IPT : Cours 7

La programmation informatique en 3 étapes

MPSI : Prytanée National Militaire

Pascal Delahaye

16 janvier 2016

1 La démarche de programmation

Contrairement à ce que vous pourriez éventuellement penser, pour construire un programme, un informaticien commence par prendre un papier et un crayon pour analyser le problème qu'il cherche à résoudre...

Dans ce chapitre, nous allons présenter la démarche de programmation en concevant un programme de calcul du terme général d'une suite récurrente $\begin{cases} u_0 \\ u_{n+1} = f(u_n) \end{cases}$.

1.1 Etape 1 : Les éléments de base

Toute conception d'un programme commence par donner des réponses aux questions suivantes :

1. Quel nom vais-je donner à mon programme ?
2. Quels sont les arguments à prendre en compte ? De quel type sont ces arguments ?
3. Que doit me renvoyer mon programme ? Sous quelle forme ?
4. Quel algorithme vais-je choisir pour résoudre mon problème ?

1. Première version possible :

Python
```
Nom :        u
Arguments :  n  -> l'indice du terme à calculer (integer)
Sortie :     la valeur du terme Un de la suite (float)
Algorithme : Boucle "for" qui à chaque itération, calcule le terme Uk
             de la suite
```

Ce programme ne renvoie le terme u_n que d'une seule suite. Une fois ce programme conçu, il est impossible de changer par exemple la valeur initiale u_0 sans modifier certaines instructions.

2. Deuxième version possible :

```
Python
Nom :        u
Arguments :  n  -> l'indice du terme à calculer (integer)
             u0 -> le premier terme de la suite (float)
             f  -> la fonction qui définit ma suite (fonction lambda)
Sortie :     la valeur du terme Un de la suite (float)
Algorithme : Boucle "for" qui à chaque itération, calcule le terme Uk
             de la suite
```

Ce programme présente l'avantage de pouvoir s'appliquer à n'importe quelle suite récurrente d'ordre 1.

1.2 Etape 2 : Ecriture de l'algorithme en pseudo-langage

Avant de taper un programme en langage Python à l'écran de l'ordinateur, on commence par le rédiger en *pseudo langage* sur une feuille de papier. Le pseudo-langage est une sorte d'hybride entre le langage courant et la langage informatique.

A ce titre, il présente les avantages suivants :

1. Il est facilement compréhensible à la lecture
2. Il peut se transcrire facilement dans la plupart des langages informatiques
3. Il permet de mettre l'accent sur le sens de ce que l'on écrit sans se préoccuper des problèmes liées à la syntaxe du langage informatique que l'on utilise.

`Exemple 1.` Reprenons l'exemple précédent...
Il s'agit maintenant de traduire en pseudo-langage l'algorithme de calcul choisi pour le calcul de u_n.
On pourra procéder ainsi :

```
Python
Variables locales :  U -> variable de stockage du terme général
                     k -> indice de boucle

Idée : Pour calculer Un, on réitère la formule U = f(U) n fois.

Initialisation : U = u0
Boucle :         Pour k allant de 1 à n faire : U = f(U)

Sortie de boucle : la variable U contient la valeur Un

Sortie de programme : on renvoie la valeur U
```

1.3 Etape 3 : la traduction en langage de programmation

Si le travail d'analyse en Etape 1 et 2 a été effectué correctement, la traduction en langage de programmation ne pose plus que des problèmes éventuels de syntaxe.

`Exemple 2.` Reprenons l'exemple précédent...
La traduction du pseudo-langage en langage Python donne alors :

```
Python
def U(n):
    U = 2                                  # Par exemple
    for k in range(1,n+1) : U = 1 - U**2   # par exemple
    print(U)

U(10)
```

ou encore, si l'on décide d'ajouter u_0 et f en variables :

Python
```
def U(n, f, u0):
    U = u0                                # Par exemple
    for k in range(1,n+1) : U = f(U)   # par exemple
    print(U)

U(10, lambda x : 1-x**2, 0.3)
```

`Exemple 3.` Mettre en oeuvre la démarche précédente afin de construire un programme renvoyant les n premiers termes d'une suite récurrente.

Aide : on pourra utiliser une structure de liste pour stocker les valeurs.
Pour stocker la valeur "a" dans une liste "L" on dispose de l'instruction `L.append(a)`

2 Exercices

Construire les programmes suivants en prenant soin de bien suivre la méthodologie exposée dans la partie précédente.

Exercice : 1
(∗) Construire un programme qui renvoie le graphe portant en abscisses les entiers de 1 à N et en ordonnées le nombre d'étapes nécessaires pour passer de p à 1 dans l'algorithme de Syracuse.

Exercice : 2
(∗) Construire un programme qui répond VRAI si un élément donné se trouve dans une liste donnée et NON sinon.

Exercice : 3
(∗) Soit f une fonction continue sur un segment $[a,\ b]$ tel que $f(a)f(b) < 0$.
Ecrire une procédure de résolution d'une équation $f(x) = 0$ par dichotomie.

Exercice : 4
(∗∗) Construire un programme qui renvoie le nombre de chiffres d'un entier naturel donné.

Exercice : 5
(∗∗) Construire un programme qui renvoie la somme des chiffres d'un entier naturel donné.

Exercice : 6
(∗) Calcul du chiffre des unités d'un produit :

1. Construire un programme basé sur un algorithme naïf déterminant le chiffre des unités d'un produit d'entiers.

2. Autre algorithme :
   (a) Justifier que le chiffre des unités de $a \times b$ est le même que celui du produit des chiffres des unités de a et b. En d'autres termes, cette propriété s'écrit : $(ab \mod 10) = [(a \mod 10) \times (b \mod 10)] \mod 10$.
   (b) En déduire un nouveau programme de calcul du chiffre des unités d'un produit.

3. Généralisation :
   (a) Prouver que pour tout $a,\ b \in \mathbb{N}$ et $c \in \mathbb{N}^*$ tel que $c \geq 2$, on a $(ab \mod c) = [(a \mod c) \times (b \mod c)] \mod c$.
   (b) En déduire un algorithme de calcul rapide de $a^b \mod c$.
   (c) Implémenter l'algorithme précédent sous Python.