Pense bête 2

Les opérateurs arithmétiques

Symbole	Nom	Exemple	Résultat
+	Addition	2 + 3	5
-	Soustraction	2 – 3	-1
*	Multiplication	2 * 3	6
/	Division décimale	2/3	0.6666
//	Division euclidienne (division entière)	3//2 2//3	1 0
%	Modulo (reste de la division entière)	3%2 4%2	1 0
**	Elévation à la puissance	2**3	8

Les opérateurs de comparaison

Symbole	Nom	Exemple	Résultat
<	Plus petit que	2 < 3	True
>	Plus grand que	3 > 2	False
<=	Plus petit ou égal à	2 <= 2	True
>=	Plus grand ou égal à	2 >= 3	False
==	Égal à	2 == 2	True
!=	Différent de	2 != 3	True

Bien faire la différence entre **=** qui sert à **affecter** une valeur à une variable et **==** qui sert à **comparer** si une variable ou une valeur est **égale** à une autre.

Tester une condition

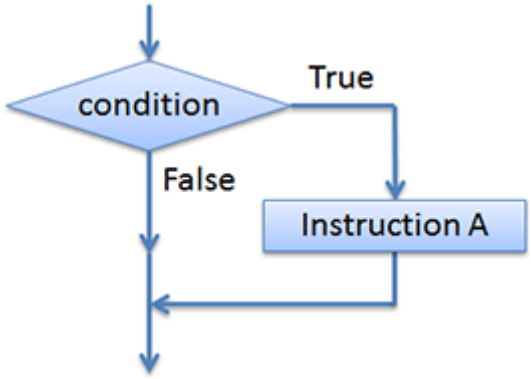
Instruction if … else…

Première forme

if condition :

    Instructions si condition vraie

Suite du programme



Deuxième forme

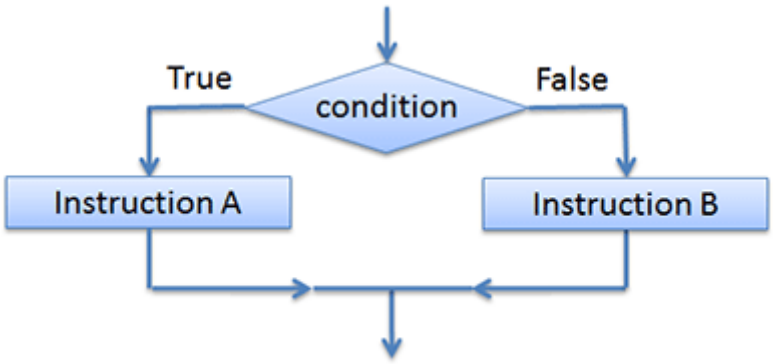
if condition :

    Instructions si condition vraie

else :

    Instructions si condition fausse

Suite du programme



Troisième forme

if condition1 :

    Instructions si condition1 vraie

elif condition2 :

    Instructions si condition2 vraie

else :

    Instructions si condition2 fausse

Suite du programme

Quatrième forme

if condition1 :

    Instructions si condition1 vraie

    if condition2 :

        Instructions si condition2 vraie

    else :

        Instructions si condition2 fausse

else :

    Instructions si condition1 fausse

Suite du programme

Les boucles

Boucle for

Boucle bornée for quand on sait combien de fois doit avoir lieu la répétition d'un bloc d'instructions.

for compteur :

    Instructions à répéter un nombre de fois égale au compteur

Suite du programme

Comment fonctionne le compteur ?

for i in range(3) :

Ici, la boucle sera exécutée 3 fois. La variable i prendra succéssivement les valeurs 0,1,2.

for i in range(2, 8) :

La boucle sera exécutée 6 fois, i prenant succéssivement les valeurs de 2 à 7.

for caractere in 'hello' :

La boucle sera exécutée 5 fois, la variable caractere prendra succéssivement les valeurs h, e, l, l, o

for i in range (10, 0, -2) :

La boucle sera exécutée 5 fois, i prenant succéssivement les valeurs de 10, 8, 6, 4, 2.

Boucles imbriquées

Exemple :

```
for i in range(3):
    for j in range(5):
        print(i,j,end = " ") #affiche i et j sur sans passer à la ligne après l'affichage
    print() #on saute à la ligne lorsque la valeur de i change
```

La boucle sur j est interne à celle sur i. Donc pour chaque valeur de i, on fait 5 tours dans la boucle sur j.

