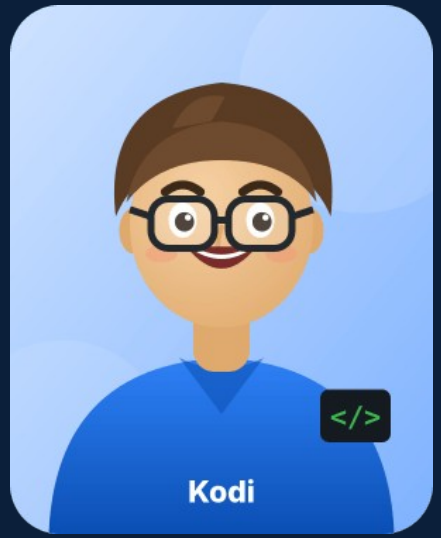
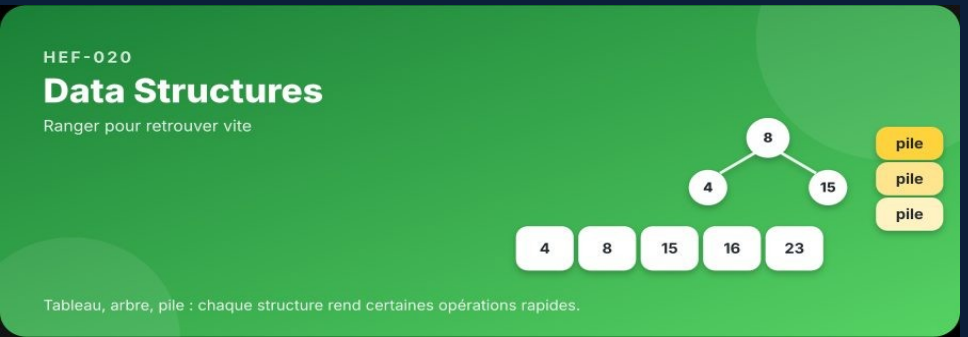
HEF-020 · MISSION 05 / 10

# Ensembles et dictionnaires

*Savoir si quelque chose est présent, associer des valeurs, combiner des groupes : trois besoins, deux structures.*



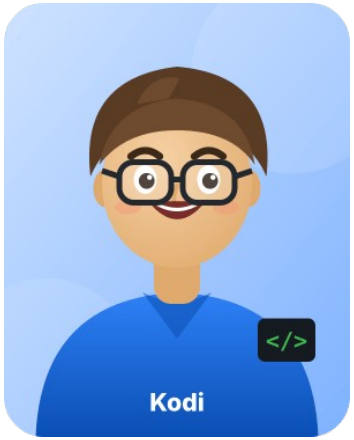

**Badge : Maître des clés**

70 min · débutant

# L'histoire

Maya prépare une campagne : elle veut **les membres inscrits à HEF-000 mais pas encore à HEF-010**, et **ceux qui sont inscrits aux deux**. Kodi commence à écrire deux boucles imbriquées.

*« Tu as un ensemble de membres pour chaque cours, dit Sam. Les mathématiques des ensembles font ça en une ligne. »*




**Tu utilises maintenant les structures à base de hachage pour résoudre des problèmes concrets.**

Ta mission

# À la fin, je serai capable de…

1. utiliser un ensemble pour tester la présence et éliminer les doublons
2. utiliser union, intersection et différence
3. utiliser un dictionnaire pour associer et compter
4. regrouper des éléments dans un dictionnaire de listes
5. choisir entre liste, ensemble et dictionnaire

Défi d'échauffement

# Défi d'échauffement

Tu as deux listes d'invités : celle de ta famille et celle de tes collègues. Certaines personnes sont dans les deux. Comment trouves-tu : (a) tous les invités sans doublons ? (b) ceux qui sont dans les deux listes ? (c) ceux qui sont uniquement collègues ? Décris ta méthode.

Le concept

# L'ensemble (set)

Un **ensemble** est une collection **sans doublons** et **sans ordre**. Tester si un élément est présent est quasi instantané (c'est une table de hachage sans valeurs).

```
a = {1, 2, 3, 4}
b = {3, 4, 5}

print(a | b)    # union : {1, 2, 3, 4, 5}
print(a & b)    # intersection : {3, 4}
print(a - b)    # différence : {1, 2}
print(a ^ b)    # éléments dans l'un ou l'autre, pas les deux : {1, 2, 5}
print(3 in a)   # True
```

Le concept

# Le dictionnaire (dict)

Un **dictionnaire** associe une **clé** à une **valeur**.

```
notes = {"Ada": 15, "Bob": 12}
notes["Cléo"] = 17            # ajouter
print(notes["Ada"])           # lire
print(notes.get("Zoé", 0))    # lire avec valeur par défaut
del notes["Bob"]              # supprimer
for nom, note in notes.items():
    print(nom, note)
```

## Trois motifs très utiles

1. **Compter** : `compteur[x] = compteur.get(x, 0) + 1`
2. **Regrouper** : `groupes.setdefault(cle, []).append(x)`

Le concept

# Trois motifs très utiles (suite)

3. **Indexer** : construire un dictionnaire pour retrouver vite un objet par son identifiant.

