

TP Informatique n° 2

Révisions sur les structures itératives

(illustrations sur les suites et les séries)

I. Les structures de contrôle

En informatique, on appelle **structure de contrôle** une commande qui contrôle l'ordre dans lequel les différentes instructions d'un programme sont exécutées.

On peut distinguer plusieurs types de structures de contrôle :

- a) les **structures séquentielles** : les différentes commandes sont exécutées les unes à la suite des autres *i.e.* dans un ordre séquentiel.
- b) les **structures conditionnelles** : qui permettent d'introduire des branchements conditionnels dans le programme. Un programme peut comporter plusieurs branches. Chacune d'elle est associée à une condition. La branche exécutée est la première dont la condition est réalisée.
- c) les **structures itératives** : qui permettent d'effectuer la répétition (*i.e.* l'itération) de commandes. Ces répétitions peuvent s'effectuer :
 - × un nombre de fois fixé explicitement par le programmeur.
On utilise alors une boucle **for**.
 - × ou peuvent avoir lieu tant qu'une condition n'est pas réalisée.
On utilise alors une boucle **while**.

II. La boucle for

II.1. Syntaxe générale

Une boucle **for** permet de réaliser la répétition d'un bloc d'instructions.

En **Python**, la syntaxe classique de cette commande est la suivante.

```
for i in range(n):  
    instruction
```

- Écrire un programme permettant d'afficher un à un les éléments de `range(5)`.

- Quel est le résultat de l'exécution de ce programme ? De manière générale que produit l'instruction `range(n)` ?

- ▶ En procédant de même, décrire ce que produit les instructions `range(3, 9)`, puis `range(3, 9, 2)` et `range(9, 3, -1)`.

- ▶ Une spécificité du langage **Python** est qu'il est possible d'itérer sur de nombreux objets. En utilisant le procédé précédent, écrire une itération sur la chaîne de caractère "ma chaîne" permettant d'afficher un à un les éléments de cet objet. Qu'obtient-on ?

II.2. Calcul du $m^{\text{ème}}$ élément d'une suite

On considère la suite $(u_n)_{n \in \mathbb{N}^*}$ définie par :

$$\begin{cases} \forall n \in \mathbb{N}^*, u_{n+1} = 2u_n + n + 1 \\ u_1 = 1 \end{cases}$$

- ▶ Écrire un programme qui :
 - × demande initialement à l'utilisateur d'entrer au clavier la valeur d'un entier m ,
 - × affiche la valeur du $m^{\text{ème}}$ élément de la suite (u_n) .

- ▶ Calculer u_{12} et u_{20} à l'aide du programme précédent.

- ▶ Modifier le programme précédent afin d'obtenir une fonction `emeSuiteU` qui :
 - × prend en paramètre une variable `m`,
 - × renvoie le $m^{\text{ème}}$ élément de la suite (u_n) .

- ▶ Calculer u_7 et u_{15} à l'aide de la fonction précédente.

- ▶ Selon vous, quels sont les avantages de la représentation sous forme de programme avec dialogue utilisateur ? Sous forme de fonction ?

II.3. Calcul des m premiers éléments d'une suite

- ▶ Écrire un programme qui :
 - × demande initialement à l'utilisateur d'entrer au clavier la valeur d'un entier `m`,
 - × affiche la liste des `m` premiers éléments de la suite (u_n) .

- ▶ Calculer les 5 premiers éléments de la suite à l'aide du programme précédent.

- ▶ Modifier ce programme afin d'obtenir une fonction `premSuiteU` qui :
 - × prend en paramètre une variable `m`,
 - × renvoie la liste des `m` premiers éléments de la suite (u_n) .

II.4. Calcul des sommes partielles d'ordre n

On considère maintenant les suites $(u_n)_{n \in \mathbb{N}}$ et (v_n) suivantes :

$$\forall n \in \mathbb{N}, u_n = \frac{n^2}{3^n} \quad \text{et} \quad \begin{cases} v_0 = 1 \\ \forall n \in \mathbb{N}, v_{n+1} = \frac{2v_n}{e^{v_n} + e^{-v_n}} \end{cases}$$

On notera par la suite $S_n = \sum_{k=0}^n u_k$ et $T_n = \sum_{k=0}^n v_k$ les sommes partielles d'ordre n des séries $\sum u_n$ et $\sum v_n$.

II.4.a) Calcul de S_n

- ▶ Écrire une fonction `calculSn` qui :
 - × prend en paramètre un entier `n`,
 - × renvoie la valeur de S_n .

- ▶ Que vaut S_0 ? S_5 ? S_{10} ? S_{100} ? S_{1000} ?

II.4.b) Calcul de T_n

Il s'agit ici d'illustrer le cas où la suite (u_n) est donnée sous forme récurrente.

