TP 8 informatique

BCPST 1 2019-2020

Les listes

1 Introduction

Lorsque l'on veut manipuler une quantité importante de données, on constate rapidement qu'il faut pouvoir stocker ces données. Or les types d'objets que l'on a vu jusqu'à maintenant ne permettent pas vraiment de faire cela : par exemple, si j'ai besoin des n premiers termes d'une suite, il nous est pour l'instant difficile de tous les garder en mémoire, tâche qui sera réalisable à l'aide d'une liste.

2 Exercice guidé : syntaxe des listes

Tout est à saisir dans l'interpréteur.

Exercice 1.

1. Saisir `L=[1,2,3,4]`. L'objet `L` est une liste d'entiers.
2. On veut modifier le premier élément. Saisir `L[0]=34`. Quelle est la nouvelle liste ?
3. On veut rajouter un élément à la fin de la liste. Saisir `L.append(8)`. Quelle est la nouvelle liste ?
4. On veut échanger deux éléments de positions. Saisir `L[1],L[4]=L[4],L[1]`. Quelle est la nouvelle liste ?
5. On veut supprimer le deuxième élément de la liste. Saisir `valeur=L.pop(1)`. Quelle est la nouvelle liste L ? Que contient `valeur` ?
6. Saisir M=`[''ratus'','jerry','speedy'']` et `L=L+M`. Quelle est la nouvelle liste `L` ?
7. Saisir `S=L` et `L[2]=48`. Quelle est la nouvelle liste `S` ? Attention, lorsque l'on modifie `L`, `S` est aussi modifiée. Si on veut modifier une copie, on procède différemment.
8. Saisir S=L[:]. Modifier la liste L. Constater que la liste S n'a pas été modifiée.
9. Il est possible d'effectuer des copies partielles d'une liste. Saisir respectivement S=L[1 :5], T=L[:3], U=L[4 :]. Quelles sont les listes que vous obtenez ?

3 Exercices

Exercice 2.

1. Initialiser une liste `L` de longueur 432 où les éléments sont :$f(1), f(2), \cdots, f(432)$ avec f qui est la fonction cube.
2. Supprimer de `L` l'élément en indice 233.
3. Ajouter en fin de liste le cube de 433.

Exercice 3. Écrire une fonction `moyenne(L)` qui retourne la moyenne des éléments de L.

Exercice 4. Écrire une fonction `Suite(n)` qui retourne une liste contenant les n premiers termes de la suite définie par :

1. $u_0 = 1, \forall n \in \mathbb{N}, u_{n+1} = u_n + n$
2. $u_0 = 1, u_1 = 5, \forall n \in \mathbb{N}, u_{n+2} = u_{n+1} + 2u_n$

Exercice 5. Écrire une fonction `estunelement(L,x)` qui retourne `True` si `x` est un élément de `L` et `False` sinon.

Exercice 6. Écrire une fonction `rechercheMax(L)` qui retourne le plus grand élément de `L` et la position de sa première occurrence.

Exercice 7. Écrire une fonction `insertion(L,i,a)` qui insère en indice i l'élément a dans L. On pourra utiliser des copies partielles.

Exercice 8. Écrire une fonction `recherche(L,x)` qui retourne la première position de x dans `L` si x est un élément de `L` et -1 sinon.

Exercice 9. Écrire une fonction `Suite(n)` qui retourne une liste contenant les n premiers termes de la suite définie par :

1. $u_0 = 1, \forall n \in \mathbb{N}, u_{n+1} = \sum_{k=0}^{n} (u_k)^2$
2. $u_0 = 1, \forall n \in \mathbb{N}, u_{n+1} = \sum_{k=0}^{n} u_k u_{n-k}$.

Exercice 10. Écrire une fonction `nbre_base(chaine)` qui prend en argument une chaîne de caractères composées uniquement de A, U, G, C et qui renvoit une liste d'entiers $L = [a, u, g, c]$ où a, u, g, c sont respectivement le nombre de A, U, G, C de la chaîne.

Exercice 11. $\star$

1. Écrire une fonction `est_somme(L,S)` qui prend en argument une liste d'entiers L et un entier S et qui renvoit `True` s'il existe $i \neq j$ vérifiant $L[i] + L[j] = S$.
2. Même question. On suppose cette fois-ci que la liste est triée dans l'ordre croissant. Le nombre d'additions effectuées ne doit pas dépasser le nombre d'éléments de la liste.