## Quelle structure choisir ?

| Besoin | Structure |
|---|---|
| Garder l'ordre, accepter les doublons | liste |
| « Est-ce présent ? », supprimer les doublons | ensemble |
| Associer une valeur à une clé | dictionnaire |
| Opérations de groupes (communs, différences) | ensembles |

⚠ *Dans un ensemble, l'ordre n'est pas garanti. Ne t'appuie jamais dessus.*

# Exemple pas à pas

**Problème de Maya** : membres de HEF-000 pas encore inscrits à HEF-010, et ceux inscrits aux deux.

```
hef000 = {"Ada", "Bob", "Cléo", "Dan"}
hef010 = {"Cléo", "Dan", "Eve"}

print(sorted(hef000 - hef010))   # ['Ada', 'Bob']  (à relancer)
print(sorted(hef000 & hef010))   # ['Cléo', 'Dan'] (déjà dans les deux)
print(sorted(hef000 | hef010))   # tous les membres concernés
```

**Regrouper** les membres par cours :

```
inscriptions = [("Ada", "HEF-000"), ("Bob", "HEF-000"), ("Cléo", "HEF-010")]
par_cours = {}
for membre, cours in inscriptions:
    par_cours.setdefault(cours, []).append(membre)
print(par_cours)   # {'HEF-000': ['Ada', 'Bob'], 'HEF-010': ['Cléo']}
```

# Exemple pas à pas (suite)

**Gain** : aucune boucle imbriquée, et le code se lit comme la question de Maya.

# À toi, avec aide

Voici les langages parlés dans trois équipes :

```
equipe_a = {"français", "anglais", "espagnol"}
equipe_b = {"anglais", "arabe"}
equipe_c = {"français", "arabe", "portugais"}
```

Trouve : (1) les langues parlées par au moins une équipe ; (2) celles parlées par **toutes** les équipes ; (3) celles parlées **uniquement** par l'équipe A.

À toi, avec aide · réponse

# Indices progressifs

## Indice 1

(1) est une union de trois ensembles.

## Indice 2

(2) est une intersection des trois.

## Indice 3

(3) est A moins (B union C).

À toi, avec aide · réponse

# Réponse

```
equipe_a = {"français", "anglais", "espagnol"}
equipe_b = {"anglais", "arabe"}
equipe_c = {"français", "arabe", "portugais"}

print(sorted(equipe_a | equipe_b | equipe_c))
print(sorted(equipe_a & equipe_b & equipe_c))    # [] : aucune
print(sorted(equipe_a - (equipe_b | equipe_c)))  # ['espagnol']
```

# À toi, en solo

Écris `anagrammes(mots)` qui regroupe les mots qui sont des anagrammes les uns des autres. Exemple : `["chien", "niche", "chat", "tach", "chine"]` → groupes `{chien, niche, chine}` et `{chat, tach}`.
Idée : deux mots sont des anagrammes s'ils ont les mêmes lettres une fois triées. Utilise cette « signature » comme clé de dictionnaire.

À toi, en solo · réponse

# Pistes de correction

```
def anagrammes(mots):
    groupes = {}
    for mot in mots:
        signature = "".join(sorted(mot))
        groupes.setdefault(signature, []).append(mot)
    return [g for g in groupes.values() if len(g) > 1]

print(anagrammes(["chien", "niche", "chat", "tach", "chine", "loup"]))
# [['chien', 'niche', 'chine'], ['chat', 'tach']]
```

# Pièges classiques

| Piège | Pourquoi c'est un problème | Comment l'éviter |
|---|---|---|
| **Compter l'ordre dans un ensemble** | L'ordre n'est pas garanti | Trier avec `sorted()` pour afficher |
| **Modifier un dictionnaire en le parcourant** | Erreur ou comportement imprévu | Parcourir une copie : `list(d)` |
| **Clé absente** | `KeyError` | Utiliser `.get()` ou `setdefault()` |
| **Choisir une liste pour « est présent ? »** | Lent sur grandes données | Utiliser un ensemble |

Mini-boss

# Mini-boss

**Recommandation de cours.** Données : pour chaque membre, l'ensemble des cours terminés.

```
termines = {
  "Ada": {"HEF-000", "HEF-010"},
  "Bob": {"HEF-000"},
  "Cléo": {"HEF-000", "HEF-010", "HEF-020"},
}
```

1. `popularite(termines)` : dictionnaire cours → nombre de membres.
2. `membres_similaires(termines, nom)` : membres qui ont terminé au moins un cours en commun avec `nom`.
3. `recommandations(termines, nom)` : cours terminés par les membres similaires mais pas par `nom`, triés par popularité décroissante.

