Les structures dans Python :

Il existe principalement deux types de structures dans Python, elles sont quasi indispensable pour écrire un code complxe. Il y a les conditions, qui permettent d'effectuer certaines actions seulement dans certains cas, et les boucle qui vous permettent de répéter une certaine opération autant de fois que nécessaire.

Les structures conditionnelles :

Sans les structures conditionnelles les programmes seraient bien pauvre. On remarque que très vite on a besoin d'effectuer une action dans un certain cas, et une autre dans un autre cas. Les structures conditionnelles sont présentes dans la quasi totalité des programmes. La structure conditionnelle la plus simple se compose uniquement du mot clé `if` suivi d'une condition contenant un opérateur de comparaison et finalement d'un `:`. Il en existe plusieurs qui ont tous une utilité différente. Ils signifient respectivement :

- == est égal
- =! est différent
- < est plus petit que
- > est plus grand que
- <= est plus petit ou égal
- >= est plus grand ou égal

Une sructure conditionnelle basic s'écrit de la mainère suivante :

In [1]:

```
x = 4
if (x < 7):
    print("x est plus petit que 7.")
```

```
x est plus petit que 7.
```

Ceci est une structure conditionnelle de base, elle vérifie que la valeur de `x` soit inférieure à 7 et écrit une pharse dans ce cas. On peut y ajouter une seconde parite à la fin de la structure en ajoutant le mot clé `else` aussi suivit d'un `:` mais sans condition. Ce mot clé peut se traduire en français par sinon. Cela veut dire que cette partie ne sera exécutée que si et seulement si la ou les conditions au-dessus ne sont pas vérifiées. Exemple de condition plus complexe :

In [2]:

```
y = 6
if (y == 5):
    print("x est égal à 5.")
else:
    print("x est différent de 5.")
```

```
x est différent de 5.
```

Dans l'exemple ci-dessus, la condition vérifie que `x` soit égal à 5 et écrit la pharse "x est égal à 5", mais si ce n'est pas la cas, comme dans notre exemple, il écrit "x est différent de 5".

De plus on peut faire une série de condition sans utiliser plusieurs fois le mot clé `if`. Il faut commencer la première condition en utilisant le `if` mais pour les condition suivante, on peut utilisé un autre mot clé, `elif`, qui signifie "sinon si". Ce mot clé doit également être suivi d'une condition. Exemple de condition complexe :

In [3]:

```
z = 9
if z < 9:
    print("z est plus petit que 9.")
elif z > 9:
    print("z est plus grand que 9.")
else:
    print("z est égal à 9.")
```

```
z est égal à 9.
```

Dans ce dernier exemple, on vérifie d'abord si z est plus petit que 9, si ce n'est pas le cas, on vérifie que 9 est plus grand que 9, et si ce n'est toujours pas le cas, on est sûr que z vaut 9. On peut ajouter autant de `elif` que l'on veut.

On peut aussi utiliser les structures conditionnelles avec d'autre type de condition. Par exemple, on peut vérifier qu'un élément donné soit dans une liste. Cela se fait à l'aide du mot clé `in`. Il est aussi possible vérifier l'inverse, à savoir qu'un élément ne soit pas dans une liste. Dans ce cas il faut utiliser `not in`. Exemple :

In [4]:

```
L = ["a", "b", "c"]

if "a" in L:
    print("L'élément a appartient à la liste L.")
else:
    print("L'element a n'appartient pas à la liste L.")
```

```
L'élément a appartient à la liste L.
```

Cette méthode peut aussi s'utiliser avec des dictionnaires `{}` ou des tuples `()`.

In [5]:

```
T = (1, 2, 3)
if 3 not in T:
    print("L'élément 3 n'appartient pas à T.")
else:
    print("L'element 3 appartient à T.")
```

```
L'élement 3 appartient à T.
```

Ici se termine la partie consacrée aux conditions, nous allons maintenant passer aux boucles.

