I Situation

Il s'agit de construire une suite dont le premier terme est un nombre entier que l'on choisit.

On établit la liste de ses diviseurs propres (le nombre ne figure pas dans la liste). On calcule la somme des diviseurs propres du nombre. Le résultat est le deuxième terme de la suite. Pour le troisième terme on prend la somme des diviseurs propres du terme précédent.
On réitère le procédé.

On peut décider de noter (SA_n) cette suite de sorte que SA_0 =le nombre de départ, SA_1 =la somme des diviseurs propres de SA_0, SA_2 la somme des diviseurs propres de SA_1 et etc.

1. `Construire « à la main » la suite aliquote initialisée à 12.`
2. `Recommencer avec` $SA_0 = 28$.

• ○ • ○ •

II Mise en œuvre algorithmique

II.1 Une liste comme représentation de la suite

Voir : `http://www.mimaths.net/IMG/pdf/algorithmique_sio.pdf`

document dans lequel est abordé l'objet `liste`. Des précisions importantes :

▷ La commande `SA=[]` crée une variable liste et l'initialise à vide ;
▷ La `méthode append` permet d'ajouter un élément à une liste : *syntaxe* → `SA.append(n)` n est l'élément ajouté
▷ Chaque élément de la liste est accessible par son index tout comme une chaîne de caractères.
Si `SA=[2,9,4,1]` alors `SA[2]` est 4.

II.2 Une fonction possible pour stocker les diviseurs propres d'un nombre dans une liste

Vous pouvez écrire la fonction `ListeDiviseurs(nombre)` qui retourne la `liste` des diviseurs propres de `nombre`.

Cette liste contient 1 donc il suffit de tester la divisibilité de `nombre` *par tous les entiers de 2 jusqu'à la « moitié » de* `nombre` on peut faire mieux, mais cela suffira.

Par exemple, `ListeDiviseurs(12)` doit retourner `[1,2,3,4,6]`.

II.3 Une autre fonction qui calcule la somme des éléments d'une liste

Vous pouvez écrire la fonction `SommeListe(liste)` qui retourne le somme des éléments de `liste`.

▷ La commande `len(liste)` renvoie la longueur de `liste`.

II.4 Fabriquer la suite aliquote d'un nombre

Vous pouvez écrire la fonction `SuiteAliquote(nombre)` qui retourne une liste contenant les termes de la suite aliquote dont il est question dans le paragraphe I.

Par exemple, `SuiteAliquote(12)` doit retourner `[12,16,15,9,4,3,1]`.

Quelques remarques : On ne sait pas à priori combien de termes composent une suite aliquote. Elle peut terminer par 1, ne contenir que des nombres identiques, contenir des « périodes », ou même avoir un comportement non identifié à ce jour. Il est conseillé de lire

`https://fr.wikipedia.org/wiki/Suite_aliquote`

Ainsi pour le « test d'arrêt » : je conseille d'arrêter la construction de la suite lorsque l'on trouve 1 ou si on dépasse 100 termes. (par exemple)