Python

Structure de données de base

Yannick CHISTEL

Lycée Dumont d'Urville - CAEN

7 octobre 2020

Les boucles

Une instruction qui doit être répétée 100 fois n'est pas écrite 100 fois dans un programme. Pour cela, on utilise une boucle.

La structure de boucle est appelée une structure **itérative**. La boucle est utilisée pour répéter autant de fois que nécessaire une ou plusieurs instructions. Il en existe deux en python :

- La boucle **FOR** qui répète **un nombre fini et connu de fois** les instructions contenues dans la boucle. C'est une **boucle bornée**.
- La boucle **WHILE** répète les instructions contenues dans la boucle **tant que la condition donnée est vraie**.

Remarques

1. Les instructions dans une **boucle** constituent un mini programme qui sera exécuté autant de fois que la boucle le demande.
2. Une **boucle** peut contenir d'autres **boucles**. On parle alors de boucles imbriquées.

La boucle FOR

Présentation générale

La boucle **for** prélève dans l'ordre les valeurs présentes dans un ensemble. La structure est la suivante :

- **for** clef **in** valeurs **:**
  instruction(s)

Exemple

Supposons l'ensemble constitué des valeurs "mot", 13, "deux", 6, 125 et "quatre" :

- **for** c **in** ["mot",13,"deux",6,125,"quatre"] :
  print(c,end=" ")

On obtient l'affichage : "mot" 13 "deux" 6 125 "quatre"

Remarque

1. La clef peut être une lettre, un mot ou même le caractère souligné _.
2. Si l'ensemble des valeurs est composé de nombres, alors il est possible d'effectuer des calculs avec la clef.
3. Une chaine de caractères constitue un ensemble de valeurs. La clef prend alors pour valeur chacune des lettres, y compris les espaces et les symboles de ponctuation (voir en exercice).

La boucle FOR

Ensemble de nombres : RANGE

La fonction **range(paramètres)** permet à une boucle d'itérer sur une suite de nombres.
La fonction **range** construit un ensemble de nombres suivant les paramètres passés en arguments.

- **range(n)** renvoie les nombres entiers positifs de 0 jusqu'à $n-1$
- **range(p,n)** renvoie les nombres entiers positifs de p jusqu'à $n-1$
- **range(p,n,k)** renvoie les nombres entiers positifs de p, $p+k$, $p+2k$, ... jusqu'à $n-1$

Exemple

- range(5) crée l'ensemble de nombres $0, 1, 2, 3, 4$
- range(2,5) crée l'ensemble de nombres $2, 3, 4$
- range(1,5,2) crée l'ensemble de nombres $1, 3$

La boucle FOR

Avec un indice de boucle

On introduit une variable de boucle accessible à l'intérieur de la boucle et dont la valeur indique le numéro du tour de boucle.
Cette variable est appelée **indice de boucle** ou **compteur de boucle**.
La structure est la suivante :

- **for "indice de boucle" in range**(nombre de répétitions) **:**
  instructions

Exemple

- for i in range(4) :
  print(i,end="<")

On obtient alors l'affichage : $0 < 1 < 2 < 3 <$

Remarque

Pour N répétitions d'une boucle **FOR**, l'indice commence à 0 et se termine au nombre de répétitions - 1, soit $N - 1$. On a bien N répétitions au total.

Indice de début différent de 0

L'indice de boucle démarre à 0. Il est possible de modifier ce comportement en ajoutant le paramètre de début de boucle.
La structure est la suivante :

- **for "indice de boucle" in range**(indice de début, nombre de répétitions) **:**
  instructions

Exemple

- for *i* in range(2,5) :
  print(*i*,end="<")

On obtient alors l'affichage : $2 < 3 < 4 <$

Remarque

Si aucun indice de début n'est indiqué, l'indice de boucle démarre à 0 par défaut.

Incrément de boucle différent de 1

L'indice de boucle augmente de 1 à chaque tour. On dit que l'indice ou compteur de boucle est **incrémenté** de 1. On peut modifier **l'incrément** d'une boucle en ajoutant le paramètre d'incrémentation après le nombre de répétitions.
La structure est la suivante :

- **for "indice de boucle" in range**(indice de début, nombre de répétitions, incrément) **:**
  instructions

Exemple

- for *i* in range(0,5,2) :
  print(*i*,end="<")

On obtient alors l'affichage : $0 < 2 < 4 <$

Remarque

Si aucun incrément de boucle n'est indiqué, l'incrément vaut 1 par défaut.
Il faut donné l'indice de début même s'il vaut 0.

La boucle FOR

Répétition d'une instruction avec "variable"

Il est possible d'utiliser une variable pour indiquer le nombre de répétitions à condition qu'une valeur lui soit allouée avant l'exécution de la boucle.

Exemple

Demander le nombre **n** de répétitions avant la boucle.

- n=int(input("Nombre de répétitions :"))
  for _ in range(n) :
      print("A")

La boucle FOR

Répéter un bloc d'instructions

Il est possible de répéter plusieurs instructions à la suite et dans le même ordre. La structure est la suivante :

- **for** _ **in range**(nombre de répétitions) **:**
  instruction 1
  instruction 2
  instruction 3
  ...

Exemple

- $x = 1$
  for _ in range(4) :
  $x = 2 * x$
  print(x,end=" ")

On obtient alors l'affichage : 2 4 8 16

Avec un accumulateur

Il est possible d'utiliser des variables dans une boucle qui ont une valeur qui change à chaque tour de boucle. Ces variables sont appelées des **accumulateurs de boucles**.

Exemple

Calculer la somme de 10 premiers nombres entiers :

```
S=0
for i in range(10) :
    S=S+i
print("S=",S)
```

La variable S est **initialisée** à 0 avant la boucle.

i	0	1	2	...	8	9
S	0	1	3	...	36	45

On obtient l'affichage : $S = 45$
La variable **S** est un accumulateur de boucle.

Remarque

Une variable accumulateur de boucle doit être initialisée avant la boucle.

Les boucles imbriquées

Une boucle peut contenir une autre bloucle. Ce sont des boucles imbriquées.

Exemple

Afficher un même caratère sous la forme d'un carré de côté 5 !

-
```
S=0
for _ in range(5) :
    for _ in range(5) :
        print("O",end="")
    print()
```

```
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
```

Le retour à la lignese fait avec la fonction **print()** sans paramètres.

Remarque

Le nombre de boucle imbriquées n'est pas limité !

La boucle WHILE

Définition

Une boucle **WHILE** s'exécute tant qu'une condition (test) est vérifiée. Dès que la condition est fausse, on sort de la boucle et les instructions situées après la boucle sont exécutées.

La condition est un test dont la valeur de retour est de **type booléen**.

Exemple

- $i = 2$
  while $i <= n$:
  print(i)
  $i = i * 2$
  print("fin")

Le nombre de tours de boucle varie selon la valeur de n.

FOR ou WHILE

Définition

Le choix de la boucle dépend du contexte. Mais 2 principes sont à respecter :

1. Si on connait le nombre d'itérations (tours de boucles) à l'avance, on s'orientera vers une boucle FOR.
2. La boucle WHILE peut entrainer une boucle infinie (qui ne s'arrête jamais) car la condition est toujours vraie. Il est indispensble de bien réfléchir à la condition et de faire des tests.

Exemple de boucle infinie

- $k = 1$
  while $k > 0$:
  $k = k * 2$
  print("k est toujours positif. La boucle est infinie !")

Seul un appui sur les touches **ctrl C** stoppera la boucle.