TP Python 3 : boucles

L.-C. LEFÈVRE

Une **boucle** permet de répéter automatiquement une instruction ou un bloc d'instructions. C'est à partir de maintenant que les choses deviennent intéressantes et que le programme peut effectuer une suite d'opérations qu'il serait trop long de faire avec sa calculatrice !
Comme avec les conditions, le **corps** de la boucle est la partie qui est répétée et est indentée vers la droite ; le code qui n'est plus indenté ne fait plus partie de la boucle.

I La boucle `for`

La boucle `for` répète les instructions un certain nombre de fois. Au cours de la boucle, une variable appelée **compteur** varie entre un nombre de départ (*start*) et un nombre final (*stop*), spécifiés par le mot-clé `range` (qui pourrait se traduire par *intervalle* ou *étendue*). Observer le code ci-dessous où la variable `x` est le compteur :

```
for x in range(0, 5):
    print("passage dans la boucle avec la valeur", x)
```

Remarque importante. La valeur finale du `range` n'est pas affichée ! En effet il faut comprendre que la boucle ci-dessus s'exécute seulement pour $x < 5$ (strictement plus petit). C'est en fait assez pratique quand on compte à partir de zéro : d'ailleurs il y a exactement 5 passages dans la boucle ci-dessus, et il y en aurait 6 si on continuait pour $x = 5\ldots$
Dans la suite, nous afficherons souvent la variable compteur même si nous ne nous en servons pas, pour vérifier que les bornes sont bien correctes.

Il est intéressant de faire des calculs sur la variable `x` qui varie. Par exemple que fait le programme suivant ?

```
for x in range(0, 10):
    print(x, x**2)
```

Question 1. Dans le chapitre *fonctions de référence*, il fallait parfois remplir un tableau de cette forme :

x	0	1	2	...	20
x^3					

Écrire un programme qui le remplit pour nous en entier (attention aux bornes !)

Question 2. On souhaite maintenant remplir un tableau qui ressemble à ceci :

x	1	1,1	1,2	...	2
x^3					

Pour cela, au lieu de travailler avec des nombres à virgule et de faire sauter x de 0,1, on fait varier une autre variable y puis on la divisera par 10 ! Ainsi quand y saute de 1 alors x saute de 0,1. Trouver les bonnes bornes :

```
for y in range(...):
    x = y / 10
    print(...)
```

II Accumulateurs

On souhaite calculer la somme $1+2+3+\cdots+100$. Pour cela il faut une boucle avec un compteur qui varie de 1 à 100. À chaque passage dans la boucle une variable appelée **accumulateur** calcule la somme « jusque là ».

```
s = 0
for x in range(...):
    print(x, s)
    s = s + x
print("résultat final :", s)
```

Question 3.

1. Compléter le programme pour calculer la somme $1+2+3+4$, et vérifier à la main, en suivant bien toutes les étapes de la boucle.
2. Calculer alors la somme jusqu'à 100 et, si le programme a l'air de bien marcher, enlever la ligne `print(x, s)`.
3. Donner directement le résultat de la somme jusqu'à 1000.

III La boucle `while`

La boucle `while` est répétée **tant que** (c'est la traduction exacte de *while*) une condition est réalisée. La syntaxe est exactement la même que pour les conditions `if` : si la condition est vraie, le bloc d'instructions est exécuté, puis la condition est à nouveau testée et éventuellement les instructions sont répétées... Pour que cela soit intéressant, il faut que la condition dépende de variables qui changent au fur et à mesure du programme.
La boucle suivante est exactement la même que la toute première de ce TP. L'observer et tenter de comprendre l'exécution du programme :

```
x = 0
while x < 5:
    print("passage dans la boucle avec la valeur", x)
    x = x + 1
print("fin avec la valeur", x)
```

Remarque importante. Si on oublie la ligne `x = x + 1` alors : au départ `x` vaut `0` et la condition `x < 5` est vraie. Le texte est donc affiché. Puis comme rien ne fait varier `x`, la condition est testée à nouveau et est encore vraie... et cela continue et rien n'arrête le programme ! On obtient une **boucle infinie**. C'est une très grande source d'erreurs dans les programmes informatique. Il faut donc être particulièrement attentif avec les boucles `while`.

Question 4. Dans la suite de nombres $1, 2, 4, 8, 16, \ldots$ (on multiplie par 2 à chaque fois), on souhaite savoir au bout de combien d'étapes on dépasse un million. On pourrait faire une boucle qui calcule un nombre fixe de termes, de cette forme :

```
n = 1
for x in range(0, 10):
    print(x, n)
    n = 2 * n
```

cependant une boucle `while` est plus adaptée avec la condition de s'arrêter quand on dépasse le million !

1. D'abord avec la boucle `for` ci-dessus, en essayant un peu plus loin que 10 termes, donner la réponse.
2. Écrire une boucle `while` équivalente à celle ci-dessus.
3. Quand elle fonctionne bien, changer la condition dans le `while` pour continuer *tant que* `x` est inférieur à un million. Afficher alors la valeur de `x` et de `n` en fin de boucle.
4. Donner directement la réponse à : quand cette suite de nombres dépasse 10^{50} ?