TP Chapitre 3 : Tuples et Listes

Objectifs du TP

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

- distinguer un type immuable (le tuple) d'un type mutable (la liste) ;
- créer des tuples et des listes, accéder à leurs éléments par indice (0, négatifs) et connaître leur longueur avec `len` ;
- utiliser la concaténation, la répétition et le test d'appartenance (`in`) ;
- modifier une liste : affectation par indice, `append`, `insert`, `remove`, `del`, `pop` ;
- parcourir une liste (`for élément in L` ou `for k in range(len(L))`) et calculer des sommes et des moyennes ;
- construire une liste par compréhension, avec ou sans condition ;
- écrire une fonction qui renvoie plusieurs valeurs (un tuple) et la décomposer ;
- lire un tableau de tableaux (une grille) avec la notation `a[i][j]`.

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. Attention cette semaine aux erreurs de type : un tuple ne se modifie pas (l'affectation `t[0] = ...` lève une `TypeError`), une liste se modifie. 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 — Le panier d'achats (créer et lire un tuple)

Au supermarché, les prix des cinq articles de ton panier sont rangés dans un **tuple** : un tuple est une collection *immuable* (on ne peut pas le modifier) et *indexée* (chaque valeur a une position, l'indice commence à 0).

Complète le programme suivant : il doit afficher le premier prix, le dernier prix (avec un indice négatif), le nombre d'articles, puis le résultat du test d'appartenance de la valeur 24,0 € au tuple.

```
prix = (12.5, 3.99, 24.0, 8.5, 45.0)   # prix des 5 articles en euros

# A COMPLETER : afficher le prix du premier article
# A COMPLETER : afficher le prix du dernier article (indice negatif)
# A COMPLETER : afficher le nombre d'articles (fonction len)
# A COMPLETER : afficher si 24.0 appartient au tuple (operateur in)
```

Point de validation

Vérifie ton programme : il doit afficher `12.5`, puis `45.0`, puis `5`, puis `True` (une valeur par ligne).

Exercice A2 — La facture du libraire (fonction qui renvoie un tuple)

Une librairie applique la TVA à 20 % : le prix TTC vaut prix HT ×1,2. On veut une fonction qui calcule la TVA et le prix TTC à partir du prix HT, et qui *renvoie les trois valeurs* sous la forme d'un tuple.

Complète le programme suivant : la fonction `facture` doit renvoyer le tuple `(prix_ht, tva, prix_ttc)`, puis le programme doit afficher le prix TTC (le troisième élément du tuple renvoyé).

```
def facture(prix_ht):
    tva = prix_ht * 0.2
    prix_ttc = prix_ht * 1.2
    # A COMPLETER : renvoyer le tuple (prix_ht, tva, prix_ttc)

resultat = facture(100)
print("Tuple renvoye :", resultat)
# A COMPLETER : afficher le prix TTC, qui est le troisieme element du tuple
```

Point de validation

Vérifie ton programme : il doit afficher `Tuple renvoye : (100, 20.0, 120.0)`, puis `120.0`.

Exercice A3 — Les abonnements (une liste est modifiable)

Ta liste d'abonnements mensuels (vidéo, musique, presse, cloud…) est rangée dans une **liste** : contrairement au tuple, une liste est *mutable*, on peut la modifier après sa création.

Complète le programme suivant, dans cet ordre : ajouter 7,99 € à la fin ; insérer 14,99 € en position 1 ; retirer la première occurrence de 4,99 ; remplacer l'élément d'indice 2 par 15,99 ; retirer le dernier élément avec `pop` et afficher la valeur retirée.

```
abonnements = [9.99, 12.99, 4.99, 19.99]    # prix mensuels en euros

# A COMPLETER : ajouter 7.99 a la fin de la liste
# A COMPLETER : inserer 14.99 en position 1
# A COMPLETER : retirer la premiere occurrence de 4.99
# A COMPLETER : remplacer l'element d'indice 2 par 15.99
# A COMPLETER : retirer le dernier element avec pop et afficher la valeur
    retiree

print("Liste finale :", abonnements)
```

Point de validation

Vérifie ton programme : il doit afficher `7.99` (la valeur retirée), puis `Liste finale : [9.99, 14.99, 15.99, 19.99]`.

Exercice A4 — Les dépenses du mois (parcourir une liste et additionner)

Tes dépenses du mois (courses, transport, sorties…) sont rangées dans une liste. On veut afficher chaque dépense avec son numéro, puis le nombre de dépenses et le total.

