PARADIGMES DE PROGRAMMATION

DIPLÔME DE FORMATION D'INGÉNIEURS EN INFORMATIQUE (FI-A1)

Module 3

Mondher Bouden

2025-2026

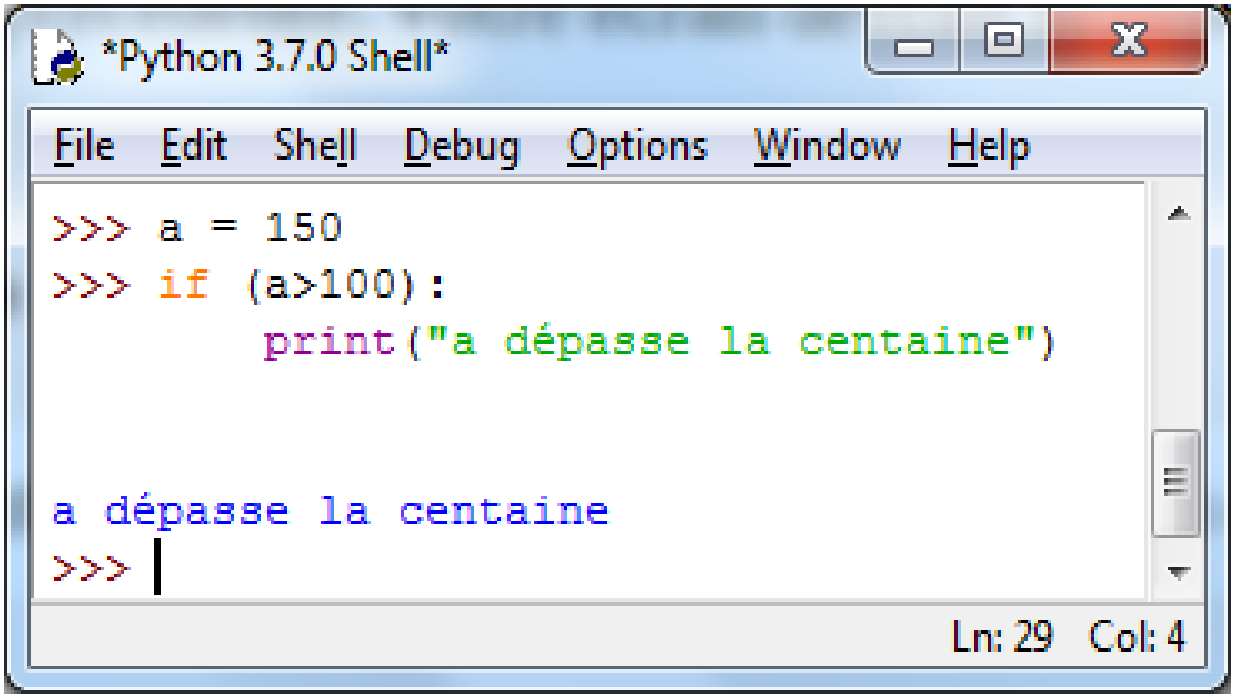
Les structures de contrôle

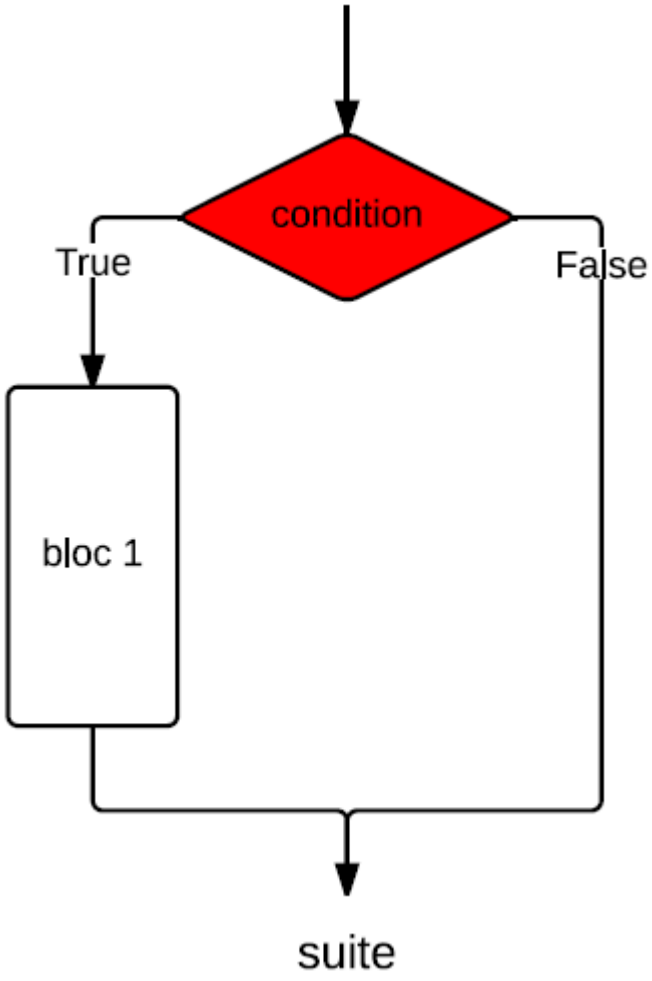
(Instructions conditionnelles vs répétitives)

