## Thème 6 - Langages et programmation

03

# Les Boucles FOR en Python



## 1. Les énumérables ou itérables

En mathématiques, on dit qu'un ensemble est *dénombrable* lorsqu'on peut associer à chaque élément de l'ensemble un nombre (traditionnellement 1, 2, 3 ...)

- les voitures qui roulent sur l'autoroute sont dénombrables.
- l'eau qui coule d'un robinet n'est pas dénombrable.

En informatique, il existe un concept similaire qui va désigner les objets que l'on peut **énumérer**, c'est-à-dire les décomposer en une succession ordonnée d'éléments. On les appelle les **énumérables** ou les **itérables** (Python utilise le mot anglais iterable).

- la variable NSI (qui est de type String) est énumérable : on peut la décomposer en N, S, I.
- la variable [4, 3, 17] (qui est de type List) est énumérable : on peut la décomposer en 4, 3, 17.
- la variable 5 (qui est de type Int ) n'est PAS énumérable : on ne peut pas la décomposer.

#### 1.1. Itérer sur les itérables : la boucle for

#### → Itérer sur une chaîne de caractères

Il existe donc une instruction permettant de faire une (ou plusieurs) action(s) à chaque itération sur un élément énumérable.

#### Exemple:

Le programme suivant :

```
for k in 'NSI':
    print(k)
```

va donner ceci:

```
Script Python

N
S
I
```

Analyse grâce à PythonTutor Étudions, grâce à PythonTutor, le détail de cette exécution.

Cliquez sur Next et observez bien l'évolution de la variable k.



La variable k prend donc **successivement** toutes les lettre de la chaîne de caractère "NSI".

Pour chaque valeur de k, la ou les instruction(s) situées de manière indentée sous la ligne du for seront exécutées.

Ici, il y a simplement un print(k), donc chaque lettre de "NSI" s'affiche l'une après l'autre.

```
Exercice 1:
Que donne le script suivant ?
Script Python
```

for m in 'NASA':
 print("bonjour")

Dans cet exercice, la variable de boucle m est muette : elle n'apparaît dans les instructions indentées sous le for .

La variable m prend successivement les valeurs 'N, 'A', 'S' et 'A', mais on ne le voit pas.

## 🖊 🖀 A retenir :

#### **△ Comment éviter les erreurs classiques**

Quand vous écrivez une boucle for , veillez bien à :

- finir la ligne du for par les deux points :
- indenter sous le for les instructions qui doivent être répétées.

```
Script Python

for ....:
instructions à répéter
```

#### → Itérer sur une liste

**Exemple:** Le programme suivant:

```
Script Python

for jour in ["lundi", "mardi", "mercredi", "jeudi", "vendredi"]:
    print(f"je vais au lycée le {jour}")
```

va donner ceci:

```
& Script Python
```

```
je vais au lycée le lundi
je vais au lycée le mardi
je vais au lycée le mercredi
je vais au lycée le jeudi
je vais au lycée le vendredi
🗋 Algo
from tutor import tutor
for jour in ["lundi", "mardi", "mercredi", "jeudi", "vendredi"]:
    print(f"je vais au lycée le {jour}")
tutor()
🗋 Algo
je vais au lycée le lundi
je vais au lycée le mardi
je vais au lycée le mercredi
je vais au lycée le jeudi
je vais au lycée le vendredi
                                                                                           Print output (drag
     for jour in ["lundi", "mardi", "mercredi", "jeudi", "vendredi"]:
   2
            print(f"je vais au lycée le {jour}")
line that has just executed
next line to execute
                                                                                           Frames
                   << First
                             < Back | Step 1 of 11 | Forward >
                                                              Last >>
```



Très souvent, l'objet itérable que la boucle va parcourir aura été au préalable stocké dans une variable :

**Exemple :** Le programme précédent est équivalent à :

```
& Script Python
```

```
semaine = ["lundi", "mardi", "mercredi", "jeudi", "vendredi"]
for jour in semaine:
    print("je vais au lycée le", jour)
```

Notez l'importance d'avoir choisi des noms de variables explicites : ils aident grandement à la lisibilité du code.

