Les dictionnaires en Python

① MÉMO

Structure clé-valeur

Un dictionnaire associe des **clés** (uniques, immuables) à des **valeurs** (quelconques).

```
d = {"nom": "Euler", "prenom": "Leonhard", "annee": 1707}
d = {}   # dictionnaire vide
d = dict()
```

Les clés peuvent être des `str`, `int`, `float`, `tuple`, mais pas des `list` ni des `dict` (types non hachables).

Opérations courantes

`d[cle]` Accès à la valeur associée à `cle` (erreur `KeyError` si absente).
`d.get(cle, defaut)` Renvoie la valeur ou `defaut` si la clé est absente.
`d[cle] = val` Ajoute ou modifie l'association.
`del d[cle]` Supprime l'entrée.
`cle in d` Teste l'existence d'une clé (`True`/`False`).
`len(d)` Nombre d'entrées.

Parcours d'un dictionnaire

```
for cle in d:                  # parcours des clés
for cle in d.keys():           # idem, explicite
for valeur in d.values():      # parcours des valeurs
for cle, valeur in d.items():  # parcours clé + valeur
```

Pièges fréquents

— Confondre `d[cle]` (accès) et `d[cle] = val` (affectation) ;
— Accéder à une clé inexistante sans `get` provoque un `KeyError` ;
— Un dictionnaire n'est pas ordonné par construction (l'ordre d'insertion est conservé depuis Python 3.7, mais ce n'est pas une propriété à exploiter algorithmiquement) ;
— Utiliser une liste comme clé provoque un `TypeError`.

② EXEMPLES

Programme 3 · Compteur d'occurrences

```python
def compter_lettres(texte):
    freq = {}
    for lettre in texte:
        if lettre in freq:
            freq[lettre] += 1
        else:
            freq[lettre] = 1
    return freq

# Version avec get
def compter_lettres_v2(texte):
    freq = {}
    for lettre in texte:
        freq[lettre] = freq.get(lettre, 0) + 1
    return freq
```

Programme 4 · Répertoire téléphonique

```python
repertoire = {}
repertoire["Alice"] = "06 12 34 56 78"
repertoire["Bob"]   = "07 98 76 54 32"

print(repertoire.get("Charlie", "Inconnu"))  # "Inconnu"
```

Programme 5 · Parcours avec items()

```python
notes = {"Alice": 15, "Bob": 12, "Charlie": 18}
for eleve, note in notes.items():
    if note >= 16:
        print(f"{eleve} a obtenu la mention Très bien")
```

Programme 6 · Dictionnaire en compréhension

```python
carres = {x: x**2 for x in range(1, 6)}
# {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Programme 7 · Table de correspondance (p-uplet comme clé)

```python
distances = {
    ("Paris", "Lyon"): 465,
    ("Paris", "Marseille"): 775,
    ("Lyon", "Marseille"): 315
}
print(distances[("Paris", "Lyon")])  # 465
```

③ EXERCICES

Exercice 1 ***Fréquence des mots*** **(3 points)** Écrire une fonction `frequence_mots(phrase)` qui prend une phrase (chaîne de caractères) et renvoie un dictionnaire donnant le nombre d'occurrences de chaque mot.

Exemple : `frequence_mots("le chat et le chien")` renvoie :

`{"le": 2, "chat": 1, "et": 1, "chien": 1}`.

Exercice 2 ***Inverser un dictionnaire*** **(2 points)** Écrire une fonction `inverser(d)` qui échange clés et valeurs d'un dictionnaire. On suppose que les valeurs sont toutes distinctes et immuables.

Exemple : `inverser({"a": 1, "b": 2})` renvoie `{1: "a", 2: "b"}`.

Exercice 3 ***Carnet de notes*** **(3 points)** On représente un carnet de notes par un dictionnaire dont les clés sont les noms des élèves et les valeurs sont des listes de notes.

```
carnet = {"Alice": [15, 12, 18], "Bob": [8, 11, 14], "Charlie": [16, 17]}
```

1. Écrire une fonction `moyenne_eleve(carnet, nom)` qui renvoie la moyenne d'un élève.
2. Écrire une fonction `meilleur_eleve(carnet)` qui renvoie le nom de l'élève ayant la meilleure moyenne.
3. Écrire une fonction `au_dessus(carnet, seuil)` qui renvoie la liste des noms des élèves dont la moyenne est supérieure ou égale au seuil.

Exercice 4 ***Fusion de dictionnaires*** **(2 points)** Écrire une fonction `fusionner(d1, d2)` qui fusionne deux dictionnaires de compteurs en additionnant les valeurs pour les clés communes.

Exemple : `fusionner({"a": 3, "b": 1}, {"b": 2, "c": 4})` renvoie `{"a": 3, "b": 3, "c": 4}`.

SOLUTIONS DES EXERCICES

Corrigé de l'exercice 1.

```python
def frequence_mots(phrase):
    mots = phrase.split()
    freq = {}
    for mot in mots:
        freq[mot] = freq.get(mot, 0) + 1
    return freq
```

Principe : `split()` découpe la phrase en une liste de mots (séparateur par défaut : les espaces). On utilise le schéma du compteur avec `get` pour éviter le `KeyError`.

Corrigé de l'exercice 2.

```python
def inverser(d):
    return {v: k for k, v in d.items()}
```

Attention : si deux clés ont la même valeur, l'une sera écrasée. L'hypothèse de valeurs distinctes est essentielle.

Corrigé de l'exercice 3.

```python
def moyenne_eleve(carnet, nom):
    notes = carnet[nom]
    return sum(notes) / len(notes)

def meilleur_eleve(carnet):
    meilleur = None
    meilleure_moy = -1
    for nom in carnet:
        moy = moyenne_eleve(carnet, nom)
        if moy > meilleure_moy:
            meilleure_moy = moy
            meilleur = nom
    return meilleur

def au_dessus(carnet, seuil):
    return [nom for nom in carnet
            if moyenne_eleve(carnet, nom) >= seuil]
```

Vérification : moyennes : Alice $= 15$, Bob $= 11$, Charlie $= 16{,}5$. Le meilleur est Charlie. Au-dessus de 14 : `["Alice", "Charlie"]`.

Corrigé de l'exercice 4.

```python
def fusionner(d1, d2):
    resultat = dict(d1)   # copie de d1
    for cle, val in d2.items():
        resultat[cle] = resultat.get(cle, 0) + val
    return resultat
```

Principe : on copie `d1`, puis pour chaque entrée de `d2`, on ajoute la valeur à celle existante (ou 0 si la clé est absente).