Les boucles :

Rappelons nous que les boucles permettent de répéter une certaine opération autant de fois que nécessaire. Il en existe deux types reliées à deux mots clés spécifiques, `while` et `for`.

Les boucles while :

Commençons par les boucles while. Ces boucle se forme avec le mot clé while suivit d'une condition et d'un `:`. Tant que la condition donnera le résultat True, la boucle recommencera les instructions présentes à l'intérieure. Il faut impérativement que la boucle ait un fin, c'est à dire que la condition donne False a un moment donné. Sinon le script python restera bloqué à cette étape et renverra une erreur. Un moyen de s'en assuré est de mettre une variable dans la condition, et de l'incrémenter (`+=`) à chaque tour de boucle. Il ne faut pas oublié d'initialiser la-dite variable avant la boucle. Voici un exemple de boucle while simple :

In [6]:

```
i = 0
while i < 9:
    print(i)
    i += 1
```

```
0
1
2
3
4
5
6
7
8
```

Dans cet exemple, tant que la variable `i` est strictement plus petite que 9, on va d'abord l'afficher, puis l'incrémenter de 1.

Les boucles for :

L'autre type de boucle, les boucles for sont un peu plus complexe mais elles ont un plus grand domaine d'application. Elles s'utilise avec le mot clé `for` et le mot clé `in`. L'instruction `for` s'utilise sur des séquences. Elle est en fait spécialisée dans le parcours d'une séquence de plusieurs données. Par exemple sur des chaines de caractère, sur des listes, des tuples, ... Elle se construit de la manière suivante :

In [7]:

```
chaine = "Bonjour"
for lettre in chaine:
    print(lettre)
```

```
B
o
n
j
o
u
r
```

Ce code ci-dessus va faire un nombre de tour de boucle en fonction du nombre de caractères contenus dans la `chaine`. On a accès à la lettre à laquelle le tour de boucle correspond en utilisant la même variable. Un autre exemple avec une liste où on fait la somme de tous les éléments :

In [8]:

```
L = [1, 2, 5, 8, 13, 18]
somme = 0

for element in L:
    somme += element

print(somme)
```

```
47
```

Une autre strucutre de boucle est faisable à l'aide de la fonction `range()` qui prend un ou deux paramètres sous forme de nombres. Si deux nombre sont donnés, la variable associée prendra successivement les valeurs entières de l'interval des deux nombres ne comprenant pas le paramètre le plus grand. Si au contraire, un seul paramètre est donné, la variable prendra les valeurs entières comprises entre 0 et l'entier précédant ce nombre. Cette fonction permet tout comme les boucles while de faire un certain nombre de tour de boucle. Elle s'utilise de la manière suivante :

In [9]:

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

L'exemple ci-dessus, va simplement afficher les nombres de 0 à 4. La variable i va simplement prendre successivement les entiers 0, 1, 2, 3 et 4, et la boucle s'arrêtera après.

Nous arrivons à la fin de la parie consacrée aux boucles. J'ajouterais simplement que les deux types de structures, les conditions et les boucles, peuvent tout à fait être combinée ensemble. Il y a aussi deux autres mots clés qui peuvent se rendre très utiles. Il s'agit de `break` et `continue`. `Break` permet d'interrompre complètement une boucle à un moment donné, ce qui suit ce sera pas exectué. Le mot-clé `continue`, quant à lui permet de revenir directement à la première ligne de la boucle pour le prochain tour sans terminer les instructions. Exemple :

sans terminer les instructions. Exemple :

In [10]:

```
k = 1

while k < 15:
    if k % 5 == 0:
        k += 2
        print("On incrémente k de 2. k est maintenant égale à ", k)
        continue # On retourne au while sans exécuter lignes suivantes
    print("La variable k =", k)
    k += 1
```

```
La variable k = 1
La variable k = 2
La variable k = 3
La variable k = 4
On incrémente k de 2. k est maintenant égale à  7
La variable k = 7
La variable k = 8
La variable k = 9
On incrémente k de 2. k est maintenant égale à  12
La variable k = 12
La variable k = 13
La variable k = 14
```