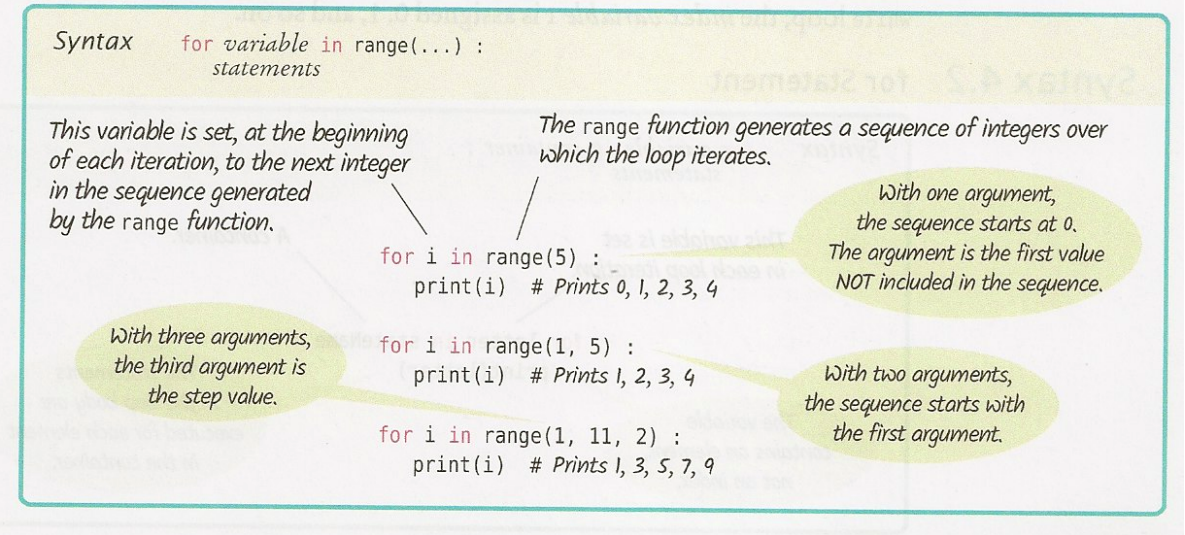
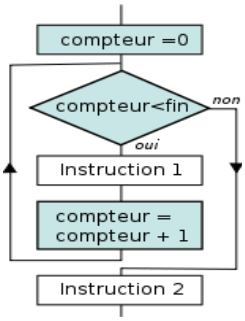
Informatique Pour Tous

Cours 4 – boucle FOR



A / C'est quoi une boucle `for` ?

La boucle `for` est en informatique une **structure de contrôle** de programmation permettant de réaliser une boucle associée à une **variable entière** qui sera **incrémentée** à chaque itération.
L'intérêt du `for` est d'épargner un peu de fatigue au programmeur, en lui évitant de gérer lui-même la progression de la variable qui lui sert de compteur (on parle d'incrémentation). On l'utilise lorsque l'on connaît le nombre d'itération dans une boucle. (D'après Wikipédia.)



Comment obtenir cet affichage ?

```
Allo ?
Allo ?
Allo ?
Non mais Allo quoi !
```

Voici le code correspondant.

```
for i in range (10,13):
    print("Allo ?")

print("Non mais Allo quoi !")
```

B / La boucle `for` dans Python

B. 1 – Ecrire un programme utilisant une boucle `for`

1 - On détermine combien de fois la boucle devra s'exécuter, ce nombre étant en général exprimé en fonction d'une ou plusieurs variables du programme.
2 – On choisit une variable pour le compteur et on identifie si elle doit jouer un rôle dans le corps de la boucle.
3 – On écrit le corps de la boucle.
4 – Comme pour la boucle `while`, il peut être nécessaire de prévoir une initialisation des variables en amont de la boucle et un post-traitement en aval.

B. 2 – Utilisation de `range()`

`Range` permet de générer une liste d'entiers croissants successifs qui peuvent être utilisés comme **index dans une itération**. `Range` peut avoir **de un à trois arguments**.

1 – Si il n'y a qu'un argument, il représente le nombre d'éléments de la liste partant **depuis zéro** et incrémenté de un à chaque fois.

```
for i in range (10):
    print(i, end=",")
```

```
0,1,2,3,4,5,6,7,8,9,
```

Note : l'instruction `end` est intéressante : `print(x,end="  ")`. Elle permet de séparer d'un espace les résultats, sinon, ils sont affichés en colonne.

2 – Si `range` est appelé avec **deux arguments**, il s'agit de la **borne de départ de liste et de celle d'arrivée non comprise.** L'incrément entre deux éléments est de un.

```
for i in range (10, 20):
    print(i, end=",")
```

```
10,11,12,13,14,15,16,17,18,19,
```

3 – Quand `range` comporte trois arguments, il s'agit de la borne de départ de la liste, celle d'arrivée non comprise et **du pas d'incrémentation**.

```
for i in range (10, 20, 3):
    print(i, end=",")
```

```
 10,13,16,19,
```

C / Quelques exemples

Exemple 1 : puissance d'un nombre

Comment calculer 2^8 avec seulement des multiplications ?

```
r=2

for i in range (0,7):
    print(i, end=",")
    r=r*2

print()
print("Le resultat de 2 puissance 8 est:",r)
```

```
0,1,2,3,4,5,6,
Le resultat de 2 puissance 8 est: 256
```

Exemple 2 : puissance d'un nombre

Comment calculer $8!$ avec seulement des multiplications ?

```
r=1

for i in range (2,9):
    print(i, end=",")
    r=r*i
print()
print("Le resultat de 8! est:",r)
```

```
2,3,4,5,6,7,8,
Le resultat de 8! est: 40320
```

D / Initialisation des variables avant la boucle

Tant que l'on n'a pas affecté une valeur à une variable, celle-ci est absente de l'état. Il faut donc s'assurer qu'à chaque fois qu'une variable est utilisée dans une expression, elle est déjà présente dans l'état, sous peine d'une erreur. Pour cela :

1– On identifie la première ligne du programme où la variable est utilisée. Lorsque l'algorithme comporte des instructions conditionnelles, il peut y avoir plusieurs lignes pour la même variable, en fonction du résultat du test.

2– On vérifie que cette première ligne est toujours une affectation de cette variable. Elle peut être rendue difficile à repérer parce qu'une instruction `for` la gère ou parce qu'elle est combinée avec une saisie au clavier.

3– Enfin, si une initialisation est manquante, il faut déterminer une valeur d'initialisation cohérente avec la suite du programme et ajouter l'instruction correspondante avant la première utilisation de la variable.

Question:

Que semble faire le programme suivant ? Corrigez le.

```
n=5

for i in range(n):
    if i%2 == 0:
        carre=i**2
    else:
        carre=0
    total=total+carre

print(total)
```

E / Boucle `for` sur une chaîne de caractère

Ici la boucle for est utilisée d'une manière nouvelle, la variable `letter` prend successivement toutes les lettres de la chaîne de caractère `'Python'`. Nous en reparlerons quand nous étudierons les listes et les chaînes de caractères.

```
for lettre in 'Python':
print ('Lettre actuelle :', lettre)
```

```
Lettre actuelle : P
Lettre actuelle : y
Lettre actuelle : t
Lettre actuelle : h
Lettre actuelle : o
Lettre actuelle : n
```