Les instructions conditionnelles

Structure *si*

- Si (une **condition** est vraie) alors
  - ▶ On exécute le bloc 1
- Suite

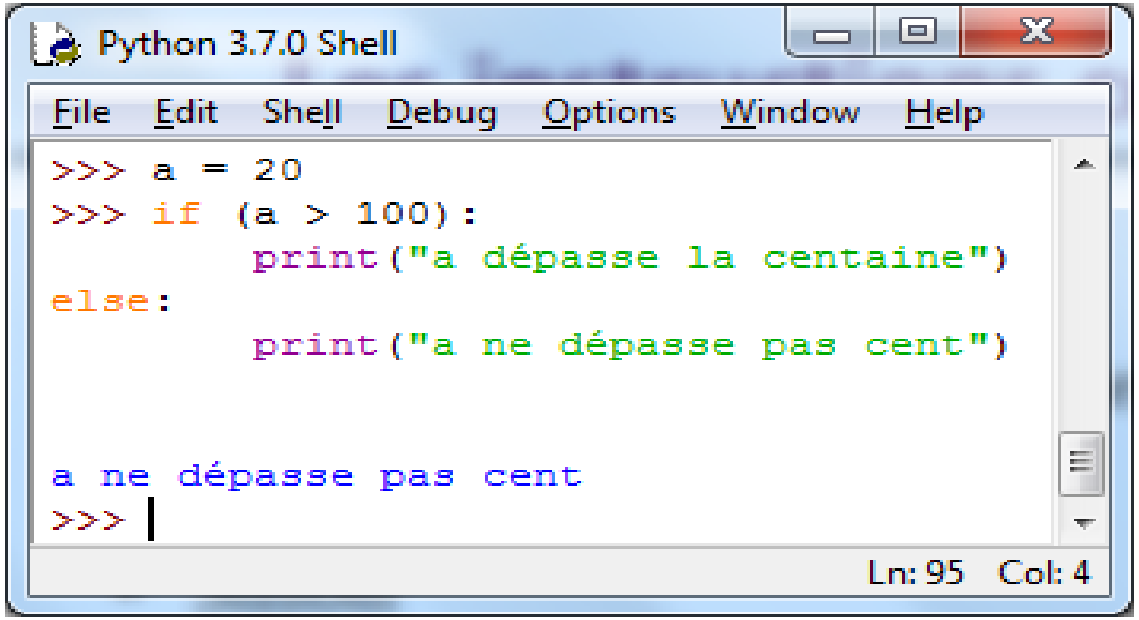


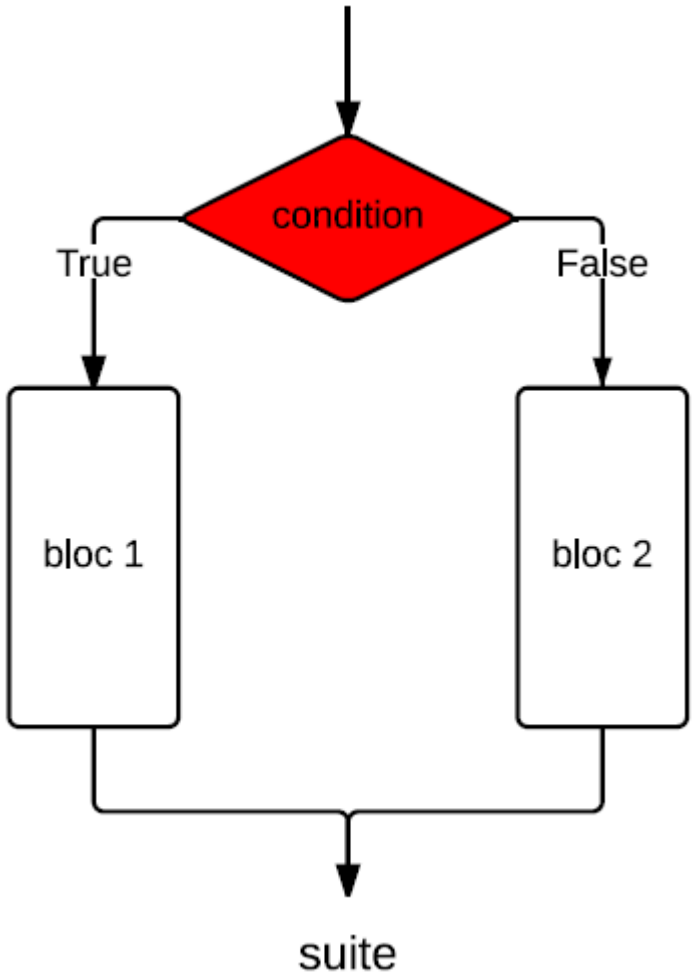


Les instructions conditionnelles

Structure *si* - *sinon*

- Si (une **condition** est vraie) alors
  - ▶ On exécute le bloc 1
- Sinon
  - ▶ On exécute le bloc 2
- Suite

