Fiche de TP/TD 2

(MSP2)

Informatique 3 : Programmation en Python

---

1. Structures de données

Exercice 1

(Intersection de listes avec doublons)
Soit deux listes d'entiers (avec possiblement des doublons).
Construire une nouvelle liste contenant les éléments qui apparaissent dans les deux listes, en préservant pour chaque élément le nombre d'occurrences minimum.
Exemple : $[1, 2, 2, 3]$ et $[2, 2, 2, 4] \rightarrow [2, 2]$.

Exercice 2

(Fusion de dictionnaires avec priorité)
Deux dictionnaires D_1 et D_2 ont des clés entières et des valeurs entières.
Construire un dictionnaire D_3 tel que pour chaque clé présente dans D_1 ou D_2,
$D_3[\text{clé}] = \max(D_1.\text{get}(\text{clé}, 0),\ D_2.\text{get}(\text{clé}, 0))$.
Ensuite afficher la clé de D_3 dont la valeur est maximale.

Exercice 3

(Liste de tuples – tri personnalisé)
On a une liste de tuples (`clé`, `valeur`), où `clé` est une chaîne et `valeur` un entier.
Trier cette liste d'abord par `valeur` décroissante, puis, en cas d'égalité, par `clé` alphabétiquement.
Afficher le résultat trié.

Exercice 4

(Dictionnaire inverse avec listes)
On dispose d'un dictionnaire D où les valeurs sont des chaînes ou entiers.
Construire un nouveau dictionnaire D_{inv} dont les clés sont les valeurs de D, et dont les valeurs sont **les listes** des clés d'origine ayant cette valeur.
Exemple : $\{'a' : 1, 'b' : 2, 'c' : 1\} \rightarrow \{1 : ['a', 'c'], 2 : ['b']\}$.

Exercice 5

(Supprimer doublons d'une liste tout en conservant l'ordre)
Saisir une liste L d'entiers ou de chaînes.
Construire une nouvelle liste L_2 qui contient les mêmes éléments dans le même ordre **sans doublons** (la première apparition est conservée, puis toute répétition éliminée).
Retourner L_2.

Exercice 6

(Liste de listes – moyenne de sous-listes)
On dispose d'une liste contenant plusieurs sous-listes d'entiers (chaque sous-liste peut avoir une

taille différente).
Pour chaque sous-liste, calculer la moyenne (entière ou flottante) des éléments.
Retourner une liste des moyennes, dans le même ordre que les sous-listes.

Exercice 7

(Dictionnaire de fréquences sauf ponctuation)
Saisir une chaîne de caractères s. Ignorer la ponctuation (., ; :! ?) ainsi que les espaces et la casse (tout passer en minuscule).
Construire un dictionnaire `freq` des fréquences de chaque caractère (lettre uniquement).
Afficher les 5 lettres les plus fréquentes avec leurs fréquences.

Exercice 8

(Extraction de sous-liste selon critère de position et valeur)
Saisir une liste L d'entiers.
Construire une sous-liste S comprenant les éléments de L qui satisfont : «leur valeur est $>$ moyenne(L) **et** leur index est pair» (indexation 0-based).
Afficher S.

Exercice 9

(Concaténation de chaînes dans liste avec filtre)
On dispose d'une liste de chaînes L.
Construire une nouvelle chaîne R égale à la concaténation de toutes les chaînes de L dont **aucune** voyelle (a, e, i, o, u, y) ne se répète dans la chaîne.
Exemple : «hello» non retenue, «world» retenue. Afficher R.

Exercice 10

(Tri par multiple de 3 puis par reste)
Saisir une liste L d'entiers.
Trier L selon le critère : tous les multiples de 3 (ordre croissant), puis tous les autres (ordre croissant).
Afficher la liste triée.

Exercice 11

(Produit scalaire de listes avec dictionnaire de pondération)
On dispose d'une liste de tuples (`clé`, `valeur`) et d'un dictionnaire `Pond` = {`clé` : coefficient}.
Calculer la somme $\sum$(`valeur` $\times$ coefficient) pour chaque tuple dont la clé figure dans `Pond`.
Exemple : $[(\text{`}a\text{'}, 10), (\text{`}b\text{'}, 5), (\text{`}c\text{'}, 2)]$ et `Pond` = $\{\text{`}a\text{'} : 2, \text{`}c\text{'} : 4\} \rightarrow$ calcul $= 10 \times 2 + 2 \times 4 = 28$.

