Travaux pratiques d'introduction 3

Les boucles «for»

Informatique tronc commun MPSI et PCSI

I Présentation

I.1 Introduction

Définition 1 : Boucles

Une boucle est une séquence d'instructions qui se répète. On distingue deux types de boucles :

- Les boucles inconditionnelles : les boucles «for» sont adaptées lorsqu'on connaît à l'avance le nombre de fois qu'une séquence devra se répéter (on parle également de boucles bornées).
- Les boucles conditionnelles : les boucles «while» feront l'objet d'un autre TP, sachez néanmoins que ces boucles présentent l'avantage d'effectuer une séquence d'instructions sans en connaître à l'avance le nombre de répétitions (on parle également de boucles non bornées).

I.2 Syntaxe

Les instructions Python pour répéter n fois un ensemble d'instructions sont

```
for i in range(n):
    instruction 1 à répéter
    instruction 2 à répéter
   ...
    instruction p à répéter
suite des instructions après répétition
```

Cette syntaxe se traduit par «pour i allant de 0 à n – 1 faire les instructions 1 à p». L'indice de boucle `i` prend successivement les valeurs $0, 1, 2 \ldots, n-2, n-1$. Donc `i` prend bien n valeurs différentes.
Les instructions à répéter forment un bloc qui est repéré par l'indentation supplémentaire (décalage de quatre espaces ou d'une tabulation). Notons que ce bloc est précédé du symbole de ponctuation « :» comme lors de la définition d'une fonction (pour rappel, dans la syntaxe python, toute instruction qui nécessite un bloc d'instructions est suivie du deux-points « :»).
Le script suivant permet d'afficher le carré des nombres de 0 à 9 :

```
for i in range(10):
     print(i*i)
```

L'indice «i» prend bien les valeurs $0, 1, 2, \ldots, 8, 9$ soit 10 valeurs.
Autre exemple, la fonction suivante calcule la somme des n premiers carrés :

```
def somme(n):
    s=0
    for i in range(n+1):
        s=s+i*i
    return s
```

Jusqu'ici, on a utilisé l'opération range avec un unique paramètre décrivant le nombre de répétitions à effectuer. La syntaxe complète de cette opération est en réalité plus riche et permet de créer une suite finie d'entiers en lui donnant 3 paramètres :
`range(début, fin, pas)`

- qui commence par la valeur `début` (0 par défaut),
- qui progresse de la valeur `pas` à chaque étape (1 par défaut),
- qui s'arrête dès qu'elle dépasse (ou atteint) la valeur `fin`.

L'opération `range(n)` est donc équivalente à `range(0, n, 1)` ou encore à `range(0, n)`.
L'opération `range(1, n+1, 1)` permet d'obtenir le même nombre de répétitions mais avec un indice qui va de 1 à n inclus.
L'opération `range(1, n, 2)` permet d'obtenir des indices impairs qui vont de 1 à n exclus.
L'opération `range(n, 0, -1)` permet d'obtenir des indices décroissants qui vont de n à 1 inclus.
Remarques :

- En python, les chaînes de caractères sont considérées comme un certain type de collections d'objets, en l'occurrence des caractères. On peut donc accéder aux éléments de cette collection à l'aide d'une boucle `for`.

  ```
  mot='Bonjour'
  for c in mot :
      print(c)
  ```

  La variable `c` prend bien les caractères 'B', 'o', 'n', 'j', 'o', 'u' et 'r'.

- Si l'indice de boucle n'est pas utilisé dans le bloc d'instructions, on peut le remplacer par le caractère «_» qui représente une variable sans nom.

  ```
  for _ in range(4) :
      print('Bonjour')
  ```

  Ce script affiche 4 fois la chaîne de caractères 'Bonjour'.