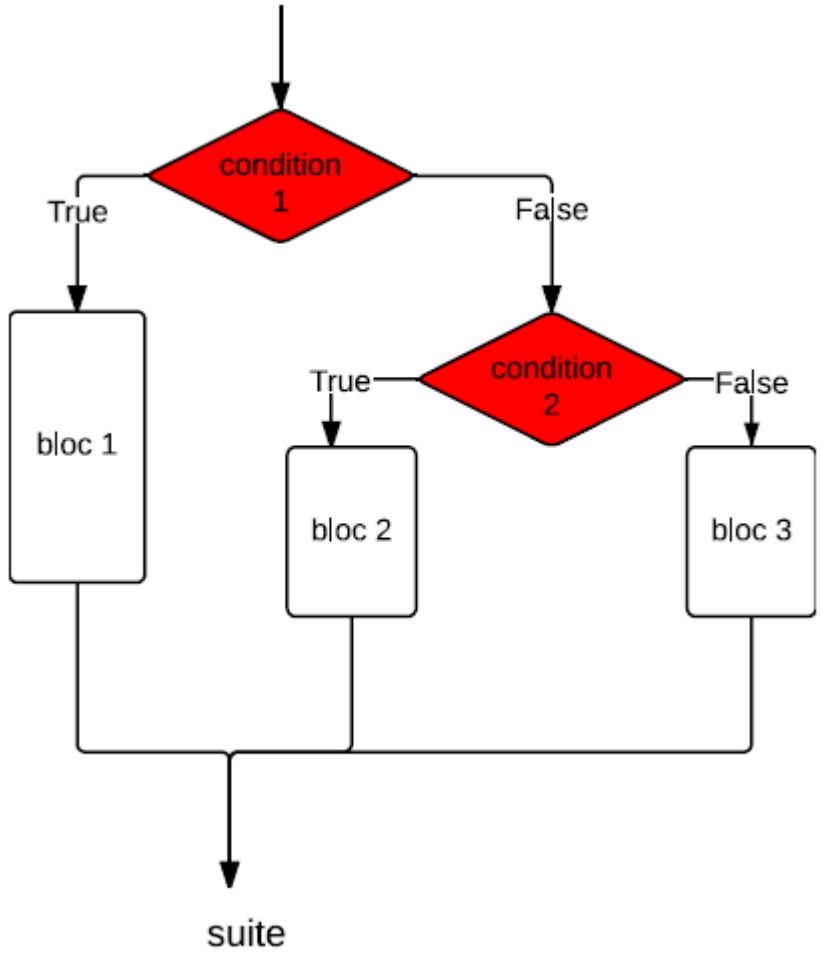




Les instructions conditionnelles

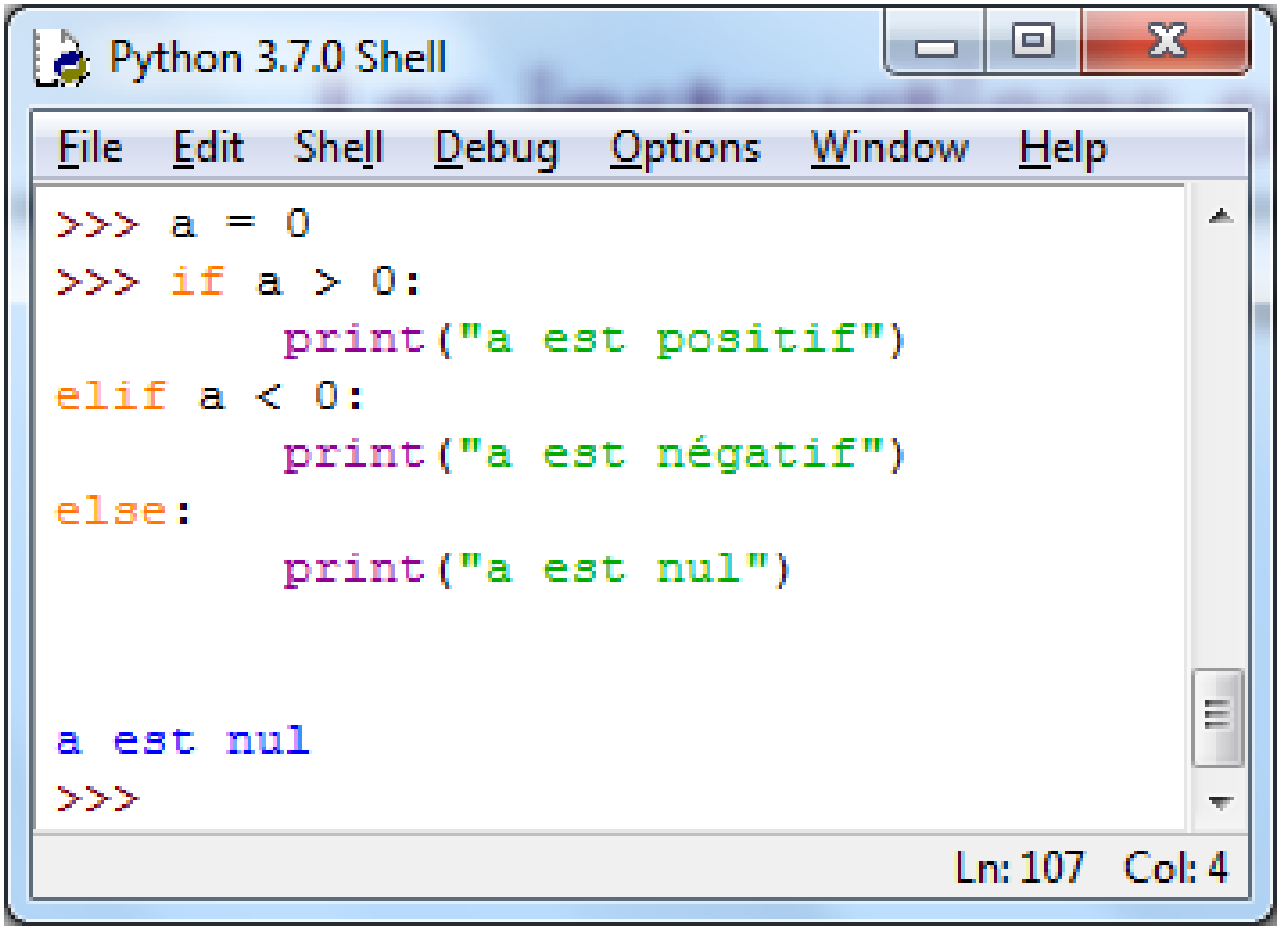
Structure *si-sinon* imbriquées

- Si condition1 alors
  - ▶ On exécute le bloc 1
- Sinon
  - ▶ Si condition2 alors
    - On exécute le bloc 2
  - ▶ Sinon
    - On exécute le bloc 3
- Suite



Les instructions conditionnelles

On peut également utiliser l'instruction **elif** (contraction de « else if ») :



Les parenthèses (pour la condition du *if)* ne sont pas obligatoires.

Exemple 1

Exempleif_ageretraite.py - C:/Exemples/Exempleif_ageretraite.py (3.7.0)

File Edit Format Run Options Window Help

```
age_embauche = int(input("Âge à l'embauche : "))
age_retraite = int(input("Âge à la retraite : "))
erreur_emb = age_retraite < age_embauche

if not erreur_emb:
    anciennete = age_retraite - age_embauche
    print("années de travail : ", anciennete)
else:
    print("Erreur: Âge à la retraite < Âge à l'embauche")
```