- ▶ Écrire une fonction `calculTn` qui :
 - × prend en paramètre un entier n ,
 - × renvoie la valeur de T_n .

- ▶ Que vaut T_0 ? T_{100} ? T_{10000} ?

II.5. Calcul des n premières sommes partielles

II.5.a) Calcul des n premières sommes partielles de $\sum u_n$

- ▶ Soit $n \in \mathbb{N}$. Quel lien y a-t-il entre S_n et S_{n+1} ?

- ▶ En tirant profit de l'égalité précédente, écrire une fonction `premS` qui :
 - × prend en paramètre une variable n ,
 - × renvoie la liste des n premières valeurs de la suite (S_n) .

On ne devra pas effectuer d'appel à `calculSn`.

- ▶ Quelles sont les 5 premières valeurs de la suite (S_n) ?

II.5.b) Calcul des n premières sommes partielles de $\sum v_n$

- Soit $n \in \mathbb{N}$. Quel lien y a-t-il entre T_n et T_{n+1} ?

- En tirant profit de l'égalité précédente, écrire une fonction `premT` qui :

- × prend en paramètre une variable `n`,
- × renvoie la liste des `n` premières valeurs de la suite (S_n) .

On ne devra pas effectuer d'appel à `calculTn`.

III. La boucle while

III.1. Calcul du premier entier tel qu'une condition est vérifiée

Le problème qui nous intéresse ici est le suivant.

Problème.

Données :

- Une suite (u_n) et une suite (d_n) telle que $d_n \xrightarrow{n \rightarrow +\infty} 0$.
- L'existence d'un réel α tel que : $\forall n \in \mathbb{N}, |u_n - \alpha| \leq d_n$.

But :

- 1) Déterminer un indice N tel que le terme u_N vérifie : $|u_N - \alpha| \leq 10^{-4}$.
- 2) En déduire une valeur approchée de α à 10^{-4} près.

III.2. Un exemple classique

On commence par illustrer le problème et sa résolution par l'étude d'une suite de type $u_{n+1} = f(u_n)$ dans le cadre de l'utilisation de l'inégalité des accroissements finis.

On considère la fonction $f : x \mapsto e^{-\frac{x^2}{2}}$ et on définit la suite (u_n) par :

$$\begin{cases} u_0 = \frac{1}{2} \\ \forall n \in \mathbb{N}, u_{n+1} = f(u_n) \end{cases}$$

Rappelons les différentes étapes de ce type d'étude. Les démonstrations sont laissées au lecteur.

- 1) En appliquant le théorème de la bijection à la fonction $g : x \mapsto f(x) - x$, on démontre que l'équation $f(x) = x$ admet une unique solution dans $[0, 1]$, que l'on note α .
- 2) Après avoir démontré que l'intervalle $[0, 1]$ est stable par f , on en déduit, par récurrence que : $\forall n \in \mathbb{N}, u_n \in [0, 1]$.
- 3) a) Par étude de la fonction f' , on démontre : $\forall x \in [0, 1], |f'(x)| \leq \frac{1}{\sqrt{e}}$.
b) On est alors dans le cadre de l'application de l'IAF, qui permet de démontrer que :

$$\forall n \in \mathbb{N}, |u_{n+1} - \alpha| \leq \frac{1}{\sqrt{e}} |u_n - \alpha|$$

(on démontre en fait que $|f(u_n) - f(\alpha)| \leq \frac{1}{\sqrt{e}} |u_n - \alpha|$)

c) On en déduit que : $\forall n \in \mathbb{N}, |u_n - \alpha| \leq \left(\frac{1}{\sqrt{e}}\right)^n$.

d) Comme $\left(\frac{1}{\sqrt{e}}\right)^n \xrightarrow{n \rightarrow +\infty} 0$, on en déduit que (u_n) est convergente, de limite α .

Le but est alors de calculer une valeur approchée de α à 10^{-4} près.

- Quelle condition permet d'assurer que $|u_n - \alpha| \leq 10^{-4}$?

- Écrire un programme permettant d'afficher le premier entier n tel que $\left(\frac{1}{\sqrt{e}}\right)^n \leq 10^{-4}$.

- Compléter le programme précédent afin qu'il affiche une valeur approchée à 10^{-4} près de α .

- Déterminer une formule mathématique donnant le premier entier n tel que $\left(\frac{1}{\sqrt{e}}\right)^n \leq 10^{-4}$.

- Comparer la valeur obtenue dans la question précédente et celle affichée par le programme.

- De même, donner la formule permettant d'obtenir le premier entier n tel que $\left(\frac{1}{\sqrt{e}}\right)^n \leq \varepsilon$.

- En déduire une fonction `calcApproch` qui prend en paramètre un réel `eps` et qui renvoie une valeur approchée de α à `eps` près à l'aide d'une boucle `for`.