Conditions et boucles

Alexandre Benoit

BCPST

Notion d'algorithme

Un algorithme est une procédure permettant de résoudre un problème, écrite de façon suffisamment détaillée pour être suivie sans posséder de compétence particulière ni même comprendre le problème que l'on est en train de résoudre.

Notion d'algorithme

Un algorithme est une procédure permettant de résoudre un problème, écrite de façon suffisamment détaillée pour être suivie sans posséder de compétence particulière ni même comprendre le problème que l'on est en train de résoudre.
Les Instructions de base de l'algorithme sont :

1. La déclaration et l'affectation de variable ;

Notion d'algorithme

Un algorithme est une procédure permettant de résoudre un problème, écrite de façon suffisamment détaillée pour être suivie sans posséder de compétence particulière ni même comprendre le problème que l'on est en train de résoudre.
Les Instructions de base de l'algorithme sont :

1. La déclaration et l'affectation de variable ;
2. la séquence, qui exécute deux instructions l'une à la suite de l'autre.

Notion d'algorithme

Un algorithme est une procédure permettant de résoudre un problème, écrite de façon suffisamment détaillée pour être suivie sans posséder de compétence particulière ni même comprendre le problème que l'on est en train de résoudre.
Les Instructions de base de l'algorithme sont :

1. La déclaration et l'affectation de variable ;
2. la séquence, qui exécute deux instructions l'une à la suite de l'autre.
3. le test, ou instruction conditionnelle, qui sert à n'exécuter une instruction que dans certains états ;

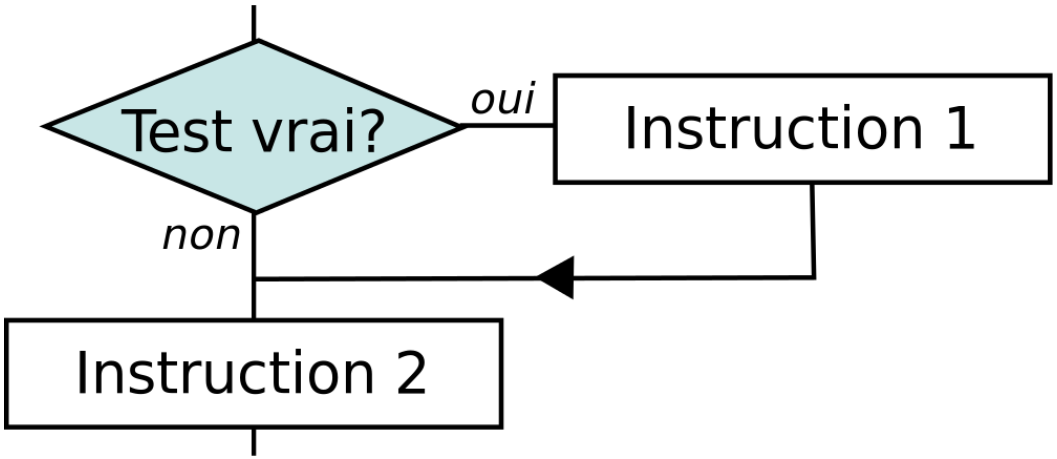
Notion d'algorithme

Un algorithme est une procédure permettant de résoudre un problème, écrite de façon suffisamment détaillée pour être suivie sans posséder de compétence particulière ni même comprendre le problème que l'on est en train de résoudre.
Les Instructions de base de l'algorithme sont :

1. La déclaration et l'affectation de variable ;
2. la séquence, qui exécute deux instructions l'une à la suite de l'autre.
3. le test, ou instruction conditionnelle, qui sert à n'exécuter une instruction que dans certains états ;
4. la boucle, qui exécute plusieurs fois la même instruction dans un programme.