Ln: 9 Col: 57

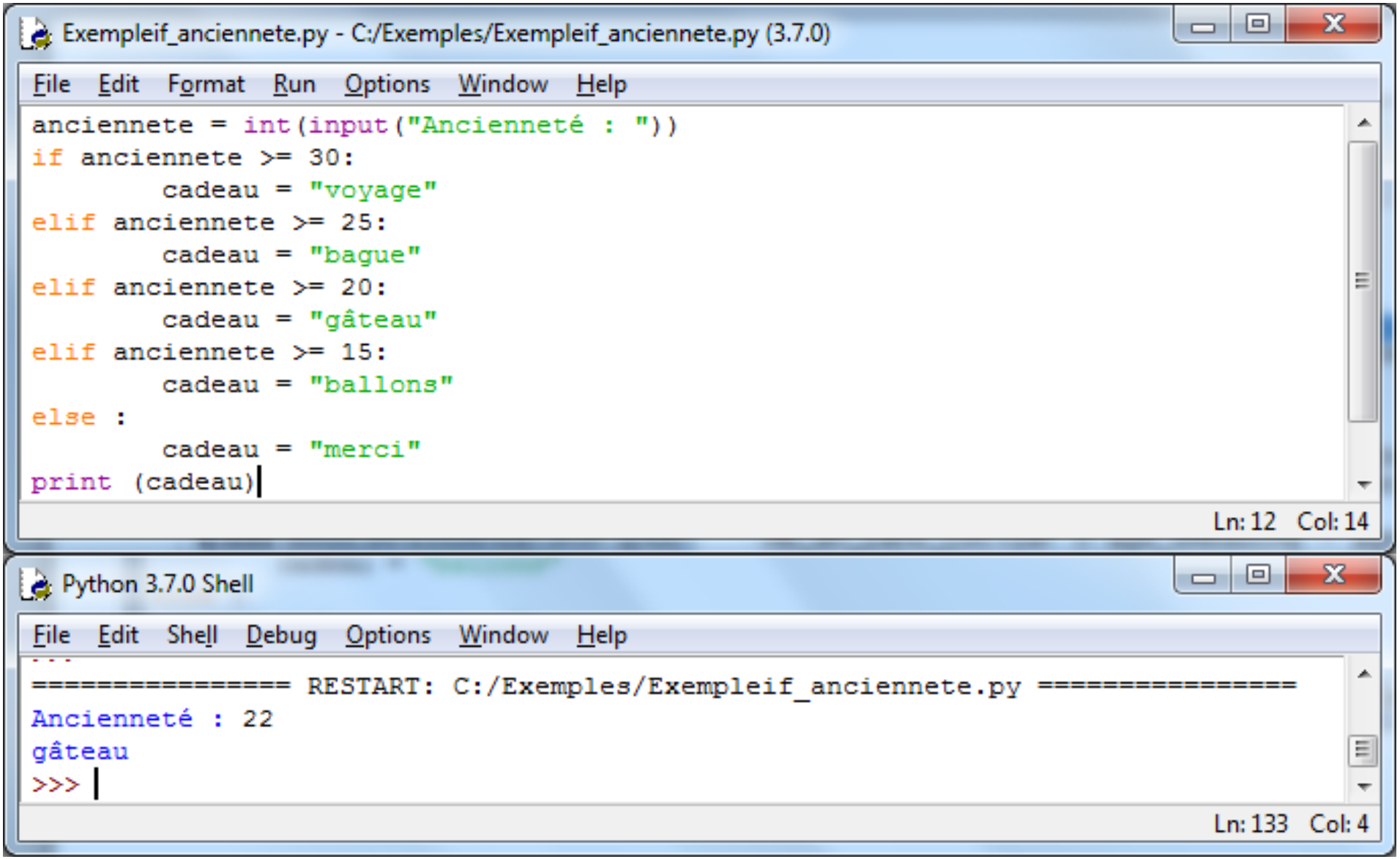
Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

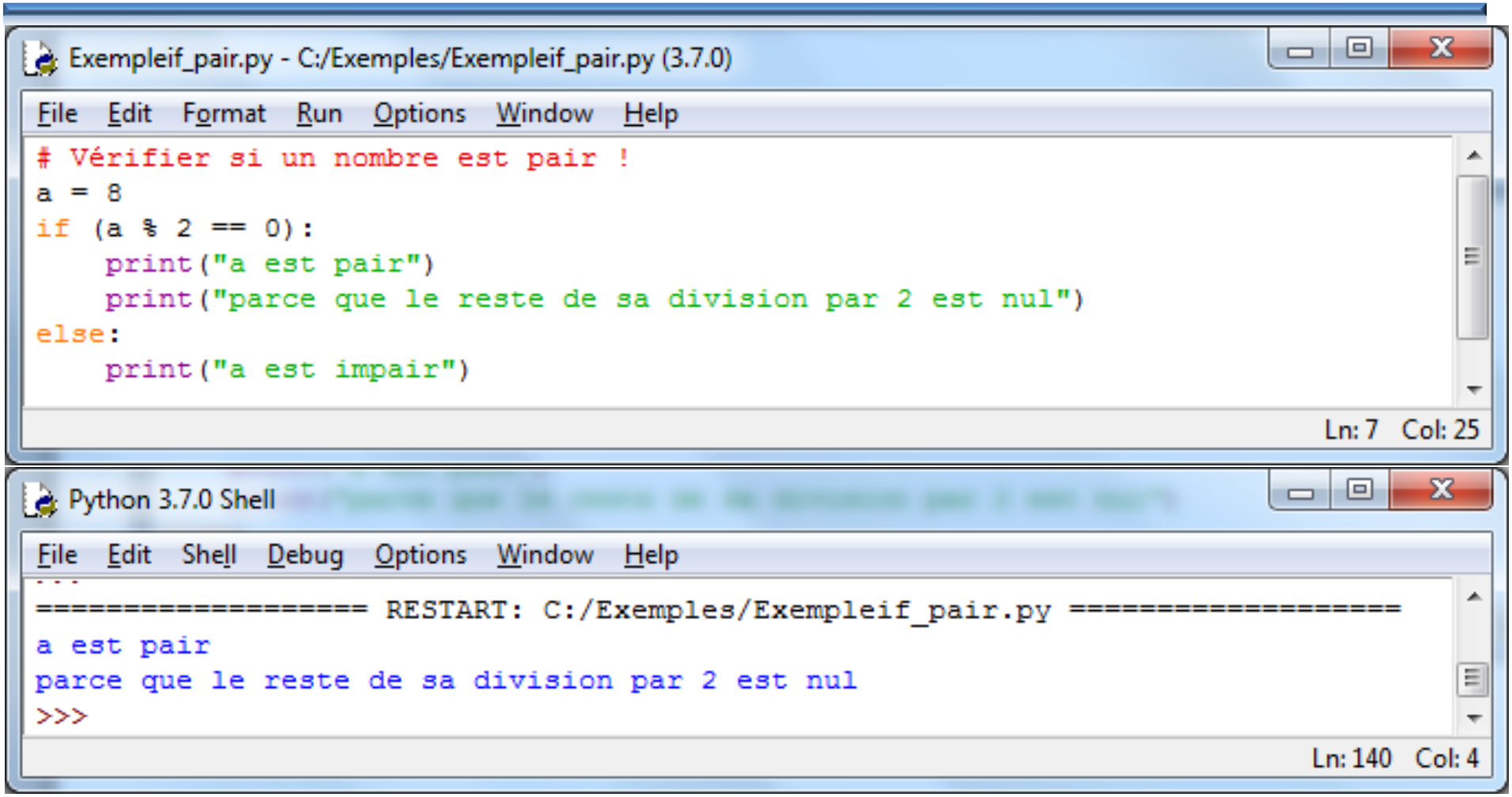
```
============== RESTART: C:/Exemples/Exempleif_ageretraite.py ==============
Âge à l'embauche : 2002
Âge à la retraite : 2023
années de travail :  21
>>>
```

