# Spécialité NSI - Année 2023 - 2024

## La boucle FOR (pour)

Une boucle "pour" ou "for" sert à répéter une action. Rappelons brièvement la principale différence entre une boucle "pour" et une boucle "tant que" (dont nous donnerons la syntaxe à un autre moment) :

- Avec une boucle "pour", on connaît à l'avance le nombre de répétitions de l'action à exécuter.
- Avec une boucle "tant que", le nombre de répétitions est conditionné par l'évolution de l'état de certaines variables du programme.

### La table de multiplication du 8

```
Pour k allant de 1 à 10 :
    afficher k*8
```

```
for k in range(1,11) :
    print(k, "* 8 = ",k*8)
```

### Indentation en Python

L'**indentation** après la ligne for ... : est obligatoire (et automatique avec un éditeur adapté) : elle permet de délimiter la portée du for.

De plus, les deux point ":" sont obligatoires.

### Fonction range()

a et b désignant deux entiers, range(a,b) est l'ensemble (ordonné) des entiers a, a+1, a+2, ..., b-1. C'est à dire l'ensemble des entiers k vérifiant a≤ k <b.

On fera attention à l'apparente dissymétrie entre les deux bornes.

### Résultat du programme

```
for k in range(1,11) :
    print(k, "* 8 = ",k*8)
```

```
1 * 8 =  8
2 * 8 =  16
3 * 8 =  24
4 * 8 =  32
5 * 8 =  40
6 * 8 =  48
7 * 8 =  56
```

# Les boucles avec pas

## Boucle avec pas positif

**Comment afficher les entiers pairs entre 2 et 20 ?** Le plus simple est d'utiliser range(a,b,p) qui permet de parcourir les valeurs supérieures ou égales à a et strictement inférieures à b de p en p : a, a+p, a+2p, a+3p, ...

```
1 for k in range(2,21,2) :
2   print(k)
```

On obtient donc :

```
1 2
2 4
3 6
4 8
5 10
6 12
7 14
8 16
9 18
10 20
```

## Boucle avec pas négatif

**Comment afficher les entiers en ordre décroissant ?** Le plus simple est d'utiliser range(a,b,p) avec une valeur négative du pas p.

```
1 for i in range(15,0,-1) :
2   print(i)
```

On obtient :

```
1 15
2 14
3 13
4 12
5 11
6 10
7 9
8 8
9 7
10 6
11 5
12 4
13 3
14 2
15 1
```

# Les erreurs à éviter

## L'indentation

```
1 for i in range(5):
2     # Ce code est dans la boucle FOR et sera exécuté 5 fois
3     print(i)
4 # Ce code n'est pas dans la boucle FOR et sera exécuté 1 fois
```

On ne le répétera jamais assez, mais en Python, votre code doit être indenté.

Cela signifie que lorsqu'un morceau de code rentre dans une instruction (ici une boucle FOR), vous devez **décaler ce morceau du bord gauche**. En pratique, on le décale en utilisant 4 espaces ou une tabulation (via la touche Tab). C'est à partir d'aujourd'hui que cette touche va vous être très utile, alors évitez de la démonter de vos claviers.

Et cette touche, en plus de décaler le texte vers la droite, peut aussi le décaler vers la gauche grâce au raccourci Shift+Tab !

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

Ainsi, le code ci-dessus est erroné car aucune instruction ne se trouve dans la boucle FOR. Python va donc planter et renvoyer l'erreur `IndentationError: expected an indented block after 'for' statement on line 1`

## Choix de la variable

```
1 i = 3
2 for j in range(8):
3     for k in range(3):
4         print(i, j, k)
```

Lorsque l'on crée une boucle FOR, il est très important de réfléchir au nom de la variable de boucle.

Une erreur classique consiste à utiliser le même nom qu'une variable déjà existante. Cela aura pour effet de **supprimer l'ancienne valeur** associée à la variable déjà utilisée, et **peut causer un bug** dans le programme.

```
1 i = 2
2 x = 1
3 for i in range(10):
4     x = x + 1
5 print(i) # 9
```

Alors que `i` valait 3, le code ci-dessus a changé la valeur de `i` alors que le développeur ne le souhaitait pas forcément. Il aurait été judicieux d'appeler sa variable de boucle `j` par exemple (ligne 3).

## Initialiser ses variables

```
somme = 0
for i in range(5):
    somme = somme + i
print(somme)
```

Comme on peut le voir, ce programme ci-dessus affiche la somme des nombres de 0 à 4.

```
for i in range(5):
    somme = somme + i
    #       ^^^^^ Variable non initialisée
print(somme)
```

Cette version est quasiment indentique, à part le fait que la variable `somme` n'a pas été initialisée. Même si pour nous le 0 semble être la valeur par défaut, Python ne peut pas le deviner et vous renverra l'erreur suivante :
`NameError: name 'somme' is not defined`

Il est donc très important d'initialiser les variables avant de les récupérer pour faire des calculs.

## Modifier la variable de boucle

```
for i in range(31):
    i = 2 * i
    print(i)
```

Contrairement à ce que l'on pourrait penser, le code ci-dessus ne va pas afficher 0, 2, 6, 14, 30 mais va afficher les nombres pairs de 0 à 60.

En effet, malgré la modification de i dans la boucle, rien ne subsiste lors des répétitions suivantes. Il est d'usage de **ne pas modifier la variable de boucle** afin d'éviter toute confusion lors de la lecture du programme.