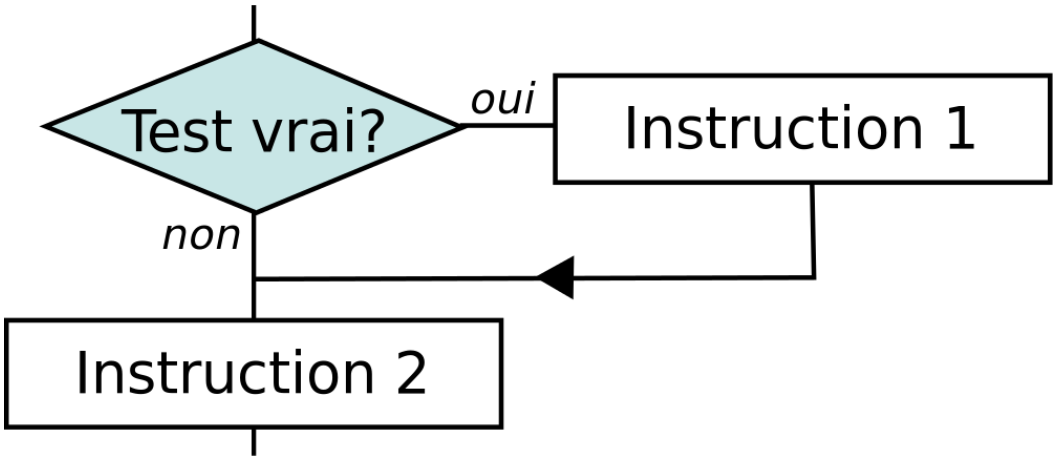
Instructions conditionnelles

If

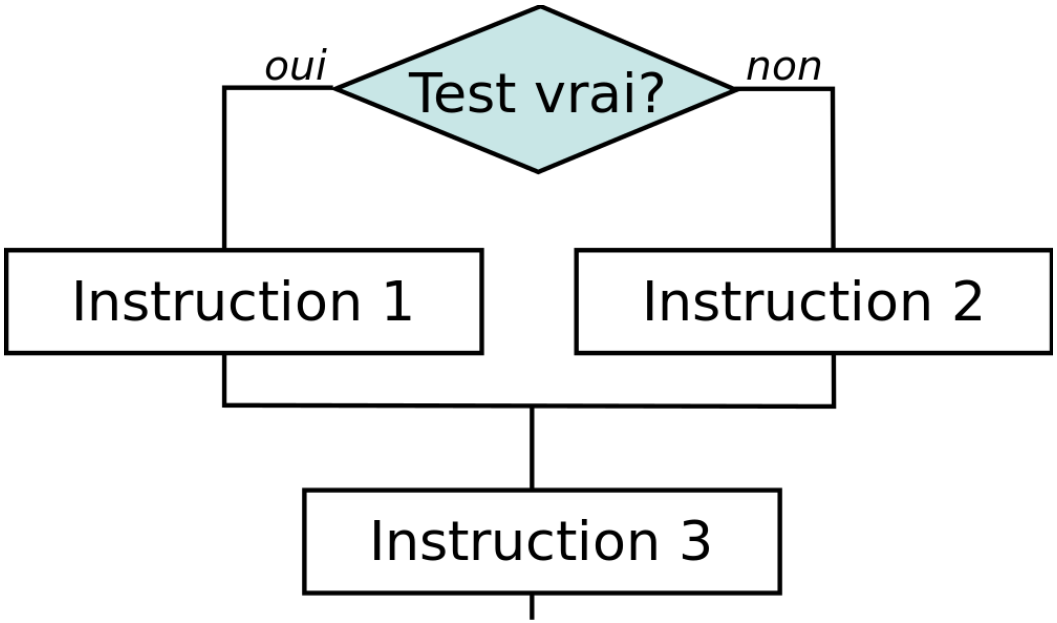


Instructions conditionnelles

If



Else



Instructions conditionnelles : Exemple

Une instruction conditionnelle n'est exécutée que si une condition donnée est vérifiée par l'état courant.

Instructions conditionnelles : Exemple

Une instruction conditionnelle n'est exécutée que si une condition donnée est vérifiée par l'état courant.

```
if x%2 == 1:
    x=x/2
else:
    x=x+1
```

Instructions conditionnelles : Exemple

Une instruction conditionnelle n'est exécutée que si une condition donnée est vérifiée par l'état courant.

```
if x%2 == 1:
    x=x/2
else:
    x=x+1
```

- Pour identifier sans ambiguïté les instructions appartenant au bloc du if, il est nécessaire de les indenter ;

Instructions conditionnelles : Exemple

Une instruction conditionnelle n'est exécutée que si une condition donnée est vérifiée par l'état courant.

```
if x%2 == 1:
    x=x/2
else:
    x=x+1
```

- Pour identifier sans ambiguïté les instructions appartenant au bloc du if, il est nécessaire de les indenter ;
- if vérifie la valeur d'un booléen. Ici on a le booléen x%2 ;

Instructions conditionnelles : Exemple

Une instruction conditionnelle n'est exécutée que si une condition donnée est vérifiée par l'état courant.

```
if x%2 == 1:
    x=x/2
else:
    x=x+1
```

- Pour identifier sans ambiguïté les instructions appartenant au bloc du if, il est nécessaire de les indenter ;
- if vérifie la valeur d'un booléen. Ici on a le booléen x%2 ;
- la ligne x=x+1 n'est exécuté que si le nombre `x` est pair.

