Python 2 – Boucles bornées en Python : for

La boucle POUR est dite **bornée** car on peut connaître à l'avance le nombre de fois où elle va se dérouler.

La syntaxe Python permettant de réaliser plusieurs fois la même chose est la suivante :

```
01 print("depart")
02 for _ in range(4):
03     print("bonjour")
04 print("fin")
```

Résultat à l'écran :

L'interpréteur Python exécute les lignes dans cet ordre :

_____

Mais on peut faire mieux : on peut réaliser utiliser la valeur de la variable de boucle qui varie à chaque tour de boucle pour réaliser presque pareil mais pas exactement la même chose.

En notant **range(4)**, la variable de boucle **x** commence à la valeur _____ et finit par _____.

```
01 print("depart")
02 for x in range(4):
03     print(x)
04 print("fin")
```

Résultat à l'écran :

```
01 print("depart")
02 for x in range(4):
03     print(x * "o")
04 print("fin")
```

Résultat à l'écran :

On peut alors utiliser cette variable pour faire des calculs progressifs, comme la somme des entiers de 1 à 100 par exemple. On va réaliser 0 + 1 + 2 + 3 + 4 + 5 + 6 + ... + 99 + 100.

```
01 s = 0
02 for x in range(101):
03     s = s + x
04 print(s)
```

L 01 Somme s au début :
L 02-03 Somme s après le tour de boucle avec x = 0 :
L 02-03 Somme s après le tour de boucle avec x = 1 :
L 02-03 Somme s après le tour de boucle avec x = 2 :
L 02-03 Somme s après le tour de boucle avec x = 3 :
L 02-03 Somme s après le tour de boucle avec x = 4 :
...

Python 3 – Instructions conditionnelles

Les instructions conditionnelles permettent de modifier l'ordre séquentiel d'exécution en exécutant un bloc (ou pas) en fonction d'une condition fournie sous forme d'une expression booléenne.

On teste d'abord une première condition qu'on place derrière le mot-clé **if** qui veut dire **SI...**
Si cette expression est évaluée à **True**, on exécute alors le bloc indenté sous le **if.**
Si cette expression est évaluée à **False**, on regarde s'il existe un **elif** ou un **else**.

Le mot-clé **elif** veut dire **SINON, SI...** : si le bloc du dessous n'a pas été validé, on évalue son expression.
Si cette expression est évaluée à **True**, on exécute alors le bloc indenté sous le **elif.**
Si cette expression est évaluée à **False**, on regarde s'il existe un **elif** ou un **else**.

Le mot-clé **else** veut dire **SINON** : si le bloc du dessous n'a pas été validé, on réalise ce bloc : aucune condition.

```
01 note = 15
02 if note >= 17:
03     app = "très bien"
04 elif note >= 13:
05     app = "bien"
06 elif note == 0:
07     app = "rien..."
08 elif note < 9:
09     app = "insuffisant"
10 else:
11     app = "moyen"
12 print("fin")
```

Les lignes exécutées si note contient 15 sont : L 01 - _____

Les lignes exécutées si note contient 19 sont : L 01 - _____

Les lignes exécutées si note contient 03 sont : L 01 - _____

Les lignes exécutées si note contient 10 sont : L 01 - _____

Point important 1 : on notera qu'on exécute que l'un des blocs if-elif-else. On sort de la structure dès que l'un des blocs a été effectué.

Point important 2 : on peut placer plusieurs instructions par bloc, le tout est de les indenter.

Point important 3 : il peut y avoir 0, 1, 2 ou plus de blocs **elif** à la suite du **if.**

Point important 4 : il peut y avoir 0 ou 1bloc **else** et il doit être le dernier de la structure**.**

Python 4 – Boucles non bornées en Python : while

Cette boucle permet de boucle sans savoir combien de fois on va le faire à l'avance. C'est pour cela qu'on la nomme **boucle non bornée**. Le principe est simple :

→ On évalue l'expression booléenne qui constitue la **condition de poursuite** de la boucle.
→ Si cette expression est évaluée à **True**, on réalise la boucle puis on évalue à nouveau la condition de poursuite.
→ Si cette expression est évaluée à **False**, on sort de la boucle.

Exemple : Peut-on trouver deux entiers successifs tels que $(x+1)^2 - x^2$ soit égal à 9 ? Donner les lignes d'exécution sachant que x = 4 valide la demande.

Existe-t-il deux nombres pour une différence de 99 ?
Que constate-t-on si on cherche une différence de 100 ?

```
01  x = 0
02
03  while ((x+1)**2 - x**2) != 9:
04      x = x + 1
05      print("Test avec x valant", x)
06
07 print("Valeurs obtenues : ", x+1, " ", x)
```

Exemple 2 : on veut lancer 2 dés à 6 faces tant qu'on n'obtient pas un double (2 dés identiques). On veut récupérer le nombre de fois où on a dû tirer les 2 dés.

Donner les lignes exécutées en supposant qu'on tire 1-3 puis 5-4 puis 1-6 puis 6-6.

```
01 from random import randint
02
03  de1 = 1
04  de2 = 0
05  nombre = 0
06
07  while de1 != de2:
08      nombre = nombre + 1
09      de1 = randint(1, 6)
10      de2 = randint(1, 6)
11      print(de1, " et ", de2)
12
13  print("Nombre de lancés :")
14  print(nombre)
```

Exemple 3 : vous avez 500 euros sur un compte qui rapporte 10 % par an. Combien d'années faut-il laisser l'argent en banque pour parvenir à dépasser à 1000 euros ?

```
01  argent = 500
02  duree = 0
03
04  while argent < 1000:
05      duree = duree + 1
06      argent = argent * 1.10
07
08  print("Argent au final :")
09  print(argent)
10  print("Durée en années :")
11  print(duree)
```