Mini-boss · réponse

# Pistes de correction

```
def popularite(termines):
    c = {}
    for cours in termines.values():
        for x in cours:
            c[x] = c.get(x, 0) + 1
    return c

def membres_similaires(termines, nom):
    moi = termines[nom]
    return {autre for autre, cours in termines.items() if autre != nom and moi & c
   ours}

def recommandations(termines, nom):
    pop = popularite(termines)
    candidats = set()
    for autre in membres_similaires(termines, nom):
        candidats |= termines[autre]
    candidats -= termines[nom]
    return sorted(candidats, key=lambda c: -pop[c])
```

Mini-boss · réponse

# Pistes de correction (suite)

```
termines = {"Ada": {"HEF-000", "HEF-010"}, "Bob": {"HEF-000"}, "Cléo": {"HEF-000",
    "HEF-010", "HEF-020"}}
print(recommandations(termines, "Bob"))   # ['HEF-010', 'HEF-020']
```

Quiz éclair

# Quiz éclair

**1.** Un ensemble contient :

A. des doublons
B. des couples clé-valeur
C. des éléments sans doublons

**2.** `a & b` donne :

A. les éléments communs
B. tous les éléments
C. les éléments de a seulement

**3.** Pour regrouper des éléments par catégorie, on utilise souvent :

A. une pile

Quiz éclair

# Quiz éclair (suite)

B. un dictionnaire de listes
C. un tri

**4.** 🔁 *Révision mission 04* : Une collision, c'est :

A. une panne
B. un doublon de valeurs
C. deux clés dans la même case

**5.** 🔁 *Révision mission 04* : Une clé de dictionnaire doit être :

A. hachable
B. modifiable
C. une liste

Quiz éclair · réponse

# Réponses

1. **C**.
2. **A**.
3. **B**.
4. **C**.
5. **A**.

Je réfléchis

# Je réfléchis

1. Quel problème de ta vie se résout bien par union, intersection ou différence ?
2. Quelle structure utilises-tu naturellement pour noter « qui a quoi » ?

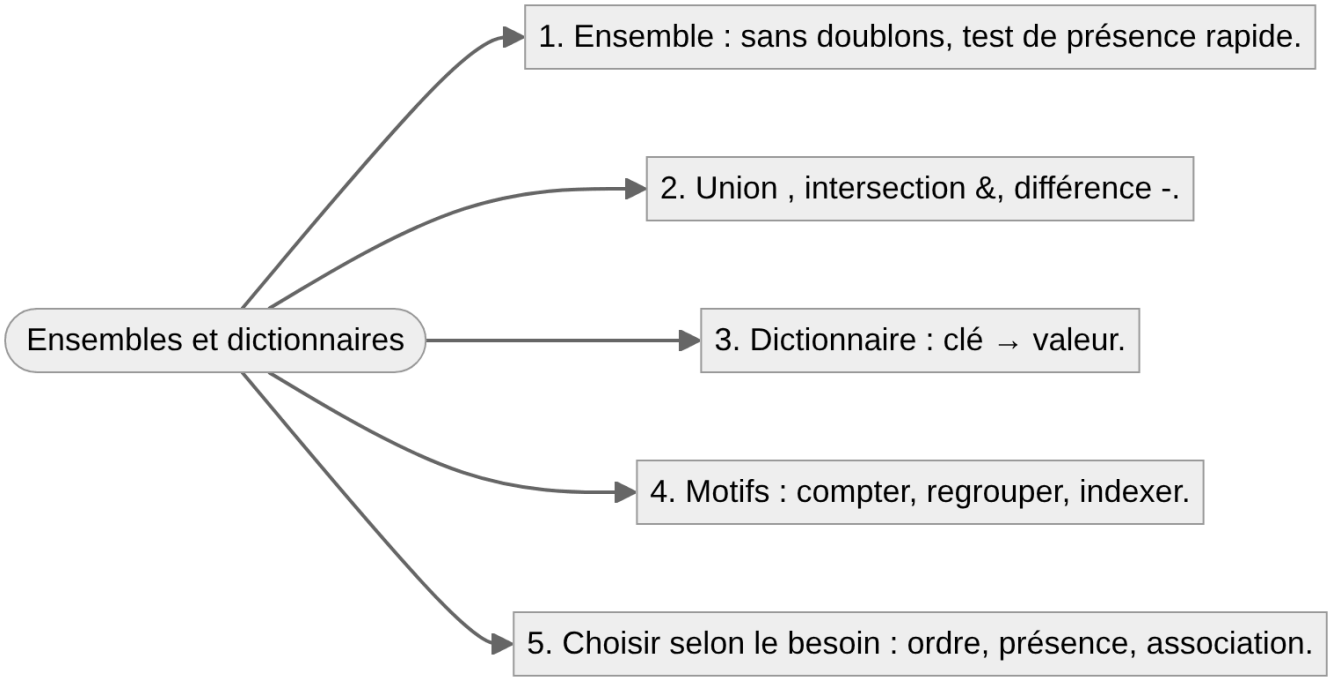
À retenir

# À retenir

1. Ensemble : sans doublons, test de présence rapide.
2. Union `|`, intersection `&`, différence `-`.
3. Dictionnaire : clé → valeur.
4. Motifs : compter, regrouper, indexer.
5. Choisir selon le besoin : ordre, présence, association.

À retenir

# Le schéma de la mission



MISSION ACCOMPLIE

# +100 XP

## Badge débloqué : Maître des clés

Prochaine mission : Arbres et arbres de recherche

