TP Chapitre 6 : Tableaux

Objectifs du TP

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

- représenter un tableau à deux dimensions par une liste de listes : chaque ligne est une liste, les éléments d'une ligne sont rangés dans l'ordre des colonnes ;
- accéder à un élément du tableau avec la notation `M[i][j]` (ligne d'indice i, colonne d'indice j) ;
- afficher un tableau ligne par ligne, ou valeur par valeur avec une double boucle ;
- construire un tableau par compréhension ;
- parcourir un tableau par valeurs ou par indices, et choisir le bon parcours selon la situation ;
- calculer une somme, une moyenne et un maximum sur un tableau ;
- rechercher une valeur ou son indice dans un tableau, avec la valeur sentinelle `-1` ;
- modifier un tableau *en place*, en passant par les indices.

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 — Les formules du traiteur (construire et afficher un tableau)

Un traiteur vend deux formules. Chaque formule a un prix pour l'entrée et un prix pour le plat : la formule 1 coûte 10 € en entrée et 20 € en plat ; la formule 2 coûte 30 € en entrée et 40 € en plat.

Écrire un programme qui représente ces prix dans un tableau à 2 lignes et 2 colonnes (liste de listes), puis qui affiche chaque ligne du tableau l'une après l'autre.

```
# A COMPLETER : construire le tableau des prix (2 lignes, 2 colonnes)

# A COMPLETER : afficher chaque ligne du tableau
```

Point de validation

Vérifie ton programme : l'affichage doit être exactement `[10, 20]` puis `[30, 40]`.

Exercice A2 — Les ventes de la pâtisserie (accès `M[i][j]` et double boucle)

Une pâtisserie vend trois gâteaux. Elle note ses ventes quotidiennes (en euros) pour quatre jours de la semaine : le tableau `ventes` a 3 lignes (les gâteaux) et 4 colonnes (les jours). On veut connaître les ventes du deuxième gâteau le mercredi, puis afficher toutes les valeurs du tableau une par une.

```
ventes = [
    [120, 150, 130, 140],
    [90, 110, 100, 95],
```

```
    [200, 180, 210, 190]
]

# A COMPLETER : afficher les ventes du deuxieme gateau (ligne d'indice 1)
# le mercredi (colonne d'indice 2)

# A COMPLETER : double boucle pour afficher toutes les valeurs une par une
```

Point de validation

Vérifie ton programme : la première ligne affichée doit être `Ventes du 2e gateau le mercredi : 100`, puis les 12 valeurs du tableau doivent s'afficher une par ligne, dans l'ordre : 120, 150, 130, 140, 90, 110, 100, 95, 200, 180, 210, 190.

Exercice A3 — Les recettes du food-truck (somme et moyenne)

Un food-truck note ses recettes quotidiennes (en euros) sur une semaine, du lundi au dimanche, dans la liste `recettes`. Le gérant veut connaître la recette totale de la semaine et la recette moyenne par jour.

```
recettes = [182, 210, 140, 168, 224, 322, 182]

# A COMPLETER : somme des recettes (accumulateur, parcours par valeurs)

moyenne = somme / len(recettes)

print("Recette totale :", somme, "euros")
print("Recette moyenne par jour :", moyenne, "euros")
```

Point de validation

Vérifie ton programme : il doit afficher `Recette totale : 1428 euros` puis `Recette moyenne par jour : 204.0 euros`.

Exercice A4 — Les soldes du magasin (maximum et modification en place)

Pendant les soldes, un magasin de chaussures applique une remise immédiate de 5 € sur chaque article. Avant d'appliquer la remise, le gérant veut connaître le prix le plus élevé du rayon, puis il veut mettre à jour les prix.

```
prix = [25, 48, 12, 39, 60]

# A COMPLETER : prix le plus eleve (parcours par valeurs, initialiser avec
    prix[0])

print("Prix le plus eleve :", maxi, "euros")

# A COMPLETER : remise de 5 euros sur chaque prix (parcours par indices)

print("Prix apres remise :", prix)
```

Point de validation

Vérifie ton programme : il doit afficher `Prix le plus eleve : 60 euros` puis `Prix apres remise : [20, 43, 7, 34, 55]`.

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 — Les recettes mensuelles de la librairie (somme, moyenne, maximum)

Une librairie note ses recettes (en euros) pour chacune des 4 semaines d'un mois. Le gérant veut un bilan simple : le total du mois, la moyenne par semaine et la meilleure semaine.

Énoncé. Écrire un programme qui calcule et affiche la recette totale du mois, la recette moyenne par semaine et la recette maximale (la meilleure semaine), à partir du tableau `recettes = [1420, 1680, 1215, 1930]` écrit en dur dans le programme.

Spécification.

— *Entrées* : aucune saisie ; le tableau `recettes` est écrit en dur.

