La boucle bornée

Site: ISN Space
Cours: Les boucles en python
Livre: La boucle bornée
Imprimé par: Kazi-Aoual Chawky
Date: Saturday 2 May 2020, 12:31

Table des matières

Les boucles s’utilisent pour répéter plusieurs fois l'exécution d’une partie du programme.

En général, quand on sait combien de fois doit avoir lieu la répétition, on utilise une boucle bornée **for** .

La fonction range()

La fonction **range()** permet de créer une **liste**. Elle peut être utilisée avec 3 paramètres **range(valeur_debut, valeur_fin, pas).**

- **valeur_debut** : contient le numéro de départ de la liste,
- **valeur_fin :** le dernier nombre de la liste et finalement
- **pas :** l'incrément entre chaque nombre généré.

Une **liste** en python est une structure de données qui contient une série de valeurs. Les valeurs sont séparées par des **virgules**, et le tout encadré par des **crochets.**

Exemples :

```
range(0,10)

Out[7]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
range(5)

Out[8]: [0, 1, 2, 3, 4]
```

Si la **valeur_debut** n'est pas spécifiée, elle est implicitement mise à zéro.

```
range(10,40,5)

Out[9]: [10, 15, 20, 25, 30, 35]
```

Lorsqu'elle est omise, la valeur **pas** est implicitement égale à 1. Cependant, elle peut être n'importe quelle valeur non nulle.

Boucle For avec range()

Syntaxe :

```
for i in range(valeur_debut,valeur_fin,pas):
    instruction
```

L'instruction **for** est une **instruction** composée, c'est-à-dire une instruction dont l'en-tête se termine par deux-points (**:**), suivie d'un bloc indenté qui constitue le **corps de la boucle**.

La variable **i** prendra successivement toutes les valeurs contenues dans (**in**) la liste crée par la fonction **range()**. La première sera ***valeur_debut*.** On dit que cette variable est un **compteur** car, elle sert à numéroter les **itérations** de la boucle. On réalise une **itération** de la boucle à chaque fois que le corps de la boucle est exécuté.

Donc, le nombre **d'itérations** = ***valeur_fin* - *valeur_debut***

La boucle inclut toujours ***valeur_debut*** et exclut ***valeur_fin*** pendant l'itération.

Et par conséquent, les expressions ***valeur_debut*** *et* ***valeur_fin*** constituent les bornes de la boucle, d'où la notion de **boucle bornée**.

Exemple:

```
for i in range(1,4):
 print("Eh oh !")
print("Il y'a quelqu'un ?")
```

Tester ici l'exemple. Cliquer pour masquer l'éditeur

Affichage après exécution :

```
Oh eh !
Oh eh !
Oh eh !
Il y'a quelqu'un ?
```

Il est possible d'obtenir le même résultat sans donner la liste des valeurs avec une forme réduite de range () : **range(valeur_fin)** , la **valeur_debut** est implicitement mis à zéro:

Exemple :

```
for i in range(3):
 print("Eh oh !")
print("Il y'a quelqu'un ?")
```

Tester ici par vous-même

Pour itérer sur une séquence décroissante, on peut utiliser une forme étendue de **range ()** . Celle à trois arguments **range(*valeur_debut*, *valeur_fin*, *pas*).** Lorsqu'elle est omise, la valeur **pas** est implicitement égale à 1. Cependant, elle peut être n'importe quelle valeur non nulle.

Exemple :

```
for i in range(10, 0, -2):
    print(i)
```

Tester ici par vous-même

Les boucles imbriquées

Quand l'instruction à exécuter à l'intérieur d'une boucle est elle aussi répétitive, le corps de cette boucle contient une seconde boucle et on dit que ces deux boucles sont imbriquées. Les bornes de la boucle interne dépendent souvent du compteur de la boucle externe.

À vous de jouer 1 :

Essayez de deviner les résultats le script suivant avant de le tester.

```
for i in range(1,4):
    for j in range(0,2):
      print(i*j)
```

Tester par vous-même

À vous de jouer 2 :

Pensez-vous que les résultats du programme ci-dessous seront les mêmes que ceux de l'exemple 4 ?

Essayez de répondre à cette question avant de tester le programme.

```
for i in range(0,3):
    for j in range(0,2):
      print(i*j)
```

La boucle bornée et les chaines de caractère

Il arrive très souvent qu'on traite des variables de chaîne de caractère, caractère par caractère, du premier jusqu'au dernier, pour effectuer à partir de chacun d'eux une opération quelconque. Nous appellerons cette opération un **parcours**.

Exemple :

```
nom ="Steve jobs"
for caracter in nom:   # la variable caracter prend la valeur de tous les ca
ractères de la variable nom
    if caracter == "j":
        caracter = "J"
    print(caracter)
```

Tester par vous-même

À vous de jouer

Écrire un script qui permet de corriger le nom suivant : "bill,gates" en "Bill Gate"