Complète le programme suivant : la boucle parcourt les indices de la liste avec `range(len(depenses))` ; à chaque tour, elle affiche une ligne de la forme `Depense 1 : 45.0 euros` (le numéro est l'indice plus un) et ajoute le montant au total.

```
depenses = [45.0, 12.5, 89.9, 23.0, 6.5, 120.0]    # en euros

total = 0
```

```python
for k in range(len(depenses)):
    # A COMPLETER : afficher "Depense 1 : 45.0 euros" (numero = k + 1)
    # A COMPLETER : ajouter le montant au total

print("Nombre de depenses :", len(depenses))
# A COMPLETER : afficher le total des depenses
```

Point de validation

Vérifie ton programme : il doit afficher les six lignes `Depense 1 : 45.0 euros` jusqu'à `Depense 6 : 120.0 euros`, puis `Nombre de depenses : 6`, puis `Total des depenses : 296.9 euros`.

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 courses du marché (construire une liste avec `append`)

Au marché, tu achètes plusieurs articles et tu veux garder la trace de leurs prix pour vérifier ton budget.

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

Spécification.

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

— *Traitement* : construire la liste des prix au fur et à mesure de la saisie ; parcourir la liste pour calculer le total ; moyenne = total $\div N$.

— *Sorties* : la liste des prix, le total, le nombre d'articles et le prix moyen (quatre lignes).

— *Contraintes* : liste construite progressivement avec `append` ; boucle `for` pour le parcours ; pas de fonction `sum`.

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

— Quelle variable contient la liste des prix ? Comment la faire grandir à chaque article saisi ?

— Pourquoi initialiser la liste à `[]` *avant* la boucle de saisie ?

— Comment calculer le total en parcourant la liste ? (quel accumulateur, où l'initialiser ?)

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

Point de validation

Vérifie ton programme : avec 3 articles à 10 €, 20 € et 30 €, il doit afficher `Liste des prix : [10.0, 20.0, 30.0]`, `Total : 60.0 euros`, `Nombre d'articles : 3` et `Prix moyen : 20.0 euros`.

Exercice B2 — Le portefeuille d'actions (recherche d'un extremum)

Tu suis le cours d'une action pendant plusieurs jours et tu veux savoir quand l'acheter ou la vendre : il faut repérer le cours le plus haut et le cours le plus bas.

Énoncé. Écrire un programme qui demande le nombre de jours, puis le cours de clôture de chaque jour, et qui affiche le cours le plus haut et le cours le plus bas, avec le jour où chacun a eu lieu.

Spécification.

— *Entrées* : un entier N (nombre de jours, $N \geq 1$), puis N cours (flottants, en euros).

— *Traitement* : stocker les cours dans une liste ; parcourir par indices ; comparer chaque cours à un maximum et à un minimum, en mémorisant le jour (l'indice +1).

— *Sorties* : le cours le plus haut et son jour ; le cours le plus bas et son jour.

— *Contraintes* : recherche manuelle avec une boucle (ni `min` ni `max`) ; les jours commencent à 1 alors que les indices commencent à 0.

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

— Pourquoi initialiser `maxi` et `mini` avec le *premier* cours plutôt qu'avec 0 ?

— Pourquoi parcourir avec `range(len(cours))` plutôt qu'avec `for valeur in cours` ?

— Comment mémoriser le jour du maximum ? Que se passe-t-il en cas d'égalité ?

— Pourquoi stocker les cours dans une liste alors qu'on pourrait traiter au fil de la saisie ?

Point de validation

Vérifie ton programme : avec 4 jours de cours 12,5 puis 15,0 puis 10,5 puis 14,0, il doit afficher `Cours le plus haut : 15.0 euros, jour 2` et `Cours le plus bas : 10.5 euros, jour 3`.

Exercice B3 — Les soldes (construire une liste par compréhension)

Un magasin de vêtements met ses articles en soldes : remise de 20 %, le prix soldé vaut prix $\times 0{,}8$.

Énoncé. Écrire un programme qui demande le nombre d'articles, puis leur prix hors taxes, et qui construit *par compréhension* la liste des prix soldés, puis *par compréhension avec condition* la liste des articles soldés à moins de 25 €.

Spécification.

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

— *Traitement* : stocker les prix hors taxes dans une liste ; compréhension simple : chaque prix $\times 0{,}8$; compréhension avec condition : ne garder que les prix soldés inférieurs à 25 €.

— *Sorties* : la liste des prix soldés ; puis la liste des prix soldés inférieurs à 25 €.

— *Contraintes* : les deux constructions doivent utiliser des *compréhensions de liste* (pas de boucle classique) ; remise de 20 %.

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

— Rappelle la syntaxe d'une compréhension simple : quelle expression transforme un prix en prix soldé ?

