Chapitre 1.2 : if, for, while

I Instruction conditionnelle

Une instruction conditonnelle se présente sous la forme :

```
if condition 1:
    bloc d instructions
elif condition 2:
    bloc d instructions
...
else:
    bloc d instructions
```

Les blocs **elif** et **else** ne sont pas indispensables. Un seul bloc d'instructions au maximum est exécuté : les conditions sont lues dans l'ordre et pour la première qui est vraie (s'il y en a une) on exécute son bloc d'instructions.
Par exemple l'exécution de :

```
x = 3
if x > 0:
    x = -x
    print(x)
elif x > 1:
    x = x - 1
    print(x)
else:
    print(x)
```

affiche ...

```
x = 3
if x > 1:
    x = x - 1
    print(x)
elif x > 0:
    x = -x
    print(x)
else:
    print(x)
```

affiche ...

```
x = 3
if x > 1:
    x = x - 1
    print(x)
if x > 0:
    x = -x
    print(x)
else:
    print(x)
```

affiche ...

Remarquez les indentations (qui sont mises automatiquement sous Pyzo) : elles permettent de délimiter quel bloc d'instructions correspond à quelle condition.

II Boucles

Lorsqu'on souhaite répéter plusieurs fois un même bloc d'instructions, on utilise une boucle. Si l'on sait a priori le nombre d'itérations, on utilise plutôt une boucle **for**.

Une telle boucle se présente sous la forme :

```
for indice in range(debut, fin+1, pas):
      bloc d instructions
```

L'indice est une variable qui n'existe que dans la boucle : on précise sa valeur de départ (optionnel) et sa valeur de fin. À chaque passage dans la boucle, le bloc d'instructions est exécuté et la valeur de l'indice augmente de la valeur du `pas`. On peut aussi utiliser `range(debut, fin+1)` et le pas vaut automatiquement 1, ou encore `range(fin+1)` et le début vaut 0 et le pas 1.
Par exemple :

```
p = 1
for i in range(10):
    p = (i+1) * p
    print(p)
```

affiche ...

```
s = 0
for i in range(3, 10, 2):
    s = s + i
    print(s)
```

affiche ...

```
for i in range(10, 0, -1):
    print(i)
```

affiche ...

Lorsqu'on ne connait pas a priori le nombre d'itération à faire, on utilise une boucle **while** (tant que en français) :

```
while condition:
    bloc d instructions
```

On peut parfois utiliser une variable qui permet de compter le nombre de passages dans la boucle. On appelle cette variable un **compteur** : il est initialisé à 0 avant la boucle et incrémenté de 1 à chaque itération.
Par exemple :

```
n = 1234
while n > 0:
    print(n)
    n = n // 10
```

affiche ...

```
u = 0
n = 0
while u < 10:
    u = u + 2
    n = n + 1
print(n)
```

affiche ...

Pour les deux types de boucles, on peut sortir prématurément de la boucle en utilisant le mot clé **break**.

III Exercices

Exercice 1 Écrire ce qui s'affiche dans la console lors de l'exécution des instructions suivantes :

1. .
```
a, b, c = 0, 2, 1
if a == 0:
    b = 4
else:
    c = 5
b = 1
print(a, b, c)
```
2. .
```
a, b, c = 0, 2, 1
if a == 0:
    b = 4
else:
    c = 5
    b = 1
print(a, b, c)
```
3. .
```
x = 3
if x % 2 == 1:
    x = x + 1
if x % 2 != 1:
    x = x // 2
print(x)
```

4. .

```
x = 3
if x % 2 == 1:
    x = x + 1
elif x % 2 != 1:
    x = x // 2
print(x)
```

5. .

```
for i in range(3, 6):
    print(i)
```

6. .

```
s = 0
for i in range(3):
    s = s + i
print(s)
```

7. .

```
s = 0
for i in range(3):
    s = s + i
    print(s)
```

8. .

```
for i in range(3, 6, 2):
    print(i)
```

9. .

```
for i in range(6, 2, -1):
    print(i)
```

10. .

```
for i in range(3):
    if i % 2 == 0:
        print(i)
    else:
        print(2*i)
```

11. .

```
c = 0
while True:
    c = c + 1
```

12. .

```
s = 0
i = 0
while s < 10:
    s = s + i
    i = i + 1
print(s, i)
```

13. .

```
from math import cos
p = 1
while cos(p) > 0:
    p = p + 0.01
print(2*p)
```

■

Exercice 2

1. Une année est bissextile si elle est divisible par 4 mais pas par 100, ou bien si elle est divisible par 400. Écrire un script qui définit une variable `annee` puis teste si elle est bissextile ou non
2. Écrire un script qui permet d'afficher la somme des carrés des 20 premiers entiers naturels.
3. Écrire un script qui définit deux variables `x` et `y` et affiche la plus grande des deux.
4. Écrire un script qui définit trois variables `a` (supposée non nulle), `b` et `c` et qui affiche le nombre de racines réelles de $ax^2 + bx + c$ et les racines le cas échéant.

■