Informatique

8

for if while

Résumé

CC BY NC SA

Dernière mise à jour	Informatique	Denis DEFAUCHY
17/10/2021	8 – for if while	Résumé

Tests dont le résultat est un booléen True ou False	
==	Vérifie l'égalité
!=	Vérifie la différence
>	Comparaison
>=	
<	
<=	
a <= b < c	On peut écrire des inégalités à plusieurs comparaisons
in	'A' in ['A','B'] renvoie True
Opérations entre booléens	
and	Fonction « et »
or	Fonction « ou »
not(Cond)	Donne la condition inverse : Not(True)->False Not(False)->True
Remarque : Lors d'opérations booléennes (and, or), l'ordre des conditions a de l'importance. Si le résultat est forcément induit par ce qui a déjà été évalué, le reste ne sera pas évalué.	
Incrémentation compteur	
i = i + 1 i += 1	Les deux expressions sont équivalentes

Boucle « for »

```
for i in range(n):
    opérations à effectuer
    opérations à effectuer
    opérations à effectuer
    opérations à effectuer
    opérations à effectuer
```

Attention
i varie du séparateur 0 au séparateur n,
soit dans la plage 0, n-1
Il y a bien n étapes

Attention : i est piloté par la boucle, s'il existe avant, il est remplacé, et il est inutile de l'incrémenter :

```
i = 10
for i in range(5):
    print(i)
    i += 5
```

On arrête une boucle for avec la commande « break » :

```
for i in range(5):
    print(i)
    if i==3:
        break
```

Utilisation de for i in L	Attention, dans ce cas, Python va prendre les valeurs de L les unes après les autres à l'aide d'un indice croissant. Si la boucle modifie L, i prend au pas suivant la valeur L[i+1], qui est peut être modifiée si L changée !!!
`for i in range(0):`	Non-exécution du code dans le for
Plusieurs boucles for	Si imbriquées, changer le nom de la variable : i, puis j, puis k Si non imbriquées, garder i à chaque boucle

Dernière mise à jour	Informatique	Denis DEFAUCHY
17/10/2021	8 – for if while	Résumé

Condition « if »

```
if Condition_1 :
    opérations à effectuer si Condition_1 vaut True
    opérations à effectuer si Condition_1 vaut True
elif Condition_2 : # Optionnel
    opérations à effectuer si Condition_2 vaut True
    opérations à effectuer si Condition_2 vaut True
else : # Optionnel
    opérations à effectuer si ni Condition_1, ni Condition_2 ne sont
vérifiées
    opérations à effectuer si ni Condition_1, ni Condition_2 ne sont
vérifiées

# Suite du programme
```

Penser à utiliser « else » même si cela n'est pas nécessaire car sur les programmes sans indentations, il les faut !

Pour ne rien exécuter sans laisser le texte vide (bug) après une condition, on peut écrire « **`continue`** ». Toutefois, préférer créer la condition inverse et ne faire que les opérations à faire !

Boucle conditionnelle « while »

à casser avec « Ctrl+i » - *boucle infinie « **`while`** `True` `:` »*

Le while est utilisé lorsque l'on ne connait pas le nombre d'itérations à réaliser

Sinon privilégier le for

```
Initialisation de la condition
while Condition :
    opérations à effectuer
    opérations à effectuer
    opérations à effectuer
    opérations à effectuer
    opérations à effectuer
    modification de la condition

# Suite du programme
```

Rappel : l'ordre des conditions a de l'importance
La commande « break » permet d'arrêter un while

On pourra vérifier l'indentation en surlignant le texte :

```
1 n = 10
2 Res = 0
3 for i in range(n):
4     Res = Res + i
5 print(Res)
```

Problème ligne 4 (barre blanche = indentation désirée), Res est décalé d'un espace
Problème ligne 5, print est décalé d'un espace

for/else
while/else

Ce qu'il y a dans le else est exécuté à la fin du for/while si aucun break n'a été exécuté

CC BY NC SA