque une erreur.

### ② LIRE UNE VALEUR

```
print(eleve["prenom"])      # Marie
print(eleve["classe"])      # KeyError
print(eleve.get("classe"))  # None
print(eleve.get("classe", "?")) # ?
```

On lit une valeur en écrivant sa clé entre crochets. Une clé absente provoque l'erreur `KeyError` ; la méthode `get` renvoie alors `None`, ou la valeur par défaut donnée en second argument.

### ③ AJOUTER, MODIFIER, SUPPRIMER

```
eleve["classe"] = "1NSI" # ajout
eleve["classe"] = "TNSI" # modification
del eleve["classe"]      # suppression
```

Une clé nouvelle provoque un ajout, une clé existante une modification : un dictionnaire ne contient jamais deux fois la même clé.

### ④ LONGUEUR ET APPARTENANCE

`len(eleve)` donne le nombre d'entrées. `in` cherche parmi les **clés** : `"nom" in eleve` vaut `True`, mais `"Dupont" in eleve` vaut `False`. Pour chercher une valeur, on écrit `"Dupont" in eleve.values()`.

### ⑤ PARCOURIR

```
for cle in eleve:          # les clés
    print(f"{cle} : {eleve[cle]}")
for valeur in eleve.values(): # les valeurs
    print(valeur)
for cle, valeur in eleve.items(): # les couples
    print(f"{cle} → {valeur}")
```

Les clés sont parcourues dans l'ordre où les entrées ont été ajoutées. `items` donne des couples (clé, valeur), que le déballage sépare en deux variables.

### ⑥ CLÉS ET VALEURS

- Une **clé** est immuable : entier, chaîne de caractères ou tuple. Une liste ne peut pas être une clé. On repère ainsi les cases d'un plateau de jeu par des tuples : `plateau[(0, 1)] = "cavalier"`.
- Une **valeur** peut être de n'importe quel type, liste ou dictionnaire compris : `fiche["notes"][2]` donne l'élément d'indice 2 de la liste `fiche["notes"]`.

### ⑦ UNE LISTE DE DICTIONNAIRES

```
classe = [{"prenom": "Marie", "moyenne": 14.5},
          {"prenom": "Alice", "moyenne": 16.0}]
for e in classe:
    print(f"{e['prenom']} : {e['moyenne']}")
```

Un ensemble de valeurs repérées par des noms, et non par des positions, s'appelle un **p-uplet nommé** : un dictionnaire le représente.

### ⑧ COMPTER LES APPARITIONS

```
frequencies = {}
for lettre in "abracadabra":
    n = frequencies.get(lettre, 0)
    frequencies[lettre] = n + 1
# {'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1}
```

Pour une lettre pas encore rencontrée, `get` renvoie 0, et le calcul donne 1.

### ⑨ DICTIONNAIRE EN COMPRÉHENSION

`{n: n ** 2 for n in range(6)}` associe à chaque entier de 0 à 5 son carré. `{prenom: note for prenom, note in eleves}` transforme une liste de couples en dictionnaire.

#### À retenir.

- On lit une valeur par sa clé : `d[cle]` ; `d.get(cle, default)` évite l'erreur `KeyError`.
- `d[cle] = valeur` ajoute ou modifie ; `del d[cle]` supprime.
- `in` cherche parmi les clés ; `keys`, `values` et `items` parcourent clés, valeurs et couples.
- Une clé est immuable ; une valeur peut être de tout type.