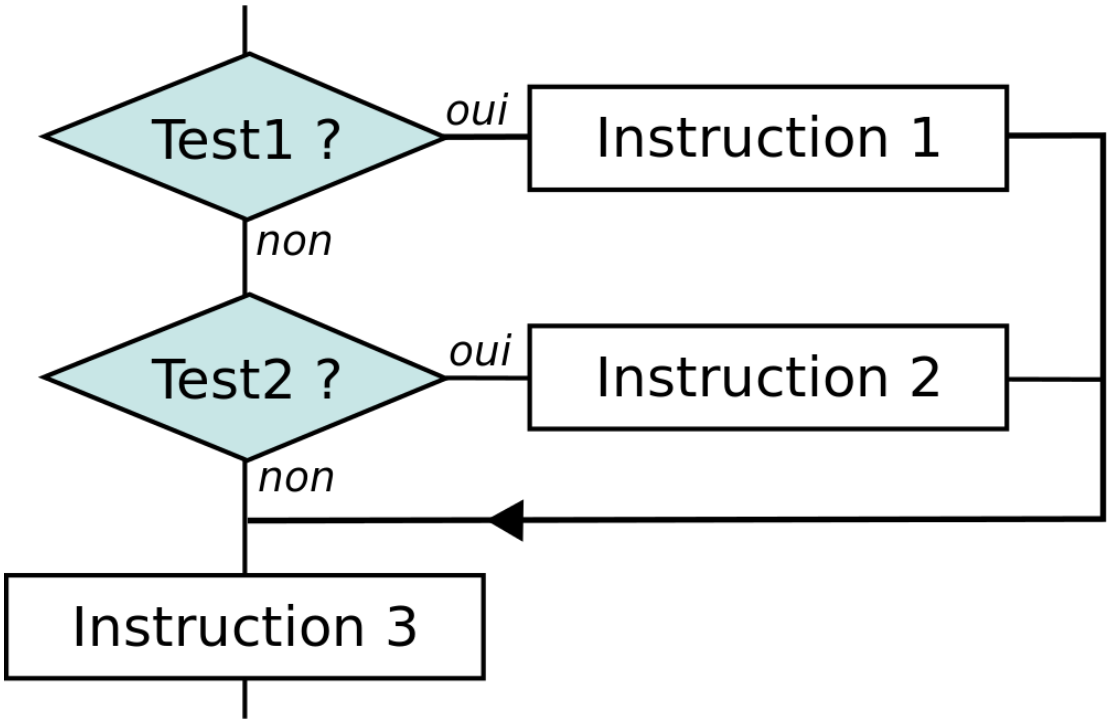
Instructions conditionnelles : Exemple

Une instruction conditionnelle n'est exécutée que si une condition donnée est vérifiée par l'état courant.