Ln: 117 Col: 0

Exemple 2


Exempleif_anciennete.py - C:/Exemples/Exempleif_anciennete.py (3.7.0)
File Edit Format Run Options Window Help
anciennete = int(input("Ancienneté : "))
if anciennete >= 30:
cadeau = "voyage"
elif anciennete >= 25:
cadeau = "bague"
elif anciennete >= 20:
cadeau = "gâteau"
elif anciennete >= 15:
cadeau = "ballons"
else :
cadeau = "merci"
print (cadeau)
Ln: 12 Col: 14
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
========== RESTART: C:/Exemples/Exempleif_anciennete.py ==========
Ancienneté : 22
gâteau
>>>
Ln: 133 Col: 4

Exemple 3

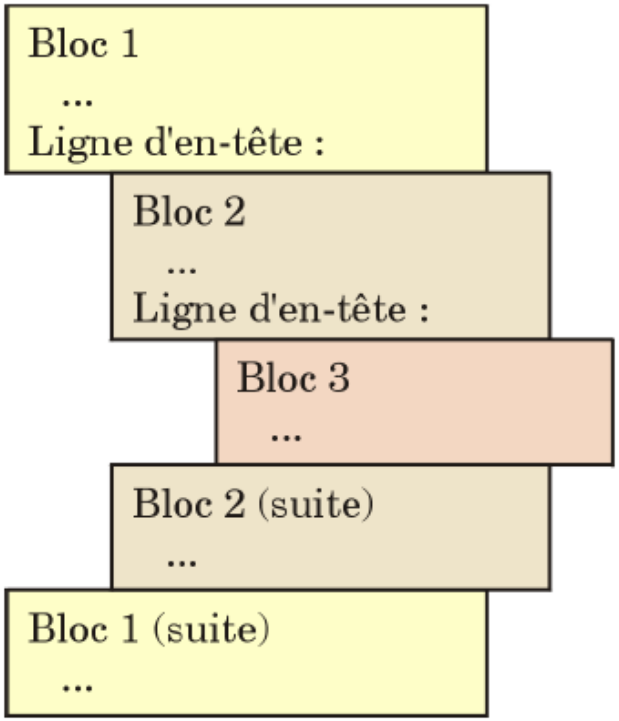


- Notez bien que l'opérateur de comparaison pour l'égalité de deux valeurs est constitué de deux signes « égale ».
- Un commentaire Python commence toujours par le caractère spécial #.

Blocs d'instructions

- Dans la plupart des autres langages, un bloc d'instructions doit être délimité par des symboles spécifiques.
  - Des accolades pour C++ et Java.
- Avec Python, vous devez utiliser les sauts à la ligne et l'indentation.
  - Permettant d'écrire du code lisible, et à prendre de bonnes habitudes.

- Les blocs d'instructions sont toujours associés à une ligne d'en-tête contenant une instruction bien spécifique (if, elif, else, while, def, etc.) se terminant par un double point.
- Les blocs sont délimités par l'indentation : toutes les lignes d'un même bloc doivent être indentées exactement de la même manière.
- Le nombre d'espaces à utiliser pour l'indentation est quelconque, mais la plupart des programmeurs utilisent des multiples de 4.