Trailer: Dans le cours spécifique sur les listes, nous verrons une toute autre manière de parcourir une liste.

```
🗋 Algo
semaine = ["lundi", "mardi", "mercredi", "jeudi", "vendredi"]
for jour in semaine:
    print("je vais au lycée le", jour)
```

```
🗋 Algo
je vais au lycée le lundi
je vais au lycée le mardi
je vais au lycée le mercredi
je vais au lycée le jeudi
je vais au lycée le vendredi
```

#### 1.2. ◆ Comment répéter n fois la même action ?

Comment répéter 10 fois la phrase "Hasta la vista, baby" ?

```
✗ Note : range
```

Le programme suivant :

```
🗞 Script Python
for i in range(5):
    print("Hasta la vista, baby")
```

va donner ceci:

```
% Script Python
Hasta la vista, baby
```

Là encore, le i est une variable muette.

```
↑ Algo
```

```
for i in range(5):
    print("Hasta la vista, baby")
```

```
Hasta la vista, baby
```

#### Utilisation minimale de l'objet range()

Syntaxe minimale de range(): Le programme suivant :

```
Script Python

for k in range(4):
    print(k)
```

va donner ceci:

```
Script Python

0
1
2
3
```

```
A retenir :
```

Faire parcourir à une variable k l'ensemble range(n) va faire prendre à k les valeurs 0, 1, 2, ..., n-1.

#### Utilisation avancée de l'objet range()

Syntaxe complète de range(): Le programme suivant :

```
Script Python

for k in range(5, 15, 2):
    print(k)
```

va donner ceci :

```
Script Python
5
7
9
11
13
```

## 💙 🤮 A retenir :

faire parcourir à k l'ensemble range(start, stop, step) fait:

- démarrer k à la valeur start
- progresser k de step en step
- tant que k est rictement inférieur stop

#### Remarques:

- si le step est omis, il vaut 1 par défaut.
- l'objet range(5) n'est pas «techniquement» égal à la liste [0, 1, 2, 3, 4], car ce n'est pas un objet de type List:

## Exercice 1 :

Faire afficher les séries de nombres suivantes.

On utilisera la syntaxe print(k, end = ' ') ) pour afficher les nombres horizontalement.

- A. 0 1 2 3 4 5
- B. 10 11 12 13 14 15
- C. 3 6 9 12
- D. 10 9 8 7 6 5 4 3 2 1 0

#### 1.3. ◆ Une situation classique : la double boucle

Il est très souvent utile d'imbriquer une boucle dans une autre, notamment lors du parcours de tous les pixels d'une image. Prenons pour l'instant un exemple numérique.

### 

Ecrire un script permettant d'afficher

#### **%** Script Python

```
1 * 1 = 1

1 * 2 = 2

1 * 3 = 3

2 * 1 = 2

2 * 2 = 4

2 * 3 = 6

3 * 1 = 3

3 * 2 = 6

3 * 3 = 9

4 * 1 = 4

4 * 2 = 8

4 * 3 = 12
```

## 💙 쭽 A retenir :

- La boucle for s'utilise lorsque :
  - on veut parcourir un à un les éléments d'un objet itérable (une chaîne de caractère, une liste...)
  - on veut répéter une action un nombre de fois **connu à l'avance**. On parle de boucle **bornée**.
- Les instructions répétées peuvent mais ce n'est pas obligatoire faire appel à la variable de boucle, mais il ne faut pas que ces instructions la modifient.
- Ne pas oublier les : et l'indentation !
- range(n) génère une séquence de n nombres entiers: on s'en servira dès qu'on aura besoin de répéter n fois des instructions.

## 2. Exercices

### **№** Exercice 3:

Dans l'extrait de code suivant:

- chaine est une variable initialisée avec un str vide : "";
- on veut qu'en sortie de programme cette variable contienne la valeur 'bravo'.

L'idée est d'ajouter une par une les lettres à la variable chaine.

Compléter le code.

Cette variable chaine est appelée un accumulateur.

## Exercice 4 :

En Python, la fonction ord renvoie le code Unicode d'un caractère et la fonction chr le contraire: elle renvoie le caractère correspondant à un code Unicode.

Par exemple:

```
Script Python

>>> ord('a')
97
>>> chr(97)
'a'
```

Voici une liste contenant les codes Unicode des lettres d'un mot secret...

À vous d'écrire un programme où en sortie, la variable mot\_secret contiendra la chaîne de caractères de ce mot.

```
Script Python

mystere = [111, 107, 44, 32, 98, 105, 101, 110, 32, 106, 111, 117, 233]
mot_secret = ""
```

ok, bien joué

#### **№** La Exercice 5:

On souhaite calculer la somme des 1000 premiers nombres entiers naturels, c'est-à-dire:

$$1+2+3+4+5+\cdots+999+1000$$

Écrire un programme avec une variable somme **accumulateur** (comme à l'exercice 3) qui contiendra la valeur souhaitée en fin de programme.

### Exercice 6 :

Ecrire un programme permettant de calculer

$$1 \times 2 \times 3 \times \dots 99 \times 100$$
.

### Exercice 7:

Proposer un code qui écrit la table de multiplication de 7, de 8 et de 9.