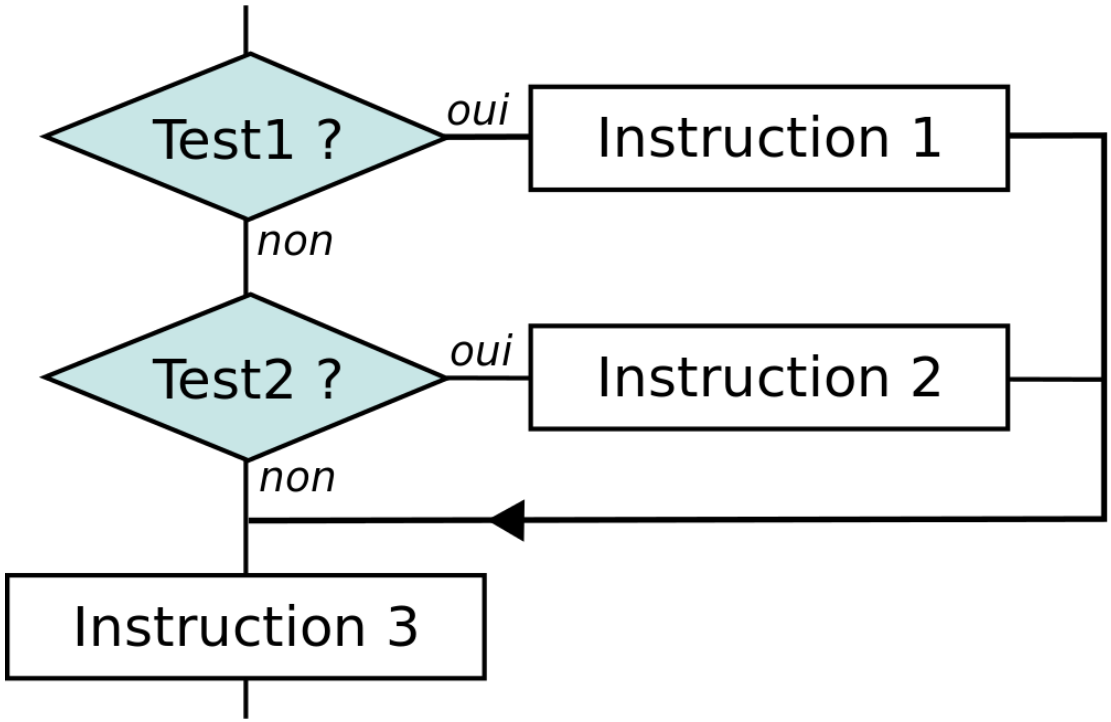
```
if x%2 == 1:
    x=x/2
else:
    x=x+1
```

- Pour identifier sans ambiguïté les instructions appartenant au bloc du if, il est nécessaire de les indenter ;
- if vérifie la valeur d'un booléen. Ici on a le booléen x%2 ;
- la ligne x=x+1 n'est exécuté que si le nombre `x` est pair.
- la ligne x=x/2 n'est exécuté que si le nombre `x` est impair.