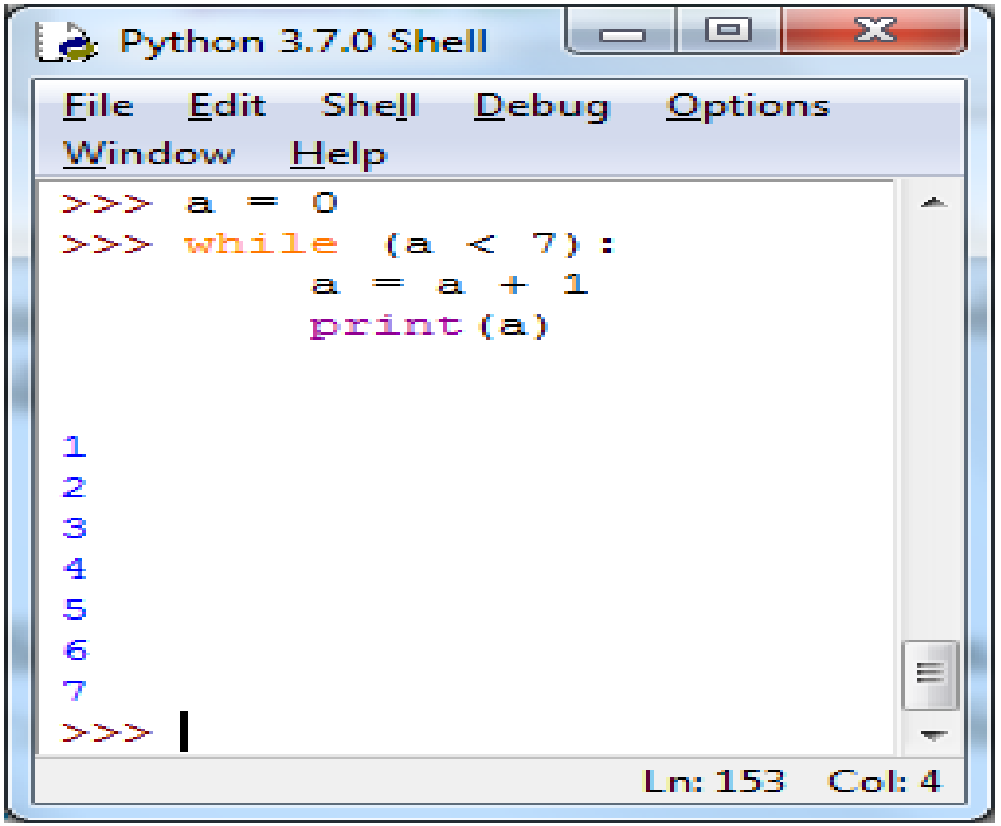


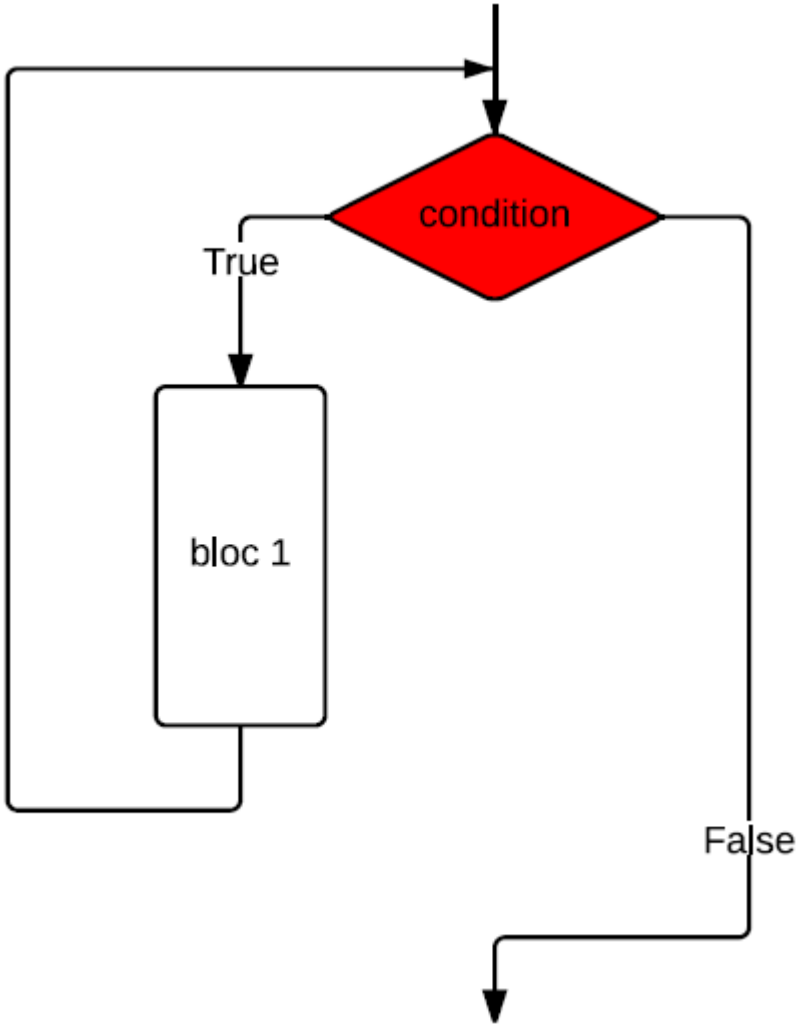
Les instructions répétitives

- Nous distinguons deux types d'instruction répétitives: les boucles while et les boucles for:
- **Boucle while :** on exécute un bloc d'instructions tant que la condition est vraie
  - On ne peut pas (facilement) prédire le nombre d'exécutions du bloc d'instructions, car cela dépend d'une condition à évaluer
- **Boucle for :** on exécute un bloc d'instructions un certain nombre de fois
  - On peut prédire (généralement par calcul) le nombre d'exécutions du bloc d'instructions
  - On exécute systématiquement le bloc d'instructions ce nombre de fois
  - Utilisée pour **parcourir** les éléments d'une structure *itérable*.

La boucle while

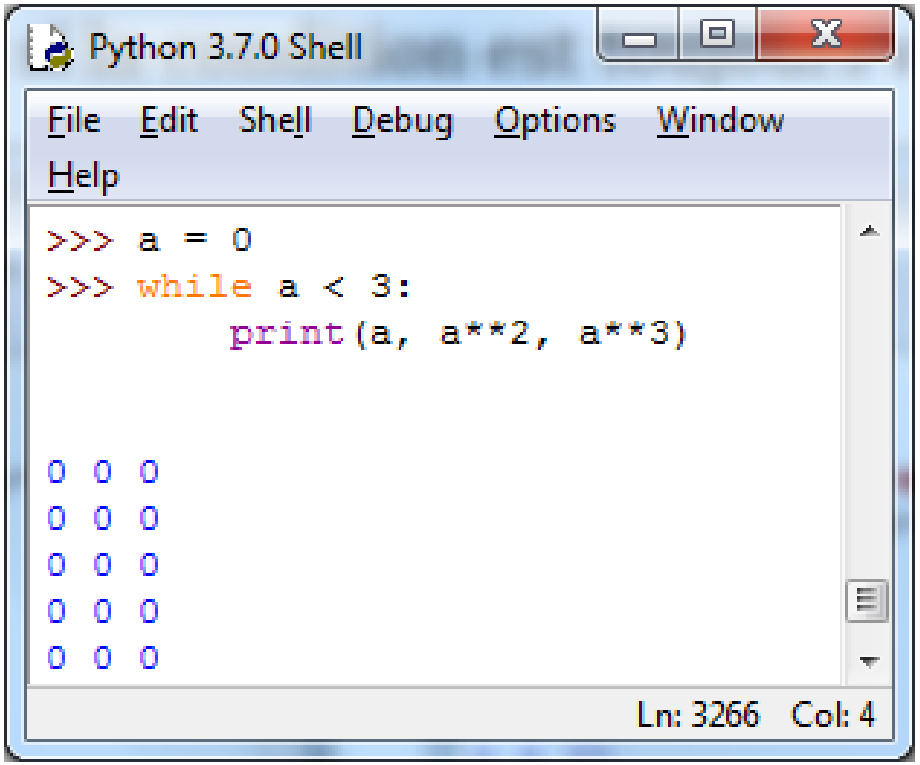
- Tant que condition :
  - ▶ On exécute le bloc 1
- Suite