Exercice 12

(Simulation de panier d'achats – listes et dictionnaires)
On dispose de : `stock` = {produit : quantité} et `prix` = {produit : prix_unitaire}.
Faire saisir à l'utilisateur une liste de plusieurs achats sous la forme (produit, qt_demandée) puis recuperer ces achants via la variable`achats` = [(produit, qt_demandée), . . .].
Pour chaque achat : si `stock`[produit] $\geq$ qt_demandée alors : déduire du stock, calculer coût = qt_demandée $\times$ prix[produit]. Sinon : afficher « stock insuffisant pour produit X » et ignorer cet

achat.
À la fin : afficher le coût total et le stock restant.

Exercice 13

(Recherche des « k plus grands uniques » dans liste)
Saisir une liste L d'entiers (peut contenir des doublons) et un entier $k > 0$.
Identifier les k plus grands éléments **uniques** de L. Retourner une liste triée décroissante de ces k éléments.
Exemple : $L = [4, 2, 5, 5, 1, 6]$, $k = 3 \rightarrow [6, 5, 4]$.

Exercice 14

(Dictionnaire imbriqué – résumé de personnes)
Saisir un dictionnaire D de personnes où chaque valeur est un dictionnaire $\{\text{nom} : \ldots, \text{notes} : [\ldots]\}$.
Calculer pour chaque personne la moyenne de ses notes. Construire un nouveau dictionnaire `Moyennes` $= \{\text{nom} : \text{moyenne}\}$.
Afficher les noms triés par moyenne décroissante avec leurs moyennes.

Exercice 15

(Liste circulaire – décalage et rotation)
Saisir une liste L d'entiers et un entier $r \geq 0$.
Réaliser la *rotation circulaire à droite* de L de r positions (les r derniers éléments passent en tête dans le même ordre).
Exemple : $L = [1, 2, 3, 4, 5]$, $r = 2 \rightarrow [4, 5, 1, 2, 3]$. Afficher le résultat.

Creation de Fonctions

Exercice 16

(Factorielle récursive ou itérative)
Écrire une fonction `factorielle(n)` qui renvoie $n!$.
Utiliser une boucle `while` ou la récursion selon votre préférence.
Vérifier le comportement pour $n = 0$ et $n < 0$.

Exercice 17

(Maximum d'une liste variable d'arguments)
Écrire une fonction `maximum(*args)` qui accepte un nombre variable d'arguments et renvoie la plus grande valeur.
Tester avec des entiers et des flottants.

Exercice 18

(Moyenne pondérée avec valeur par défaut)
Écrire une fonction `moyenne_ponderee(*notes, coeff=1.0)` qui renvoie la moyenne des notes multipliée par le coefficient.
Tester avec et sans argument `coeff`.

Exercice 19

(Compter les voyelles d'une chaîne)
Écrire une fonction `compter_voyelles(chaine)` qui renvoie le nombre de voyelles (a, e, i, o, u, y) dans la chaîne passée en argument.
Ne pas tenir compte de la casse.

Exercice 20

(Affichage d'informations variées avec **kwargs)
Écrire une fonction `afficher_infos(**kwargs)` qui affiche chaque clé et sa valeur sur une ligne.
Exemple : `afficher_infos(nom="Alice", age=22, ville="Yaoundé")`.

Exercice 21

(Tri personnalisé)
Écrire une fonction `tri_personnalise(liste, *, reverse=False)` qui trie une liste d'entiers ou de chaînes.
Le paramètre `reverse` permet d'inverser l'ordre du tri.

Exercice 22

(Composition de fonctions)
Écrire une fonction `compose(f, g)` qui renvoie une nouvelle fonction $h(x) = f(g(x))$.
Tester avec des lambdas simples : $f(x) = x + 1$ et $g(x) = 2x$.

Exercice 23

(Générateur de fonctions — closures)
Écrire une fonction `generateur_fonctions(n)` qui renvoie une liste de n fonctions.
La i-ème fonction renvoie $x + i$.
Tester les effets de la fermeture sur les variables.

Exercice 24

(Filtrage d'une liste selon une fonction)
Écrire une fonction `filtrer_par_fonction(liste, f)` qui renvoie la liste des éléments pour lesquels `f(element)` est vraie.
Tester avec une fonction qui sélectionne les nombres pairs.

