TP Chapitre 7 : Compréhension

Objectifs du TP

Objectifs du TP. À l'issue de cette séance, tu dois être capable de :

- définir ce qu'est un *itérable* et citer des exemples (liste, chaîne de caractères, objet `range`, dictionnaire) ;
- construire une liste par compréhension : `[expression for variable in itérable]` ;
- construire un dictionnaire par compréhension en parcourant les couples (clé, valeur) avec `items()` ;
- distinguer le `if` *filtre* (placé après le `for`) du `if/else` de *transformation* (placé dans l'expression) ;
- réécrire une boucle classique (liste vide + `append`) en une compréhension ;
- expliquer pourquoi une compréhension ne peut pas contenir de boucle `while`.

Consignes générales. Tous les exercices se traitent en Python (console ou fichier). Enregistre régulièrement ton travail. Quand un programme ne fonctionne pas, lis le message d'erreur : il indique souvent la ligne et le type d'erreur. La partie B se fait *en autonomie* : tu écris des programmes entiers à partir d'une spécification, aucune ligne de code n'est fournie.

Partie A — Application guidée

Les exercices suivants se programment en Python. Ils sont classés du plus simple au plus difficile. Chaque exercice se termine par un *point de validation* (encadré vert) : le résultat exact que doit produire ton programme pour être juste. Complète les squelettes de code en respectant l'indentation.

Exercice A1 — La liste des prix TTC (compréhension simple)

Un magasin affiche ses prix hors taxes et applique la TVA à 20 % : le prix TTC vaut prix HT × 1,2. On dispose de la liste des prix hors taxes de quatre articles.

Écrire une compréhension qui construit la liste des prix TTC à partir de la liste `prix_ht`.

```
prix_ht = [10.0, 20.0, 30.0, 40.0]
# A COMPLETER : liste des prix TTC (chaque prix multiplie par 1.2)

print(prix_ttc)
```

Point de validation

Vérifie ton programme : il doit afficher `[12.0, 24.0, 36.0, 48.0]`.

Exercice A2 — Les soldes d'hiver (`if/else` dans l'expression)

Pendant les soldes, le magasin applique une remise de 20 % sur les articles dont le prix est supérieur ou égal à 50 €, et laisse les autres articles à leur prix initial.

Écrire une compréhension qui construit la liste `prix_reduits` : chaque prix est transformé, selon sa valeur, par la règle de remise.

```
prix = [15.0, 45.0, 60.0, 90.0]
# A COMPLETER : prix reduits : 20% de remise si prix >= 50, sinon prix
    inchange

```

```
print(prix_reduits)
```

Point de validation

Vérifie ton programme : il doit afficher `[15.0, 45.0, 48.0, 72.0]`.

Exercice A3 — Des salaires bruts aux salaires nets (compréhension de dictionnaire)

Le salaire net mensuel vaut environ 78 % du salaire brut (cotisations sociales). On dispose d'un dictionnaire associant à chaque personne son salaire brut mensuel en euros.

Écrire une compréhension de dictionnaire qui construit `nets` : pour chaque couple (nom, brut) de `bruts`, on crée l'entrée `nom : brut × 0,78`. On rappelle que la méthode `items()` fournit les couples (clé, valeur).

```
bruts = {"Lea": 1500.0, "Adam": 2100.0, "Nora": 3200.0}
# A COMPLETER : dictionnaire des salaires nets (brut * 0.78) en parcourant
    bruts.items()

print(nets)
```

Point de validation

Vérifie ton programme : il doit afficher `{'Lea': 1170.0, 'Adam': 1638.0, 'Nora': 2496.0}`.

Exercice A4 — Les comptes à découvert (filtre `if` sur un dictionnaire)

Une banque suit le solde des comptes de ses clients dans un dictionnaire. Un compte est *à découvert* lorsque son solde est strictement négatif.

Écrire une compréhension de dictionnaire qui ne garde que les comptes à découvert : le `if` se place *après* le `for`.

```
comptes = {"client1": 120.0, "client2": -45.0, "client3": 0.0, "client4":
    -8.5}
# A COMPLETER : dictionnaire des comptes a decouvert (solde < 0) seulement

print(decouverts)
```

Point de validation

Vérifie ton programme : il doit afficher `{'client2': -45.0, 'client4': -8.5}`.

Partie B — En autonomie

Tu écris maintenant des programmes *entiers*, sans squelette. Pour chaque exercice : lis la spécification (entrées, traitement, sorties, contraintes), réponds d'abord aux questions de conception de l'encadré orange (sur papier ou à voix haute), puis écris ton programme dans un fichier et teste-le avec les valeurs de l'encadré vert. L'objectif est de concevoir toi-même la solution : ne demande pas de lignes de code toutes faites.

Exercice B1 — La caisse du petit commerce (compréhension et total)

Un petit commerce affiche ses prix hors taxes et applique la TVA à 20 % en caisse. Le gérant veut un programme de caisse minimal : on saisit les prix un par un, le programme les transforme en prix TTC et fait le total.

