Syntaxe Python : fiche récapitulative

Les instructions

Chaque ligne d'un programme python représente en général une et une seule instruction.
Le sens de lecture de ces instructions est très simple : pour savoir ce que fait un programme, il suffit de lire chaque ligne dans l'ordre, de bas en haut.

Il est possible de faire appel à des instructions de base (déclaration de variable, opérations élémentaires, affichage, boucle, etc...), mais aussi d'utiliser des instructions pré-programmées.

L'affichage de texte (print())

Une instruction qu'il est particulièrement utile de connaître est l'instruction print(). Il s'agit de l'instruction permettant d'afficher une information à l'écran.

```
print("Bonjour")
```

affiche à l'écran :

```
Bonjour
```

Attention : chaque instruction print() entraine automatiquement un retour à la ligne lors de l'affichage. Pour éviter un retour à la ligne, il est possible de préciser une valeur qui remplace ce retour à la ligne :

```
print("Bonjour", end=" ")
print("Comment ça va ?")
```

Affiche :

```
Bonjour Comment ça va ?
```

L'affichage se produit en une seule ligne, car nous avons remplacé le retour à la ligne après Bonjour par un espace.

Cette instruction print() permet aussi d'afficher la valeur de calculs (cf. "Les opérations") et de variables (cf. "Les variables").

Les boucles bornées (for)

Il est parfois nécessaire de répéter une même instruction (ou ligne de code) plusieurs fois. Afin d'éviter d'avoir à écrire cette ligne de code autant de fois qu'il est nécessaire de l'exécuter, on peut faire appel aux boucles, qui permettent de répéter une instruction un nombre de fois déterminé.

Si, par exemple, nous avons besoin d'écrire 5 fois "Bonjour" sur l'écran, au lieu d'écrire :

```
print("Bonjour")
print("Bonjour")
print("Bonjour")
print("Bonjour")
print("Bonjour")
```

Il est possible d'utiliser une boucle "for" :

```
for loop in range(5):
    print("Bonjour")
```

Dans les deux cas, "Bonjour" sera affiché 5 fois à l'écran.
La méthode des boucles présente plusieurs avantages :

- Elle est plus facile à lire (on sait automatiquement combien de fois on va faire le tour de la boucle)
- Elle est plus facile à rédiger (on ne risque pas d'oublier une instruction par erreur à force de copier-coller)
- Elle est plus facile à modifier (si l'on souhaite finalement écrire 1000 fois "Bonjour", il suffit de changer un nombre).

Syntaxe de la boucle "for"

On peut décomposer la ligne de code d'une boucle for en les éléments suivants :

- **for** : indique que l'on démarre une boucle
- **loop** : nom d'une variable propre à la boucle, que l'on peut manipuler (il est possible de donner un autre nom que loop)
- **in range(nombre):** : permet de signaler le nombre de tours de boucle que l'on souhaite réaliser. Si l'on écrit range(10), la boucle s'exécutera 10 fois. A chaque passage de boucle, notre variable (ici, "loop"), changera de valeur jusqu'à arriver au nombre que l'on a fixé : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Boucles "for" et indentation

Une boucle for permet de répéter une ou plusieurs instructions (on parle alors de "bloc d'instructions") plusieurs fois. Afin de savoir quelles sont les instructions que l'on souhaite répéter plusieurs fois, il est nécessaire de l'indiquer à la machine en respectant l'indentation.
L'indentation correspond à l'espace que l'on laisse au début de chaque ligne de code.

```
for loop in range(5):  # Cette ligne de code touche le bord
    print("Je suis indenté")  # Cette ligne de code est décalée d'une tabulation
                              # ce qui nous permet de savoir qu'elle est
                              # "à l'intérieur" de notre boucle for et sera donc
                              # répétée plusieurs fois
```

Une indentation correspond à une tabulation (la touche tab est généralement un cran au-dessus de la touche VERR. MAJ sur un clavier).