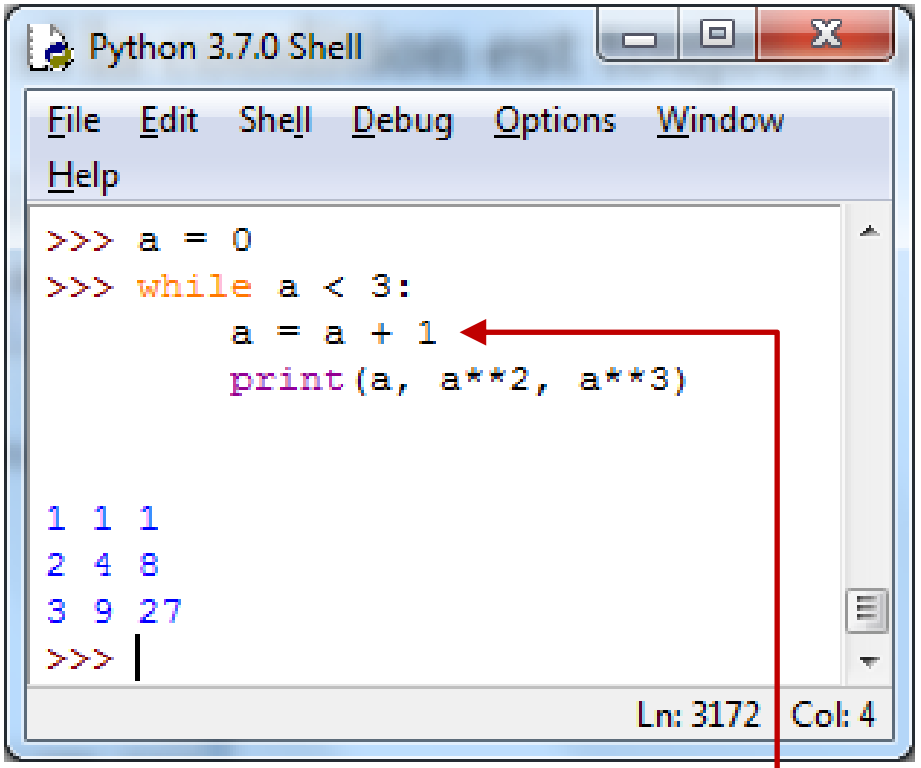




La boucle while

- Qu'arrive-t-il si la condition est toujours vraie ?

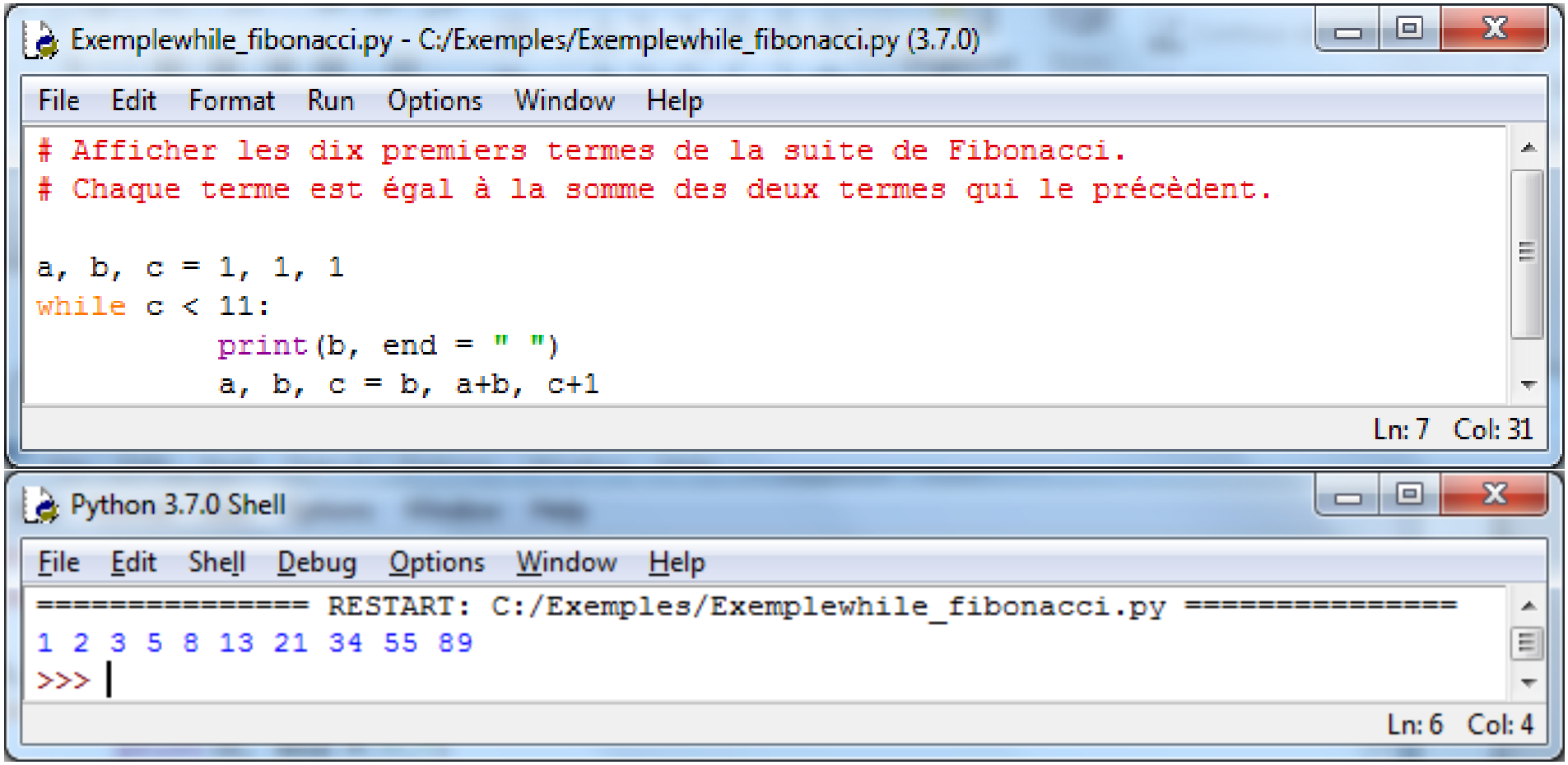




La fonction **print()** permet d'afficher plusieurs expressions l'une à la suite de l'autre sur la même ligne. Python insère automatiquement un espace entre les éléments affichés.

Incrémentation (ou a += 1)

Exemple 1



- L'argument **end =" "** permet de remplacer le saut à la ligne par un simple espace.
- Si vous supprimez cet argument, les nombres seront affichés les uns en-dessous des autres.