Énoncé. Écrire un programme qui demande d'abord le nombre d'articles achetés, puis le prix hors taxes de chaque article, et qui affiche la liste des prix TTC puis le total TTC.

Spécification.

— *Entrées* : un entier N (nombre d'articles, $N \geq 1$), puis N prix hors taxes (flottants, en euros).

— *Traitement* : stocker les prix hors taxes dans une liste ; construire par compréhension la liste des prix TTC (chaque prix $\times$ 1,2) ; total TTC = somme des prix TTC.

— *Sorties* : la liste des prix TTC, puis le total TTC (deux lignes).

— *Contraintes* : la liste des prix TTC doit être construite par une compréhension (pas par une boucle `for` avec `append`) ; la fonction `sum` est autorisée pour le total.

Pour t'aider — questions de conception (sans donner le code)

— Pourquoi faut-il stocker les prix dans une liste avant de calculer le total, alors qu'une simple addition au fil de la saisie suffirait pour le total ? (pense à la contrainte « compréhension »)

— Comment transformer chaque prix hors taxes en prix TTC ? Quelle expression écrire dans la compréhension ?

— Pourquoi ne peut-on pas utiliser une boucle `while` dans une compréhension ?

— Que risque-t-il de se passer si $N = 0$? (réponds en une phrase)

Point de validation

Vérifie ton programme : avec 3 articles à 10 €, 20 € et 30 €, il doit afficher `Prix TTC : [12.0, 24.0, 36.0]` puis `Total TTC : 72.0 euros`.

Exercice B2 — Les soldes d'hiver (transformation conditionnelle)

Le magasin prépare ses soldes : une remise de 30 % sur les articles coûtant au moins 50 €, et une remise de 10 % sur les autres. Le gérant veut connaître les prix soldés et l'économie réalisée par le client.

Énoncé. Écrire un programme qui demande le nombre d'articles, puis le prix de chaque article, et qui affiche la liste des prix soldés puis l'économie totale (somme des prix initiaux moins somme des prix soldés).

Spécification.

— *Entrées* : un entier N (nombre d'articles, $N \geq 1$), puis N prix (flottants, en euros).

— *Traitement* : stocker les prix dans une liste ; construire par compréhension la liste des prix soldés : prix $\times$ 0,7 si prix $\geq$ 50, sinon prix $\times$ 0,9 ; économie = somme des prix initiaux − somme des prix soldés.

— *Sorties* : la liste des prix soldés, puis l'économie totale (deux lignes).

— *Contraintes* : le `if/else` doit se trouver *dans l'expression* de la compréhension (transformation conditionnelle, ce n'est pas un filtre) ; la fonction `sum` est autorisée.

Pour t'aider — questions de conception (sans donner le code)

- Où se place le `if/else` dans la compréhension ? Quel est son rôle : filtrer ou transformer ?
- Écris en français la règle de remise appliquée à chaque prix, avec ses deux cas.
- Pourquoi l'économie se calcule-t-elle par `sum(prix) - sum(prix_soldes)` ?
- Teste mentalement avec un article à 50 € exactement : quelle remise s'applique ? Le seuil est-il inclus ou exclu ?

Point de validation

Vérifie ton programme : avec 4 articles à 20 €, 40 €, 60 € et 100 €, il doit afficher `Prix soldes : [18.0, 36.0, 42.0, 70.0]` puis `Economie totale : 54.0 euros`.

Exercice B3 — Le portefeuille d'actions (filtre avec seuil)

Tu suis les prix de quelques actions en bourse (en euros). Tu veux repérer les actions qui dépassent un seuil que tu fixes toi-même.

Énoncé. Écrire un programme qui demande le nombre d'actions, puis le prix de chaque action, puis un seuil, et qui affiche la liste des actions dont le prix est strictement supérieur au seuil, leur nombre, et leur valeur moyenne (si la liste n'est pas vide).

Spécification.

- *Entrées* : un entier N (nombre d'actions, $N \geq 1$), puis N prix (flottants), puis un seuil (flottant).
- *Traitement* : stocker les prix dans une liste ; construire par compréhension la liste des prix strictement supérieurs au seuil (filtre `if` après le `for`) ; valeur moyenne = somme des prix retenus ÷ nombre de prix retenus.
- *Sorties* : la liste filtrée, le nombre d'actions retenues, puis la valeur moyenne ou le message `Aucune action au-dessus du seuil.`
- *Contraintes* : compréhension avec filtre `if` après le `for` ; test de liste vide avant le calcul de la moyenne (sinon division par zéro).

Pour t'aider — questions de conception (sans donner le code)

- Dans cette compréhension, que fait le `if` placé après le `for` : il filtre ou il transforme ?
- Quelle condition écrire pour ne garder qu'un prix strictement supérieur au seuil ?
- Pourquoi faut-il tester la longueur de la liste avant de calculer la moyenne ? (que se passe-t-il avec `sum([]) / len([])` ?)
- Propose un autre seuil qui changerait le résultat de la validation, et prédis ce que le programme afficherait.