— *Traitement* : somme par accumulateur ; moyenne = somme ÷ `len(recettes)` ; maximum initialisé avec `recettes[0]`.

— *Sorties* : trois lignes : la recette totale, la recette moyenne par semaine, la meilleure semaine (sa recette).

— *Contraintes* : parcours par valeurs ; boucles uniquement (pas de `sum()` ni de `max()` intégrés) ; tu peux changer les valeurs du tableau pour tester.

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

— Quelle variable joue le rôle d'accumulateur ? Où doit-elle être initialisée, avant ou dans la boucle, et pourquoi ?

— Pourquoi initialiser le maximum avec `recettes[0]` plutôt qu'avec 0 ?

— Pourquoi la moyenne se calcule-t-elle *après* la boucle, et pas pendant ?

— Comment vérifier le résultat à la main avec ce tableau de 4 valeurs ?

Point de validation

Vérifie ton programme : il doit afficher `Recette totale du mois : 6245 euros`, `Recette moyenne par semaine : 1561.25 euros` et `Meilleure semaine : 1930 euros`.

Exercice B2 — Retrouver un article dans le catalogue (recherche d'indice et sentinelle)

Un catalogue propose des articles à prix entiers (en euros) : `catalogue = [12, 25, 8, 30, 25, 15]`. Un client demande le prix d'un article ; le vendeur doit indiquer *où* se trouve cet article dans le catalogue, c'est-à-dire son indice.

Énoncé. Écrire un programme qui demande un prix cible (entier saisi au clavier), parcourt le tableau `catalogue` et affiche l'indice de l'article à ce prix ; si le prix n'est pas dans le catalogue, le programme affiche `-1`.

Spécification.

— *Entrées* : un prix cible (entier) saisi au clavier ; le tableau `catalogue` est écrit en dur.

— *Traitement* : parcours par indices ; quand `catalogue[i]` est égal à la cible, mémoriser l'indice ; la variable commence à `-1` (valeur sentinelle).

— *Sorties* : l'indice de l'article au prix demandé, ou `-1` si aucun article n'a ce prix.

— *Contraintes* : parcours par indices obligatoire (on a besoin de la position) ; pas de méthode `.index()` (non vue en cours).

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

— Pourquoi faut-il parcourir par indices et pas par valeurs ici ?

— Pourquoi `-1` est-il une bonne valeur pour signaler « absent » ?

— Le prix 25 apparaît deux fois dans le catalogue (indices 1 et 4). Quel indice le programme affiche-t-il ? Pourquoi ?

— Comment tester le programme avec un prix présent, puis avec un prix absent ?

Point de validation

Vérifie ton programme : avec 25, il doit afficher `4` ; avec 99, il doit afficher `-1` ; avec 12, il doit afficher `0`.

Exercice B3 — La hausse des salaires (modification en place et arrondis)

Une entreprise accorde une augmentation de 5 % à tous ses salariés. Les salaires mensuels (en euros) sont rangés dans le tableau `salaires = [1850, 2100, 1650, 2400, 1980]`.

Énoncé. Écrire un programme qui multiplie chaque salaire par 1,05, arrondit le résultat à deux décimales (fonction `round(valeur, 2)` — la bonne pratique quand on manipule de l'argent avec des flottants), modifie le tableau *en place*, puis affiche le tableau modifié.

Spécification.

— *Entrées* : aucune saisie ; le tableau `salaires` est écrit en dur.

— *Traitement* : pour chaque i, `salaires[i] = round(salaires[i] * 1.05, 2)` — le tableau est modifié en place, on n'en crée pas de nouveau.

— *Sorties* : le tableau modifié (une seule ligne d'affichage).

— *Contraintes* : parcours par indices obligatoire (on modifie le tableau) ; arrondi à deux décimales avec `round`.

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

— Pourquoi le parcours par indices est-il obligatoire ici ? Que se passerait-il avec `for salaire in salaires` ?

— Pourquoi arrondir à deux décimales quand on manipule de l'argent ? (pense à `0.1 + 0.2` qui ne vaut pas exactement `0.3`)

— Que vaut le premier salaire après l'augmentation ? (calcul à la main)

— Que fait exactement `round(1850 * 1.05, 2)` ?

Point de validation

Vérifie ton programme : il doit afficher `[1942.5, 2205.0, 1732.5, 2520.0, 2079.0]`.

Exercice B4 — Le chiffre d'affaires de la chaîne de boutiques (tableaux à deux dimensions)

Une chaîne possède trois boutiques. Pour chacune, on note le chiffre d'affaires hebdomadaire (en euros) sur quatre semaines. Les données forment un tableau à 3 lignes et 4 colonnes : `ca = [[3200, 4100, 3850, 4600], [2800, 2950, 3100, 3350], [5100, 4750, 5200, 4900]]`.