L'indentation est le seul moyen de savoir, pour la machine comme pour le développeur, comment le programme est supposé se comporter.

```
for loop in range(5):
    for loop in range(2):
        print("Bonjour")  # Cette instruction est à l'intérieur d'une boucle for
                          # qui est elle-même à l'intérieur d'une boucle for
```

Dans l'exemple ci-dessus, la boucle for de valeur 2 est à l'intérieur de la boucle for de valeur 5. Cela signifie que que la boucle for de valeur 2 s'exécutera 5 fois d'affilée. Par conséquent, l'instruction `print("Bonjour")` s'exécutera 5x2 fois, c'est-à-dire 10 fois.

```
for i in range(3):
    print("Je suis dans la boucle !")
    print("Et moi aussi !")
print("Mais moi je n'y suis pas.")
```

L'exemple ci-dessus donnerait le résultat suivant :

```
Je suis dans la boucle !
Et moi aussi !
Je suis dans la boucle !
Et moi aussi !
Je suis dans la boucle !
Et moi aussi !
Mais moi je n'y suis pas.
```

Les opérations

Il est possible de réaliser, dans Python, toutes les opérations élémentaires mathématiques :

```
print(5 + 1)
print(4 - 2)
print(3 * 6)
print(10 / 2)
```

Dans le programme ci-dessus, on affiche les valeurs résultant des opérations 5 + 1, 4 - 2, 3 * 6 et 10 / 2.
Le programme affiche :

```
6
2
18
5
```

Les variables

Les variables sont le moyen qu'utilise un programme pour se souvenir de valeurs qui sont utilisées dans notre code.
Elles sont indispensables pour tout programme d'une certaine taille. Par exemple, elles peuvent être utilisées au sein d'un jeu vidéo pour retenir des informations qui peuvent changer au cours du temps, comme le temps restant dans un niveau, les points de vie ou le score d'un joueur, etc...

Déclarer une variable

Pour pouvoir utiliser une variable, il faut d'abord la déclarer en lui donnant un nom. En effet, pour préciser notre intention, il faut indiquer à la machine que nous allons utiliser une variable et que lorsque nous utiliserons un nom précis, il faudra utiliser une valeur spécifique retenue en mémoire.

```
maVariable = 5
```

Le code ci-dessus n'affiche rien. Il permet cependant de dire à la machine de faire une place dans sa mémoire pour garder la valeur 5 (on parle d'**initialisation**). Lorsque dans notre programme, on écrira maVariable, la machine saura où elle doit chercher en mémoire pour récupérer une valeur. Pour l'instant, cette valeur vaut 5, mais elle est susceptible de changer (c'est le but !) au cours de l'exécution du programme.

Il est possible de manipuler ces variables en réalisant des opérations élémentaires dessus ou en les utilisant comme valeurs d'entrée de certaines instructions.

```
nombreDeTours = 3  # On initialise la variable nombreDeTours à 3
nombreDeTours = nombreDeTours + 2  # On augmente de 2 la valeur de la variable
for loop in range(nombreDeTours):  # On peut utiliser notre variable au sein
                                   # d'un for
    print("Je fais un tour de boucle !")
```

Dans l'exemple ci-dessus, la variable nombreDeTours vaut 3 lors de la première ligne. On y rajoute 2 dans la ligne suivante : nombreDeTours vaut maintenant 5.
Dans la troisième ligne, on utilise la variable nombreDeTours pour déterminer le nombre de tours que l'on fera dans notre boucle for, c'est à dire 5, la valeur de nombreDeTours.

Nommer une variable

Une variable peut porter le nom que l'on souhaite : a, b, toto, nombreDeTours, nombrePoints, etc... L'important est de donner un nom qui soit cohérent avec l'utilité de la variable.
Si une variable représente quelque chose de spécifique, comme un nombre de points de vie, on essaye de lui donner un nom reflétant cette fonction. On pourrait par exemple appeler une telle variable `pointsVie`.

Il est important de respecter certaines règles concernant le nommage de variables :

- Toujours donner un nom qui a du sens
- Ne jamais commencer un nom de variable par un chiffre (cela ne fonctionne pas)
- Respecter une certaine nomenclature, comme par exemple écrire tout en minuscules, sauf la première lettre de chaque mot suivant le premier (comme dans nombreDeTours). On appelle cette notation "camelCase"
- Eviter les signes de ponctuation
- Ne pas donner un nom de variable qui correspond à un mot-clé réservé par Python (il est impossible, par exemple, de nommer une variable "for". Sinon, comment faire la différence avec une boucle ?)

Utilisation des variables

Les variables peuvent se déclarer, s'initialiser et être manipulées à tout moment dans notre programme. On peut les utiliser au sein de blocs d'instructions, à l'intérieur de boucles for, etc... Il s'agit réellement de l'outil de base du développeur.

```
maValeur = 1  # On initialise maValeur à 1
for loop in range(10):  # On rentre dans une boucle de taille 10
    print(maValeur)  # On affiche la valeur actuelle de maValeur
    maValeur = maValeur + 1  # On augmente la valeur de maValeur de 1 avant de
                             # recommencer un tour de boucle
```

Ce programme affiche :

```
1
2
3
4
5
6
7
8
9
10
```