Le JEU pour REVISER les bases du PYTHON

Récapitulatif des cours présents dans le jeu • **Chapitre 3**

Table des matières

Les boucles

Les **boucles** sont l'une des notions les plus importantes en programmation.

Elles sont utiles pour créer une répétition.

Il en existe **deux sortes** :

- Les boucles **bornées** : On sait à l'avance le **nombre d'itération** nécessaire.
- Les boucles **non bornées** : On ne sait **pas** à l'avance le **nombre d'itération** nécessaire.

1. Les boucles bornées : for

Les boucles **bornées** sont aussi appelées des boucles **for**.

On les dit « bornée » puisque l'on connaît le nombre d'itérations nécessaires pour effectuer la boucle.

Exemples :

- La boucle doit aller de 6 à 52 inclus avec un pas de 1
  (*6, 7, 8, 9, … , 51, 52 ; soit 47 itérations de la boucle)*.
- La boucle doit aller de 0 à 11 inclus avec un pas de 2
  (0, 2, 4, 6, 8, 10 ; soit 6 itérations de la boucle).

A. La structure

La structure Python de la boucle **for** est la suivante :

```
for variable in bornes :
    # les instructions de la boucle
    # Elles sont décalées avec une tabulation.
```

La **variable** de la boucle sera égale à un élément différent à chaque tour de boucle. La variable n'a pas besoin d'être définie à l'avance. Elle est souvent nommée « i » mais elle peut avoir n'importe quel nom (voire cours sur les variables).

La **borne** varie en fonction de l'utilisation. C'est elle qui détermine les éléments qui seront un à un attribués à la **variable**.

La borne peut être :

range(y) ou range(x, y) ou range(x, y, z)	La fonction **range** est utilisée par les boucles for. Elle a d'autres utilisation, mais on ne va pas rentrer dans les détails ici. Elle a 3 principaux paramètres (des entiers) : - **x** est la valeur de départ *(6 ou 0 d'après les exemples précédents)* - **y** est la valeur d'arrivée non-inclus *(53 ou 12 d'après les mêmes exemples)* - **z** est le pas *(1 ou 2 d'après les mêmes exemples)* Par défaut, **x** est égal à **0** et **z** est égal à **1**.
Structures de données	A la place de la fonction « range », il est possible de placer une **liste**, un **tuple** ou un **dictionnaire**.
Fonctions	Il est aussi possible de mettre une autre fonction que « range ». Par exemple, une fonction qui retourne une liste.

B. Les différentes utilisations

Voici quelques exemples d'utilisations : avec **range** ou avec une **liste**, un **tuple** ou un **dictionnaire**.

Range

```
for i in range(5):              # Affiche 0  1  2  3  4
    print(i)

for i in range(0, 5):           # Affiche 0  1  2  3  4
    print(i)

for i in range(3, 12):          # Affiche 3  4  5  ...  9  10  11
    print(i)

for i in range(3, 12, 2):       # Affiche 3  5  7  9  11
    print(i)

for i in range(11, 2, -2):      # Affiche 11  9  7  5  3
    print(i)
```

Boucles avec une liste (fonctionne aussi avec un tuple et une chaîne de caractères)

```
tab = ["a", "b", 5, 6, [True, 3]]

for val in tab:                 # Affiche "a"  "b"  5  6  [True, 3]
    print(val)
```

Boucles avec un dictionnaire

```
dico = {"a": 1, "b": 5, "z": 9}

for cle in dico:                # Affiche "a"  "b"  "z"
    print(cle)

for cle in dico.keys():         # Affiche "a"  "b"  "z"
    print(cle)

for val in dico.values():       # Affiche 1  5  9
    print(val)

for cle, val in dico.items():
    print("La cle :", cle, "et la valeur :", val)

    # Affiche "La cle : a et la valeur : 1"
              "La cle : b et la valeur : 5"
              "La cle : z et la valeur : 9"

for it in dico.items():
    cle = it[0]
    val = it[1]
    print("La cle :", cle, "et la valeur :", val)

    # Affiche "La cle : a et la valeur : 1"
              "La cle : b et la valeur : 5"
              "La cle : z et la valeur : 9"
```

Boucles avec range et une liste (fonctionne aussi avec un tuple et une chaîne de caractères)

```
tab = ["a", "b", 5, 6, [True, 3]]


for i in range(len(tab)):     # Affiche 0  1  2  3  4
    print(i)


for i in range(len(tab)):     # Affiche "a"  "b"  5  6  [True, 3]
    print(tab[i])
```

2. Les boucles non-bornées

Les boucles **non-bornées** sont aussi appelées des boucles **while**.

On les dit « non-bornée » puisque l'on **ne** connaît **pas** le nombre d'itérations nécessaires pour effectuer la boucle.

Exemples :

- La boucle doit s'arrêter lorsque la variable x est égale à 13
- La boucle doit s'arrêter lorsque la variable num est inférieure à 0 ou lorsque la variable tab est différente de "nsi".

A. La structure

La structure Python de la boucle **while** est la suivante :

```
while condition(s) :
    # les instructions de la boucle
    # Elles sont décalées avec une tabulation.
```

La ou les **condition(s)** de la boucle donne un résultat **booléen** (voire le Chapitre 2 sur les conditions).

Si le résultat **booléen** des conditions de la boucle est :

- **True** : la boucle continue
- **False** : la boucle s'arrête

B. Les différentes utilisations

Voici quelques exemples d'utilisations.

Conditions

```
i = 0

while i < 5:                # Affiche  1  2  3  4  5
    i = i + 1
    print(i)                # Arrêt si i supérieure ou égale à 5


a = 6

while a != 3:               # Affiche  6  5  4
    print(a)
    a = a - 1               # Arrêt si a égale à 3


a = 13
i = 2

while a > 3 and i < 5:      # Affiche  11  3
    a = a - 2                            9  4
    i = i + 1                            7  5
    print(a, i)             # Arrêt si a est =< à 3 ou si i => 5


a = 13
i = 2

while a > 3 or i < 5:       # Affiche  11  3
    a = a - 2                            9  4
    i = i + 1                            7  5
    print(a, i)                          5  6
                                         3  7

                            # Arrêt si a est =< à 3 et si i => 5
```

Parcours d'entiers (comparaison avec la boucle for)

```
for i in range(5):                  # Affiche 0  1  2  3  4
    print(i)


i = 0
while i < 5:                        # Affiche 0  1  2  3  4
    print(i)
    i = i + 1
```

Parcours d'une liste (comparaison avec la boucle for)

```
tab = ["a", "b", 5, 6, [True, 3]]


for i in range(len(tab)):    # Affiche "a"  "b"  5  6  [True, 3]
    print(tab[i])


i = 0
while i < len(tab):          # Affiche "a"  "b"  5  6  [True, 3]
    print(tab[i])
    i = i + 1
```

/!\ VIGILENCE

Les **valeurs de variables** présentes dans la (ou les) condition(s) doivent être **modifiées** dans les instructions de la boucle afin d'atteindre un résultat **False**.

Si ce n'est pas le cas, le résultat ne passera jamais à **False** et la boucle ne **finira jamais**.

Exemple :

```
i = 10
while i > 5:                 # Affiche 10  11  12  13 ...
    print(i)
    i = i + 1                # i ne sera jamais =< à 5.
```

3. En savoir plus sur les boucles

- https://www.data-transitionnumerique.com/boucle-for-python/
- https://courspython.com/boucles.html#boucle-while
- https://www.pierre-giraud.com/python-apprendre-programmer-cours/boucle-for-while/
- Ou demande à ton/ta prof de NSI :)