La variable i prend la valeur 0, puis la variable j prend les valeurs 0 à 4.

La variable i prend la valeur 1, puis la variable j prend les valeurs 0 à 4.

La variable i prend la valeur 2, puis la variable j prend les valeurs 0 à 4.

Le résultat du programme est (i en rouge):

0 0 0 1 0 2 0 3 0 4

1 0 1 1 1 2 1 3 1 4

2 0 2 1 2 2 2 3 2 4

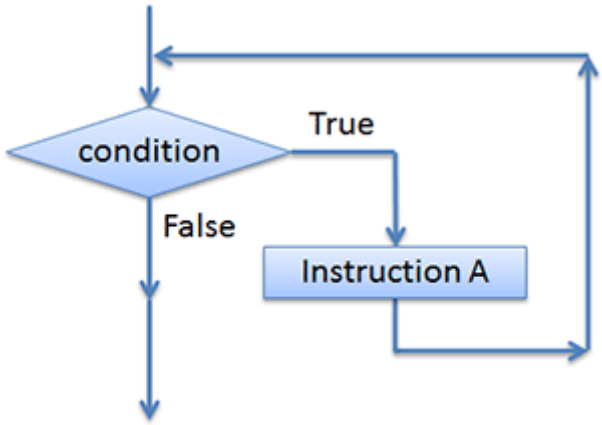
Boucle while

Boucle while, qui ne peut s'arrêter que par un test.

While condition :

Instructions à exécuter tant que la condition est vraie

Suite du programme



Les instructions break et continue et la clause else dans les boucles

Attention : ces instructions sont déconseillées mais peuvent parfois aider.

Break

L'instruction **break** permet de casser l'exécution d'une boucle for ou while. Elle permet de sortir de la boucle, même si celle-ci n'est pas terminée.

Reprenons l'exemple des boucles imbriquées et rajoutons un test qui nous fait sortie de la boucle sur j, lorsque j est supérieur à i .

```
for i in range(3):
    for j in range(5):
        if j > i :
            break
        print(i,j,end = " ")
    print()
```

Résultat: (i en rouge)

0 0

1 0 1 1

2 0 2 1 2 2

On peut remarquer que la valeur de j, ne dépasse jamais celle de i.

Continue

L'instruction **continue**, permer de remonter en haut d'une boucle et de passer au tour suivant dans la boucle, sans exécuter les instructions qui la suivent et qui font partie des instructions de la boucle.

Exemple :

```
for num in range(2, 16):
   if num % 2 == 0:
      print(num, " est pair")
   if num % 3 == 0:
      print(num, " est divisible par 3")
   if num % 5 == 0:
      print(num, " est divisible par 5")
```

Résultat :

```
2  est pair
3  est divisible par 3
4  est pair
5  est divisible par 5
6  est pair
6  est divisible par 3
8  est pair
9  est divisible par 3
10  est pair
10  est divisible par 5
12  est pair
12  est divisible par 3
14  est pair
15  est divisible par 3
15  est divisible par 5
```

Si on rajoute une instruction continue, dans chaque if

```
for num in range(2, 16):
    if num % 2 == 0:
        print(num, " est pair")
        continue
    if num % 3 == 0:
        print(num, " est divisible par 3")
        continue
    if num % 5 == 0:
        print(num, " est divisible par 5")
        continue
```

Résultat :

2 est pair

3 est divisible par 3

4 est pair

5 est divisible par 5

6 est pair → dès que l'on a déterminé que 6 était pair, on passe à la valeur de num suivante

8 est pair

9 est divisible par 3

10 est pair → on ne passe plus sur les tests suivants

12 est pair → on ne passe plus sur les tests suivants

14 est pair

15 est divisible par 3 → on ne passe plus sur le test suivant

Else en fin de boucle

La clause **else** dans un boucle permet de définir un bloc d'instructions qui sera exécuté à la fin seulement si la boucle s'est déroulée complétement sans être interrompue par un **break**.

```
i = 1

while i <= 10:

    print(i," ", end=" ")

    i = i+1

else:

    print()

    print('La boucle se termine avec i =', i)
```

Résultat :

```
1   2   3   4   5   6   7   8   9   10

La boucle se termine avec i = 11
```