Point de validation

Vérifie : avec les prix 85, 120, 55 et 200 et un seuil de 100, il doit afficher `Actions au-dessus du seuil : [120.0, 200.0]`, `Nombre d'actions : 2` et `Valeur moyenne : 160.0 euros`. Avec le même seuil de 300 : liste vide, `Nombre d'actions : 0` et le message.

Exercice B4 — Les comptes clients à découvert (dictionnaire filtré)

Une banque veut surveiller ses comptes clients. Chaque client a un nom et un solde (positif ou négatif). Le logiciel doit construire la liste des comptes, puis isoler ceux qui sont à découvert.

Énoncé. Écrire un programme qui demande le nombre de clients, puis pour chaque client son nom et son solde, et qui affiche le dictionnaire des comptes à découvert (solde strictement négatif) et leur nombre.

Spécification.

— *Entrées* : un entier N (nombre de clients, $N \geq 1$), puis pour chaque client : un nom (chaîne de caractères) et un solde (flottant signé, en euros).

— *Traitement* : construire un dictionnaire `comptes` associant chaque nom à son solde ; construire par compréhension le dictionnaire des comptes à découvert : `{nom: solde for nom, solde in comptes.items() if solde < 0}`.

— *Sorties* : le dictionnaire des comptes à découvert, puis leur nombre (deux lignes).

— *Contraintes* : dictionnaire construit par saisie ; compréhension de dictionnaire avec `items()` ; filtre `if` après le `for`, sans `else`.

Pour t'aider — questions de conception (sans donner le code)

— Quelle méthode fournit les couples (nom, solde) à parcourir ? Pourquoi est-elle indispensable pour une compréhension de dictionnaire ?

— Où se place le filtre dans une compréhension de dictionnaire ? Un `else` est-il autorisé à cette place ?

— Que doit afficher le programme si aucun compte n'est à découvert ?

— Pourquoi le signe du solde est-il la donnée clé pour décider si un compte est à découvert ?

Point de validation

Vérifie : avec 3 clients — Léa (150,0), Adam (−30,5), Nora (0,0) — il doit afficher `Comptes a decouvert : {'Adam': -30.5}` puis `Nombre de comptes a decouvert : 1`.

Partie C — Exercice de réflexion (à finir à la maison)

À finir à la maison

Mon budget mensuel : dépenses, alertes et bilan.

Contexte. Tu gères ton budget du mois : argent de poche ou salaire, loyer, courses, transports, loisirs, abonnements…Tu veux un programme qui transforme et filtre ces montants *par compréhension*, et qui réponde à des questions simples : quelles sont mes dépenses importantes ? Quel est le total de mes dépenses ? Quelle part représentent-elles ? Mon solde est-il positif ?

Objectif. Écrire un programme qui :

— stocke des dépenses — plusieurs modélisations sont possibles : une liste de montants, ou un dictionnaire associant chaque catégorie (loyer, courses…) à son montant ;

— construit par compréhension la liste (ou le dictionnaire) des dépenses supérieures à un seuil d'alerte ;

— calcule le total des dépenses et le solde (recettes − dépenses) ;

— affiche un bilan clair.

Questions à se poser avant de coder.

— Comment représenter une dépense : par un simple montant, ou par un couple (catégorie, montant) ? Qu'est-ce que cela change pour les compréhensions ?

— Quelles sont les données d'entrée ? (recettes, liste des dépenses, seuil d'alerte)

— Quelle compréhension permet de ne garder que les dépenses supérieures au seuil : `if` filtre ou `if/else` ? Pourquoi ?

— Comment calculer le total des dépenses ? (`sum` sur une liste, ou sur les valeurs d'un dictionnaire)

— Comment vérifier que le programme est juste ? (penser à un cas simple : deux dépenses dont on connaît le total à la main)

Étapes suggérées.

1. Version 0, liste de montants : saisir N dépenses, les stocker dans une liste, afficher le total avec `sum` — et vérifier le résultat à la main (ex. 10 et 20 € → total 30 €).
2. Version 1 : construire par compréhension la liste des dépenses importantes (dépense > seuil) et l'afficher.
3. Version 2 : passer au dictionnaire `{catégorie : montant}` et construire par compréhension le dictionnaire des dépenses importantes, avec `items()`.
4. Version 3 : bilan complet — total des dépenses, solde (recettes − dépenses), dépenses importantes et leur part du total en pourcentage.
5. Bonus : tester avec tes vraies dépenses du mois ; que se passe-t-il si le seuil vaut 0 ? Si les dépenses dépassent les recettes ?

Contraintes. Données crédibles (recettes entre 0 et 3000 €, dépenses entre 0 et 1500 €, seuil entre 0 et 200 €) ; programme testé et commenté ; les messages peuvent être écrits sans accents (ex. `Depense`, `total`) si ton éditeur pose problème.

Fin du TP — la partie C est à finir à la maison.