Énoncé. Écrire un programme qui :

1. affiche le tableau ligne par ligne ;
2. calcule et affiche le chiffre d'affaires total de la chaîne (les 12 valeurs) ;
3. calcule et affiche le chiffre d'affaires de chaque boutique (le total de chaque ligne) ;
4. affiche le numéro de la boutique qui réalise le meilleur chiffre d'affaires (1, 2 ou 3).

Spécification.

— *Entrées* : aucune saisie ; le tableau `ca` est écrit en dur.

— *Traitement* : double boucle (externe sur les lignes, interne sur les colonnes) ; pour chaque ligne, un accumulateur `total_ligne` ; le maximum des totaux de ligne est suivi avec son numéro de boutique.

— *Sorties* : le tableau (3 lignes) ; le total de chaque boutique ; le total général ; la meilleure boutique.

— *Contraintes* : double boucle avec `range(len(...))` ; boucles uniquement ; numéros de boutique affichés de 1 à 3 (l'indice i donne la boutique $i + 1$).

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

— Que parcourt la boucle externe ? Que parcourt la boucle interne ?

— Où faut-il remettre `total_ligne` à zéro, et pourquoi ?

— On initialise `meilleur_total` à 0 : est-ce toujours correct ? Pourquoi est-ce acceptable ici alors que le cours initialise le maximum avec `t[0]` ?

— Pourquoi afficher `i + 1` comme numéro de boutique ?

Point de validation

Vérifie ton programme : il doit afficher `Boutique 1 : 15750 euros`, `Boutique 2 : 12200 euros`, `Boutique 3 : 19950 euros`, `Chiffre d'affaires total : 47900 euros` et `Meilleure boutique : 3`.

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

À finir à la maison

Le tableau de bord des dépenses de la famille Martin.

Contexte. La famille Martin note ses dépenses mensuelles (en euros) par catégorie, pendant une année. Le tableau `depenses` a 12 lignes (les mois, de janvier à décembre) et 4 colonnes (les catégories : logement, alimentation, transport, loisirs). Le tableau est déjà construit : chaque ligne vaut `[logement, alimentation, transport, loisirs]`.

```
depenses = [
    [950, 420, 130, 80],
    [950, 400, 120, 60],
    [950, 450, 140, 100],
    [950, 430, 135, 150],
```

```
    [950, 460, 145, 120],
    [950, 440, 160, 180],
    [950, 480, 150, 250],
    [950, 500, 140, 300],
    [950, 450, 145, 110],
    [950, 470, 150, 90],
    [950, 460, 160, 130],
    [950, 520, 140, 280]
]
```

Objectif. Écrire un programme qui :

1. affiche le tableau mois par mois ;
2. calcule et affiche le total annuel des dépenses ;
3. calcule et affiche le total dépensé chaque mois, puis le mois le plus dépensier ;
4. calcule et affiche le total dépensé par catégorie sur l'année, puis la catégorie la plus coûteuse.

Questions à se poser avant de coder.

— Comment est organisé le tableau : que représente une ligne ? une colonne ? combien de lignes ? combien de colonnes ?

— Quel parcours faut-il pour le total annuel (toutes les valeurs) ? Quelle est la double boucle à écrire ?

— Pour le total par mois, quel indice faut-il fixer ? Pour le total par catégorie, quel indice faut-il fixer et quel indice faut-il faire varier ? (attention : les deux boucles ne sont pas dans le même ordre)

— Comment mémoriser le mois le plus dépensier au fil du parcours ? (même idée que le maximum du cours, avec une variable qui garde le numéro du mois)

— Comment vérifier le programme ? (faire le total d'un seul mois à la main et comparer ; vérifier que la somme des totaux des 4 catégories redonne bien le total annuel)

Étapes suggérées.

1. Version 0 : afficher le tableau ligne par ligne, vérifier que les 12 lignes apparaissent.
2. Version 1 : total annuel (double boucle) ; vérifier à la main sur janvier (1580 €).
3. Version 2 : totaux par mois et mois le plus dépensier (boucle externe sur les lignes).
4. Version 3 : totaux par catégorie et catégorie la plus coûteuse (boucle externe sur les colonnes).
5. Bonus : comparer chaque mois à un budget de 2000 € et afficher les mois qui le dépassent ; simuler une réduction de 10 % sur les loisirs en modifiant le tableau en place ; afficher la part de chaque catégorie en pourcentage du total annuel.

Contraintes. Programme commenté et testé ; les données du tableau ne sont pas modifiées dans les versions 0 à 3 (seul le bonus peut les modifier) ; les messages peuvent être écrits sans accents (ex. `janvier`, `recette`) si ton éditeur pose problème ; aucune fonction de liste non vue en cours (pas de `.index()`, pas de tranches).

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