Python au lycée (3) : Les boucles

L'un des intérêts de la programmation est de pouvoir faire exécuter facilement à une machine des tâches **répétitives**.

Le langage Python propose deux instructions : « for » et « while » qui permettent de répéter automatiquement l'exécution de certains blocs de code.
\id{h05}

1. Les boucles « for » (Boucles bornées)

Une boucle « for » (ou boucle « pour » ou boucle bornée) est généralement utilisée lorsque l'on connaît le nombre de répétitions que l'on souhaite exécuter.

La syntaxe de cette instruction est la suivante :

```
for variable in [liste de valeurs] :
   # bloc d'instructions à répéter
# instructions à exécuter une fois la boucle terminée
```

Ce programme se déroule de la manière suivante :

- les « instructions à répéter » sont exécutées en donnant à la « variable » chacune des valeurs de la « liste de valeurs » ; c »est l'indentation (écriture décalée vers la droite) qui détermine la taille du bloc d'instructions à répéter
- une fois que tous les items de la liste ont été parcourus, le programme passe au « instructions à exécuter une fois la boucle terminée ».

Par exemple, le programme Python ci-dessous :

```
for i in [1, 2, 5, 11] :
   j = i**2 # calcul du carré de i
```

```
   print(j, end=' - ') # affichage~; on sépare les valeurs par des tirets
print("fin")
```

affichera :
1 – 4 – 25 – 121 – fin
ce qui correspond aux carrés des nombres de la liste.

Remarque : on aurait pu faire l'économie de la variable j, en écrivant plus simplement « print(i**2, end=' – ') » mais le but, ici, était de montrer qu'un bloc pouvait comporter plusieurs lignes.

Avec l'instruction for on utilise fréquemment la fonction range ;
en effet, range(a,b) renvoie la liste des entiers compris (au sens large) entre a et b – 1.

> **Attention**
>
> L'instruction range(a,b) créer une liste qui s'arrête à l'entier b -1 et non à l'entier b !

Par exemple, le programme suivant affiche les doubles des nombres entiers compris entre 3 et 5 :

```
for i in range(3, 6) :
   print(2*i, end=' - ') # affiche 6 - 8 - 10 -
```

Remarque : si l'on utilise l'instruction range avec un seul paramètre b, celle-ci retournera la liste des entiers compris entre 0 et b – 1 :

```
for i in range(4):
   print(i, end=' - ') # affiche 0 - 1 - 2 - 3 -
```

\id{h10}

2. Les boucles « while » (Boucles non bornées)

On utilise une boucle while (ou boucle « Tant que » ou boucle non bornée) lorsque l'on doit répéter l'exécution d'un bloc d'instructions **tant qu'une condition est vérifiée** (mais en général, on ne sait pas au préalable le nombre de répétitions que l'on devra effectuer). Là encore, c'est l'indentation qui détermine la fin du bloc d'instructions à répéter.

La syntaxe de l'instruction « while » est :

```
while condition :
   #bloc d'instructions à répéter
#instructions à exécuter une fois la boucle terminée
```

Le programme se déroule alors de la façon suivante :

- tant que la « condition » de la ligne 1. est vraie, les « instructions à répéter » de la ligne 2. sont exécutées
- dès que la « condition » de la ligne 1. devient fausse, le programme passe aux « instructions à exécuter une fois la boucle terminée » (ligne 3.).

Par exemple, le programme ci-dessous affiche la plus petite puissance de 2 qui est supérieure ou égale à 1 000 :

```
i = 1 # initialisation de i
while i < 1000 :
   i = i * 2
print(i) # affiche 1024
```

À chaque passage à la ligne 3., on multiplie la valeur précédente de de i par 2. On obtient ainsi les puissances successives de 2.

Pour bien comprendre comment fonctionne ce programme, il peut être utile de représenter les différentes valeurs de i et de la condition i < 1000 dans un tableau :

i	1	2	4	8	16	32	64	128	256	512	102
$i < 1000$	vrai	vrai	vrai	vrai	vrai	vrai	vrai	vrai	vrai	vrai	fau

L’utilisation de l’instruction while peut être assez délicate ; si, suite à une erreur de programmation, la condition figurant dans le while est toujours vérifiée, le programme ne sortira jamais de la boucle et tournera indéfiniment (sauf si un intervenant extérieur met fin à son exécution...)

En particulier, il faut faire attention aux points suivants :

- penser à initialiser les variables avant l’instruction while : dans notre exemple, python provoquerait une erreur si la ligne 1. (initialisation de i) était manquante.
- la condition figurant après l’instruction while est celle qui permet de **rester** dans la boucle ; c’est donc le **contraire** de la condition de **sortie** de boucle. Dans l’exemple précédent, on souhaitait **sortir** de la boucle lorsque la valeur de i était **supérieure ou égale à 1000** ; il fallait donc coder i < 1000 à l’intérieur de l’instruction while.