— Où se place la condition dans une compréhension ? Écris en français la condition « prix soldé inférieur à 25 euros ».

— Peut-on réutiliser la première liste pour construire la seconde ? (réponds en une phrase)

— À quelle boucle avec `append` une compréhension correspond-elle ? (donne un exemple simple)

Point de validation

Vérifie ton programme : avec 4 articles à 30 €, 40 €, 20 € et 60 €, il doit afficher `Prix soldes : [24.0, 32.0, 16.0, 48.0]`, puis `Prix soldes inferieurs a 25 euros : [24.0, 16.0]`.

Exercice B4 — Le bureau de change (liste de tuples)

Au bureau de change, un euro vaut plusieurs devises. Les taux de change sont rangés dans une liste de couples `(devise, taux)` : c'est une **liste de tuples**.

Énoncé. Écrire un programme qui demande un montant en euros et affiche, pour chaque devise, le montant converti, arrondi à 2 décimales. On utilisera la liste fournie :

```
devises = [("dollar US", 1.08), ("livre sterling", 0.85),
           ("yen japonais", 162.5), ("franc suisse", 0.94)]
```

Spécification.

— *Entrées* : un montant en euros (flottant ≥ 0) ; la liste des devises est fournie.

— *Traitement* : pour chaque couple `(devise, taux)` de la liste : montant converti = montant × taux, arrondi à 2 décimales.

— *Sorties* : une ligne par devise, de la forme `100.0 euros = 108.0 dollar US`.

— *Contraintes* : liste de tuples ; décomposition `(devise, taux)` dans la boucle ; pas de dictionnaires ; arrondi avec `round(..., 2)`.

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

— Comment écrire la boucle pour décomposer chaque tuple en deux variables ?

— Sans décomposition, comment accéder au taux à l'aide d'un indice ? (écris l'expression)

— Pourquoi un tuple pour chaque couple `(devise, taux)` et pas une liste ?

— Pourquoi arrondir le résultat avant de l'afficher ? (pense aux flottants et à l'argent)

Point de validation

Vérifie ton programme : avec 100 euros, il doit afficher `100.0 euros = 108.0 dollar US`, `100.0 euros = 85.0 livre sterling`, `100.0 euros = 16250.0 yen japonais` et `100.0 euros = 94.0 franc suisse`.

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

À finir à la maison

Le comparateur de prix.

Contexte. Tu dois acheter trois articles (une calculatrice, une clé USB, un casque) et tu compares les prix dans quatre magasins. Les prix sont réunis dans une grille : une ligne par magasin, une colonne par article.

	Calculatrice	Clé USB	Casque
Magasin 1	12,50	3,99	24,00
Magasin 2	11,90	4,50	25,50
Magasin 3	13,00	3,50	23,90
Magasin 4	12,00	4,20	24,50

Tu es libre de choisir d'autres articles ou d'autres magasins, mais garde des données réalistes (prix entre 1 € et 100 €).

Objectif. Écrire un programme qui :

— affiche la grille des prix, magasin par magasin ;

— calcule le total de chaque magasin pour le panier complet (les trois articles) ;

— indique le magasin le moins cher pour ce panier ;

— (prolongement au choix) : le meilleur prix article par article, les prix TTC avec la TVA à 20 %, un affichage plus lisible…

Questions à se poser avant de coder.

— Comment représenter la grille en Python ? Qu'est-ce qu'une liste de listes ? Quel est le rôle du premier indice, du deuxième ?

— Comment parcourir toutes les cases de la grille ? (deux boucles imbriquées)

— Comment calculer le total d'une ligne ? Où initialiser l'accumulateur ?

— Comment trouver le magasin le moins cher ? (le motif de la recherche du minimum, déjà utilisé dans l'exercice B2)

— Comment vérifier que ton programme est juste ? (calcule à la main le total du magasin 1 : 12,50 + 3,99 + 24,00 = 40,49)

Étapes suggérées.

1. Version 1 : afficher la grille, magasin par magasin (une ligne = une sous-liste).
2. Version 2 : calculer et afficher le total de chaque magasin avec deux boucles imbriquées.
3. Version 3 : mémoriser les totaux dans une liste, puis chercher le plus petit et son numéro de magasin.
4. Version 4 (bonus) : choisir un prolongement parmi ceux proposés ci-dessus et l'ajouter.

Contraintes. Données crédibles (prix entre 1 € et 100 €) ; programme testé et commenté ; les messages peuvent être écrits sans accents (ex. `Total du magasin 1 : 40.49 euros`) si ton éditeur pose problème.

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