
ITC MPSI - TP 4 :

N-UPLETS, LISTES PAR COMPRÉHENSION

Exercice 1 (Produit scalaire en dimension 2)

Codez une fonction `produit_scalaire(x, y)` qui prend en entrée deux paires, `x` et `y`, représentant chacune un vecteur de $\mathbb{R}^2$, et qui renvoie le produit scalaire de ces vecteurs.

Votre code utilisera le dépaquetage des n-uplets en entrée.

On rappelle la définition du produit scalaire de deux vecteurs (x_0, x_1) et (y_0, y_1)

$$x_0y_0 + x_1y_1$$

Exercice 2 (Norme en dimension arbitraire)

Codez une fonction `norme(v)` qui prend en entrée un n-uplet `v` représentant un vecteur de $\mathbb{R}^n$, et qui renvoie la norme du vecteur.

Pour rappel, la norme d'un vecteur $(x_0, \dots, x_{n-1})$ est définie par

$$\sqrt{\sum_{i=0}^{n-1} x_i^2}$$

Exercice 3 (Liste par compréhension)

Écrivez une fonction `longueurs` qui prend en entrée une liste de chaînes de caractères, et qui renvoie en sortie la liste des longueurs de ces chaînes.

Par exemple, l'appel `longueurs(["Bonjour", "à", "vous"])` devra renvoyer la liste `[7, 1, 4]`.

Votre code utilisera une liste définie par compréhension

Exercice 4 (Échauffement sur les listes)

1. Écrire une fonction `pairs(n)` qui prend en entrée un entier `n` et renvoie une liste contenant tous les entiers naturels pairs plus petits que `n`.
2. Écrire une fonction `modifier_sans_erreur(liste, indice, valeur)` qui prend en entrée
 - Une liste `liste`
 - Un entier relatif `indice`
 - Une valeur `valeur`et qui
 - si `indice` est un indice valable pour `liste`, modifie `liste` pour qu'elle contienne `valeur` à l'indice `indice`
 - sinon, ne fait rien (en particulier, ne lève pas d'erreur)

Exercice 5 (Listes palindromes)

Un palindrome est une suite de lettres qui se lisent indifféremment de gauche à droite ou de droite à gauche, par exemple : “radar”, “été”, “rotor”, “ressasser”, “Laval”, “engage le jeu que je le gagne” (Alain Damasio). On peut étendre la notion aux entiers, par exemple 1246421 est palindromique. Dans cet exercice on étend ce concept aux listes : on dira qu’une liste est palindromique si la suite de ses éléments est la même qu’on les regarde de gauche à droite (par indices croissants) ou de droite à gauche (par indices décroissants).

1. Créer une fonction `retourne` ayant comme argument une liste et qui renvoie la liste "retournée".
2. Écrire une fonction `est_palindrome` qui prend une liste en argument et renvoie un booléen qui indique si la liste est palindromique, en utilisant la fonction `retourne` de la question 1.
3. Pensez-vous à une façon de coder la question de l’exercice 2 sans créer une version retournée de la liste ?

Exercice 6 (Bonus : Autres palindromes)

1. Écrire une fonction `palindrome_avec_espaces(chaine)` qui prend en entrée un chaîne de caractères et renvoie un booléen qui indique si c’est un palindrome. On n’ignorera pas les espaces.
2. Écrire une fonction `palindrome_sans_espaces(chaine)` qui prend en entrée un chaîne de caractères et renvoie un booléen qui indique si c’est un palindrome. On ignorera les espaces.
3. Écrire une fonction `entier_palindromique(n)` qui prend en entrée un entier et renvoie un booléen qui indique si l’écriture en base 10 de l’entier en entrée est un palindrome.