UE121 - Programmation : Langage

HAUTE ÉCOLE DE NAMUR-LIÈGE-LUXEMBOURG

Technologie de l'informatique - bloc 1

Sécurité des systèmes - bloc 1

Exercices 4 : Séquences

Objectifs

- Continuer à manipuler les notions de base du langage : variables, littéraux, affectation, opérations arithmétiques, entrées/sorties...
- Apprendre à manipuler les séquences : listes, chaînes de caractères et n-uplets.
- Apprendre à manipuler les structures de contrôles sur les séquences : alternatives et répétitives.
- veiller au Clean Code,
- veiller à la portabilité du programme

A. Introduction

Les séries d'exercices visent à mettre en pratique les notions vues lors des ateliers et des séances de mise en commun. Il est important de faire un maximum d'exercices pour vous familiariser avec l'IDLE et avec le langage Python.

Au cas où vous ne termineriez pas les exercices durant la séance, nous vous conseillons de les achever chez vous au plus tôt et surtout avant l'atelier suivant.

Cette quatrième série concerne la manipulation des séquences; listes, strings et tuples.

Note.

Dans les exemples qui suivent, les passages en texte normal sont à sortir tels quels.

Les parties en *italique souligné* correspondent aux entrées de l'utilisateur.

Les portions en **gras** varient en fonction des entrées.

Sur votre partition de travail (`U:`), créez un répertoire appelé Python. Ensuite, pour chaque exercice ci-après, ajoutez-y un fichier intitulé `ex4-XX.py` avec `XX` à remplacer par le numéro de l'exercice en question.

B. Exercices à réaliser dans l'IDLE

Écrivez les fonctions/procédures permettant d'effectuer les actions suivantes. Un en-tête avec *type hinting* vous est proposé, tentez d'écrire l'algorithme correspondant :

Exercice 1

Renvoyer si une liste est vide ou non.

def is_empty(lst: list) -> bool

Exercice 2

Afficher le maximum d'une liste ou un message d'erreur si elle est vide. Servez-vous de la fonction de l'exercice précédent.

def print_max(lst: list) -> None

Exercice 3

Remplacer tous les éléments non-nuls d'une liste de nombres par 1.

def replace_non_zero(lst: list) -> None

Exercice 4

Supprimer tous les nombres négatifs d'une liste de nombres. Prenez garde de ne pas chambouler les éléments que vous n'avez pas encore parcourus…

def remove_negatives(lst: list) -> None

Exercice 5

Renvoie un tuple contenant deux listes, l'une avec les éléments d'indice pair, l'autre d'indice impair.

def split_even_odd(lst: list) -> tuple

Exercice 6

Inverser l'élément minimum d'une liste avec l'élément de la première cellule.

def swap_min_first(lst: list) -> None

Exercice 7

Afficher tous les éléments d'une liste, puis tous les éléments sauf le premier, puis tous les éléments sauf les deux premiers et ainsi de suite. N'utilisez pas le slicing.

def print_tails(lst: list) -> None

Exemple :

```
>>> print_tails([3, 7, 9])
3
7
9
7
9
9
```

Exercice 8

Trier tous les éléments d'une liste en combinant les deux exercices précédents : inverser le minimum avec le premier élément, puis inverser le minimum du reste de la liste avec le premier élément du reste de la liste et ainsi de suite. (Indice: cet algorithme s'appelle le tri par sélection)

def selection_sort(lst: list) -> None

Exercice 9

Renvoyer une copie d'une chaîne de caractères où tous les caractères sont mis en minuscule.

def str_to_low(s: str) -> str

Exercice 10

Renvoyer une chaîne de caractères dans l'ordre inverse.

def reverse_str(s: str) -> str

Exercice 11

Renvoyer combien de fois des caractères d'une chaîne de caractères apparaissent dans une chaîne de caractère.

def count_chars(chars: str, s: str) -> int

Exemple :

```
>>> count_chars("ac", "abbcccddd")
4
# car 1 * "a" + 3 * "c"
```

Exercice 12

Renvoyer si une chaîne de caractères est un palindrome.

def is_palindrome(s: str) -> bool

Exercice 13

Afficher un tuple de coordonnées correspondant à l'antipode d'un tuple de coordonnées géographiques ((latitude et longitudes décimales, en degrés, Sud et Ouest négatifs). Afficher une erreur si les coordonnées ne sont pas valides.

def print_antipode(coord: tuple) -> None

Exercice 14

Renvoyer le nombre de kilomètres séparant deux coordonnées géographiques, des tuples, en considérant que la Terre est une sphère de rayon R = 6378km.

def distance_on_earth(coord1: tuple, coord2: tuple) -> float

La formule est rayon * arccos(sin(lat1)sin(lat2) + cos(lat1)cos(lat2)cos(|lon1-lon2|)) avec les latitudes et longitudes en radians. Utilisez le module math.

Exercice 15

Ajouter tous les tuples d'une liste de tuples à une autre liste de tuples, sauf ceux qui y sont déjà présents. Afficher le nombre de tuples ajoutés.

def merge_uniques(lst_from: list, lst_to: list) -> None