Exercice 25

(Utilisation de `lambda` avec `map`)
Écrire un lambda `carre = lambda x: x**2`.
Utiliser-le dans `map` pour calculer les carrés d'une liste d'entiers et afficher la liste résultante.

Exercice 26

(Somme croissante avec paramètres optionnels)
Écrire une fonction `somme_croissante(*args, start=0)` qui additionne tous les arguments à partir

d'une valeur initiale `start`.
La fonction renvoie à la fois la somme et le nombre d'éléments additionnés.

Exercice 27

(Division sécurisée avec tuple de retour)
Écrire une fonction `diviser(a, b=1)` qui renvoie un tuple `(quotient, reste)` correspondant à la division entière.
Gérer le cas $b = 0$ en affichant un message d'erreur.

Exercice 28

(Information formatée et dictionnaire de retour)
Écrire une fonction `info_personne(nom, age, ville="Inconnue")` qui affiche :
`"nom (age ans) - Ville : ville"` et renvoie un dictionnaire contenant ces informations.

Exercice 29

(Application successive de plusieurs fonctions)
Écrire une fonction `appliquer_fonctions(x, *funcs)` qui applique successivement les fonctions données à l'argument `x` et renvoie la liste des résultats intermédiaires.
Tester avec plusieurs lambdas.

Exercice 30

(Conversion de températures généralisée)
Écrire une fonction `convert_temperatures(*temps, format="C2F")` qui convertit chaque température selon le format spécifié :
— `"C2F"` : Celsius → Fahrenheit,
— `"F2C"` : Fahrenheit → Celsius.
Renvoie une liste de résultats.

Exercice 31

(Test de primalité et liste des premiers)
Écrire une fonction `est_premier(n)` qui renvoie `True` si n est premier.
Puis une fonction `liste_premiers(n)` qui renvoie la liste des nombres premiers inférieurs ou égaux à n.
Optimiser avec une boucle bornée à $\sqrt{n}$.

Exercice 32

(Comptage multiple dans une chaîne)
Écrire une fonction `count_occurrences(chaine, *motifs)` qui renvoie un dictionnaire associant chaque motif à son nombre d'occurrences dans la chaîne (y compris les chevauchements).

Exercice 33

(Fusion avancée de dictionnaires)
Écrire une fonction `merge_dicts(*dicts, strategy="sum")` qui fusionne plusieurs dictionnaires :

— Si une clé est présente plusieurs fois, additionner les valeurs (`strategy="sum"`).
— Sinon, prendre la valeur maximale (`strategy="max"`).

Exercice 34

(Fabrique de lambdas — lambda factory)
Écrire une fonction `lambda_factory(n)` qui renvoie une liste de n fonctions lambda, où la i-ème renvoie $x + i$.
Tester la différence entre définition dans une boucle et fermeture correcte.

Exercice 35

(Affichage automatique de documentation)
Écrire une fonction `doc_print(f)` qui affiche la docstring de la fonction `f`, puis exécute un test d'appel avec des arguments d'exemple.
Tester avec `help()` et `carre.__doc__`.

Exercice 36

(Currying / Uncurrying génériques)
Écrire `curry(f)` qui transforme une fonction f à n paramètres positionnels en une chaîne de n fonctions unaire (appel curried), et `uncurry(g, n)` qui fait l'inverse.
Contraintes : (i) préservation de l'ordre des paramètres ; (ii) support des valeurs par défaut ; (iii) tolérance aux appels partiels (`curry(f)(x)(y)`).

Exemples :
`f = lambda a,b,c: a + 2*b + 3*c`
`g = curry(f)`
`g(1)(2)(3)` $\rightarrow$ `14`
`uncurry(g,3)(1,2,3)` $\rightarrow$ `14`

Exercice 37

(Fonctions numériques d'ordre supérieur : dérivée et intégration précises)
Écrire `derivee(f, h=1e-6, scheme="central")` qui renvoie une fonction $x \mapsto f'(x)$ via différences finies, et `integrer(f, a, b, n, regle)` pour les approximations numériques.

Exemples :
`f = lambda x: x**3`
`df = derivee(f)`
`df(2)` $\approx$ `12.0`
`integrer(f, 0, 2, 100, "trapèzes")` $\approx$ `4.0`
(Vérifie numériquement que $\int f'(x)dx = f(b) - f(a)$.)