elif

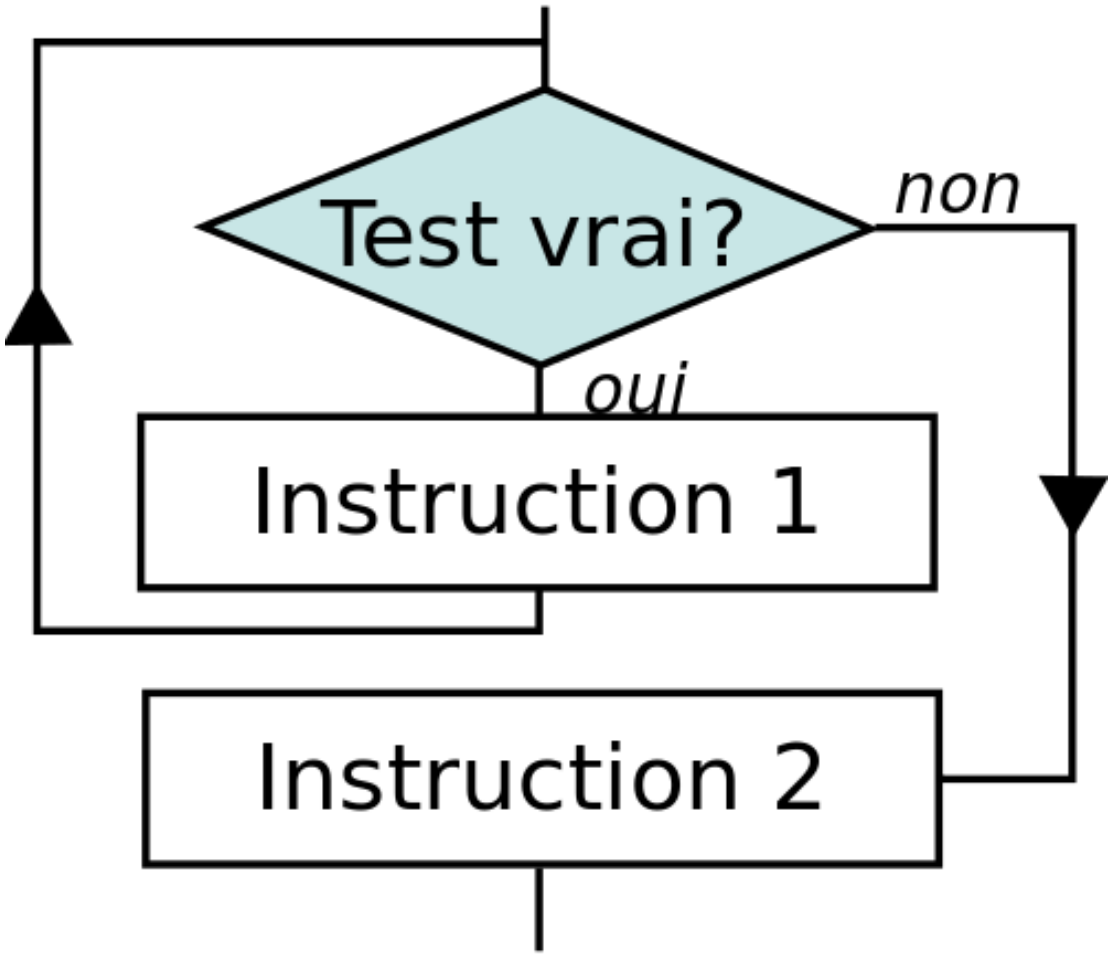

Test1 ?
oui
Instruction 1
non
Test2 ?
oui
Instruction 2
non
Instruction 3

elif




```
if x%2 == 1:
    x = x/2
elif x%3 == 1:
    x = x/3
else:
     x = x + 1
```

Boucle conditionnelle



Boucle conditionnelle : Exemple 1

On se donne les lignes suivantes :

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

- On indente les instructions appartenant au bloc du while comme pour le if ;

Boucle conditionnelle : Exemple 1

On se donne les lignes suivantes :

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

- On indente les instructions appartenant au bloc du while comme pour le if ;
- Ce programme bouclera autour des instructions tant que le test ne sera pas vérifié ;

Boucle conditionnelle : Exemple 1

On se donne les lignes suivantes :

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

- On indente les instructions appartenant au bloc du while comme pour le if ;
- Ce programme bouclera autour des instructions tant que le test ne sera pas vérifié ;
- Ce bloc d'instructions est appelé corps de la boucle et chaque passage est appelé une itération ;

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p						
c						
c>0						

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1					
c						
c>0						

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1					
c	5					
c>0						

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1					
c	5					
c>0	Vrai					

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2				
c	5					
c>0	Vrai					

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2				
c	5	4				
c>0	Vrai					

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2				
c	5	4				
c>0	Vrai	Vrai				

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2	4			
c	5	4				
c>0	Vrai	Vrai				

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2	4			
c	5	4	3			
c>0	Vrai	Vrai				

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2	4			
c	5	4	3			
c>0	Vrai	Vrai	Vrai			

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2	4	8	16	32
c	5	4	3	2	1	0
c>0	Vrai	Vrai	Vrai	Vrai	Vrai	

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2	4	8	16	32
c	5	4	3	2	1	0
c>0	Vrai	Vrai	Vrai	Vrai	Vrai	Faux

Boucle conditionnelle : Exemple 2

```
p = 1
c = 5
while c>0:
    p = p*2
    c = c-1
print(p)
```

État des variables :

p	1	2	4	8	16	32
c	5	4	3	2	1	0
c>0	Vrai	Vrai	Vrai	Vrai	Vrai	Faux

Cet algorithme affiche 32

Boucle conditionnelle : Terminaison

Les boucles **while** peuvent ne pas finir. Par exemple :

```
p = 1
c = 5
while c>0:
    p = p*2
```

