

Labo d'introduction à l'informatique

Yann Thorimbert



**UNIVERSITÉ
DE GENÈVE**

CENTRE UNIVERSITAIRE
D'INFORMATIQUE

Semaine 4.1

Indices de boucle



La semaine passée

Solution du problème du nombre plus grand que 9 :

```
x = [4, 12, -8, ... , 4] # Liste de longueur quelconque !  
for n in x: # On itère sur chaque élément  
    if n > 9:  
        print("Il y a un nombre plus grand que 9 !")
```

Mais comment connaître l'emplacement de ce nombre dans la liste ?



Un autre problème

Comment changer la valeur des éléments au sein de la boucle ?

```
x = [4, 12, -8, 4]
for n in x: # On itère sur chaque élément
    n = n + 2

print(x) # x n'a pas été modifiée
```

Fonctionnement d'une boucle | **Variable de boucle**

- Une boucle for se sert d'une **variable de boucle** dont le nom est choisi par le programmeur.
- Dans notre cas, on voudrait que la valeur de la variable de boucle coïncide avec la valeur des indices des éléments de la liste. On pourrait passer par une liste intermédiaire

```
x = [4, 12, -8, 4]
indices = [0, 1, 2, 3]
for i in indices: # On itère sur chaque élément
    x[i] = x[i] + 2
print(x) # Cette fois-ci, ça fonctionne
```



Fonctionnement d'une boucle | **Variable de boucle**

En Python, la fonction `range` est idéale pour cela, car elle est faite pour itérer sur des entiers entre deux valeurs.

```
x = [4, 12, -8, 4]
n = len(x)
for i in range(n):
    x[i] = x[i] + 2
    print("J'ai traité l'élément d'indice", i)
print(x)
```

On indique le nombre d'éléments à itérer



Fonctionnement d'une boucle | **Variable de boucle**

La fonction range peut prendre 1, 2 ou 3 arguments (cf. futur chapitre sur les fonctions).

```
for i in range(10): #de 0 (par défaut) à 9
    print("Valeur:", i)
```

```
for i in range(2, 10): #de 2 à 9
    print("Valeur2:", i)
```

```
for i in range(2, 10, 3): #de 2 à 9, avec sauts de 3
    print("Valeur:", i)
```

Le "saut" s'appelle l'incrément (il vaut 1 par défaut).



Fonctionnement d'une boucle *for* | **Bloc répété**

- Une boucle *for* permet d'**automatiser la répétition d'instructions** un certain nombre de fois, tout en **gardant trace** de la variable de boucle.
- Par exemple, nous allons automatiser ce code :

```
print("Salut")  
print(2)  
print("Salut")  
print(4)  
print("Salut")  
print(6)  
print("Salut")  
print(8)
```

On voit que les instructions:
`print("Salut")` et `print(x)` sont
répétées, où `x` "saute" de 2 en 2.



Fonctionnement d'une boucle *for* | **Bloc répété**

- Une boucle *for* permet d'**automatiser la répétition d'instructions** un certain nombre de fois, tout en gardant trace de la variable de boucle.
- Solution :

```
for i in range(4):  
    print("Salut")  
    print(2 * i)
```



Terminologie "boucle for"

- Se nomme ainsi parce qu'on itère **pour une variable de boucle** donnée, d'une valeur de départ à une valeur finale.
- Représentez-vous "for i in range(n)" comme "**répéter n fois**, de $i=0$ jusqu'à $i=n-1$ ".
- On verra bientôt la boucle `while`, qui répète un bloc d'instruction tant que certaines conditions sont remplies, plutôt qu'en fonction d'une variable de boucle.