Expliquer ce qu'il se passe

Boucle conditionnelle : Terminaison

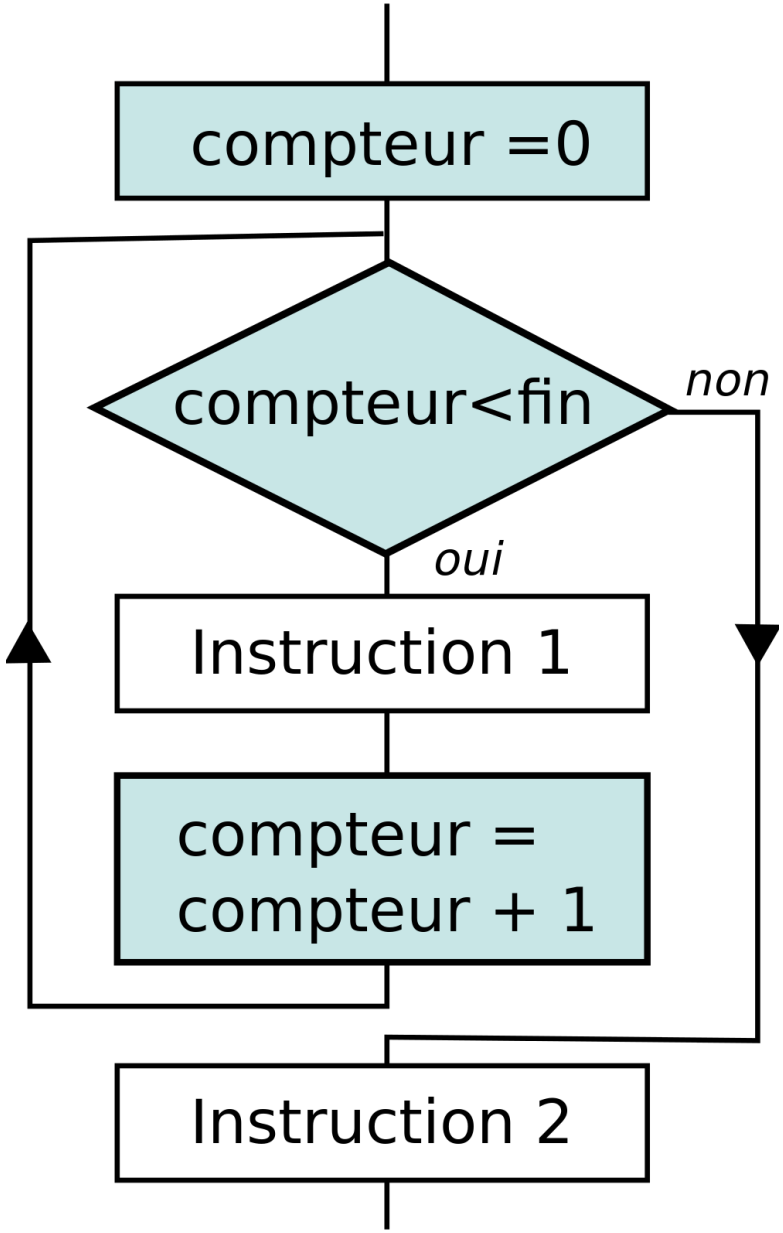
Les boucles **while** peuvent ne pas finir. Par exemple :

```
p = 1
c = 5
while c>0:
    p = p*2
```

Expliquer ce qu'il se passe

p	1	2	4	8	16	.....
c	5	5	5	5	5	.....
c>0	Vrai	Vrai	Vrai	Vrai	Vrai	.....

Boucle for


compteur =0
compteur<fin
non
oui
Instruction 1
compteur =
compteur + 1
Instruction 2

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c						
p	1					

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c		0				
p	1					

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c		0				
p	1	2				

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c		0	1			
p	1	2				

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c		0	1			
p	1	2	4			

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c		0	1	2	3	4
p	1	2	4	8	16	

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c		0	1	2	3	4
p	1	2	4	8	16	32

Boucle for : Exemple

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(5):
    p = p*2
print(p)
```

- Ici range(n) représente les nombres entre 0 et $n-1$;
-

c		0	1	2	3	4
p	1	2	4	8	16	32

Cet algorithme affiche encore 32

Je dois savoir

- Écrire un petit test en python (exemple : valeur absolue)

Je dois savoir

- Écrire un petit test en python (exemple : valeur absolue)
- Écrire une boucle en python :

Je dois savoir

- Écrire un petit test en python (exemple : valeur absolue)
- Écrire une boucle en python :
  - afficher les premiers termes d'une suite définie par récurrence (exercice 7).

Je dois savoir

- Écrire un petit test en python (exemple : valeur absolue)
- Écrire une boucle en python :
  - afficher les premiers termes d'une suite définie par récurrence (exercice 7).
  - écrire un algorithme de seuil (exercice 7)

Je dois savoir

- Écrire un petit test en python (exemple : valeur absolue)
- Écrire une boucle en python :
  - afficher les premiers termes d'une suite définie par récurrence (exercice 7).
  - écrire un algorithme de seuil (exercice 7)
  - afficher la somme des termes d'une liste (exercice 5).

Je dois savoir

- Écrire un petit test en python (exemple : valeur absolue)
- Écrire une boucle en python :
  - afficher les premiers termes d'une suite définie par récurrence (exercice 7).
  - écrire un algorithme de seuil (exercice 7)
  - afficher la somme des termes d'une liste (exercice 5).
  - Les exercices de 1 à 7

Références

Ce cours est issu en grande partie du livre *Informatique pour tous* de Wack et autres aux éditions **Eyrolles**
Les illustrations sont dues à Ske.