Exemple 2 : calculer x^y

$$
\begin{aligned}
x^y &= x \cdot x \cdot x \cdot \ldots \cdot x \quad (y - 1 \text{ multipications}) \\
x^4 &= x \cdot x \cdot x \cdot x \\
x^3 &= x \cdot x \cdot x \\
x^2 &= x \cdot x \\
x^1 &= x \\
x^0 &= 1
\end{aligned}
$$

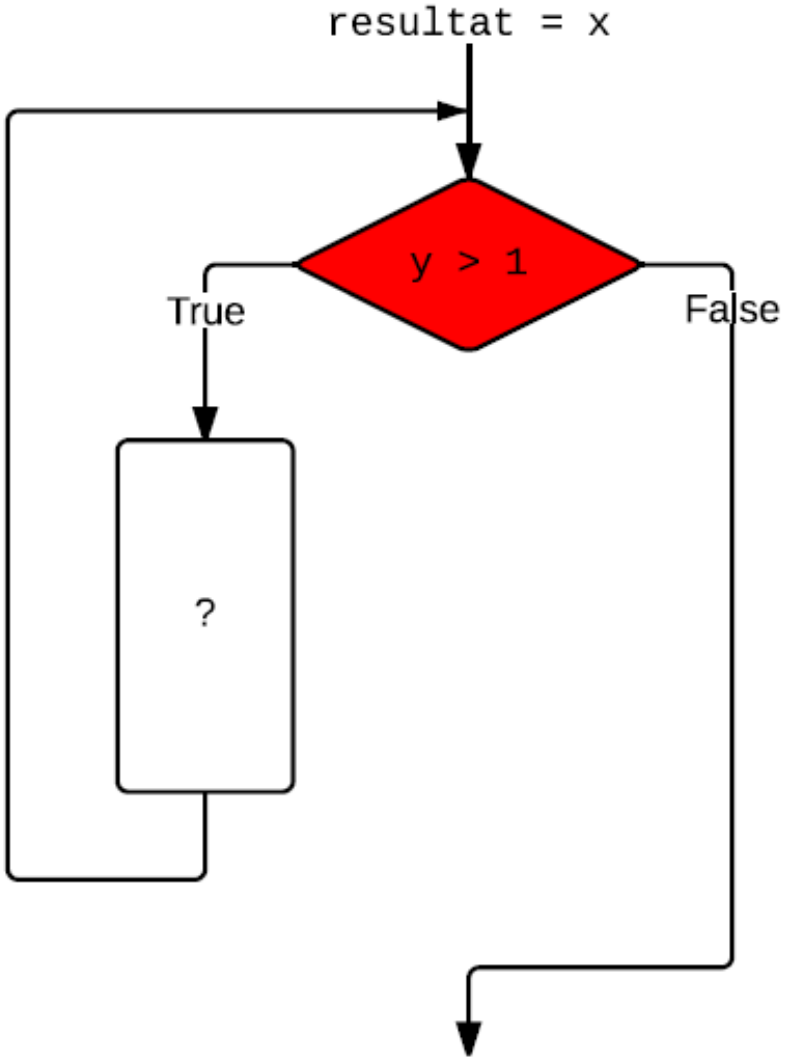
Exemple 2 : calculer x^y

$$x^y = x \cdot x \cdot x \cdot \ldots \cdot x \quad (y - 1 \text{ fois})$$

```
resultat = x

...

print(resultat)
```



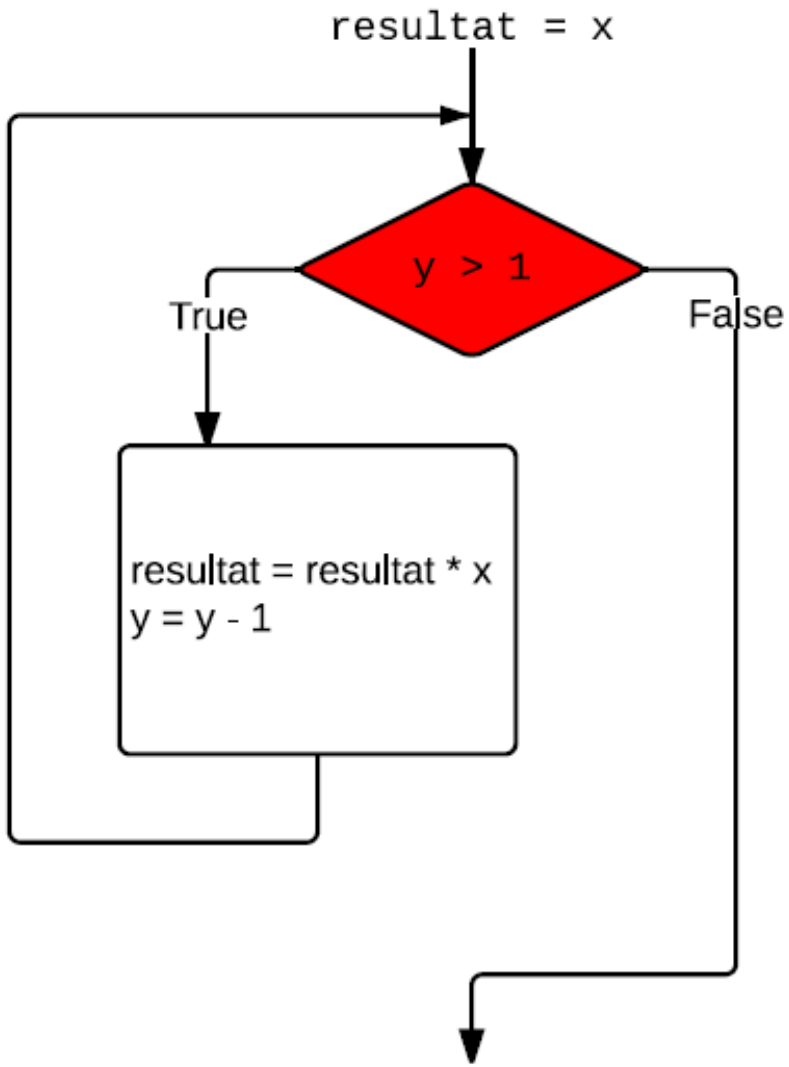
Exemple 2 : calculer x^y

$$x^y = x \cdot x \cdot x \cdot \ldots \cdot x \quad (y - 1 \text{ fois})$$

```
resultat = x

while y > 1:
    resultat = resultat * x
    y = y - 1

print(resultat)
```



Exemple 2 : calculer x^y

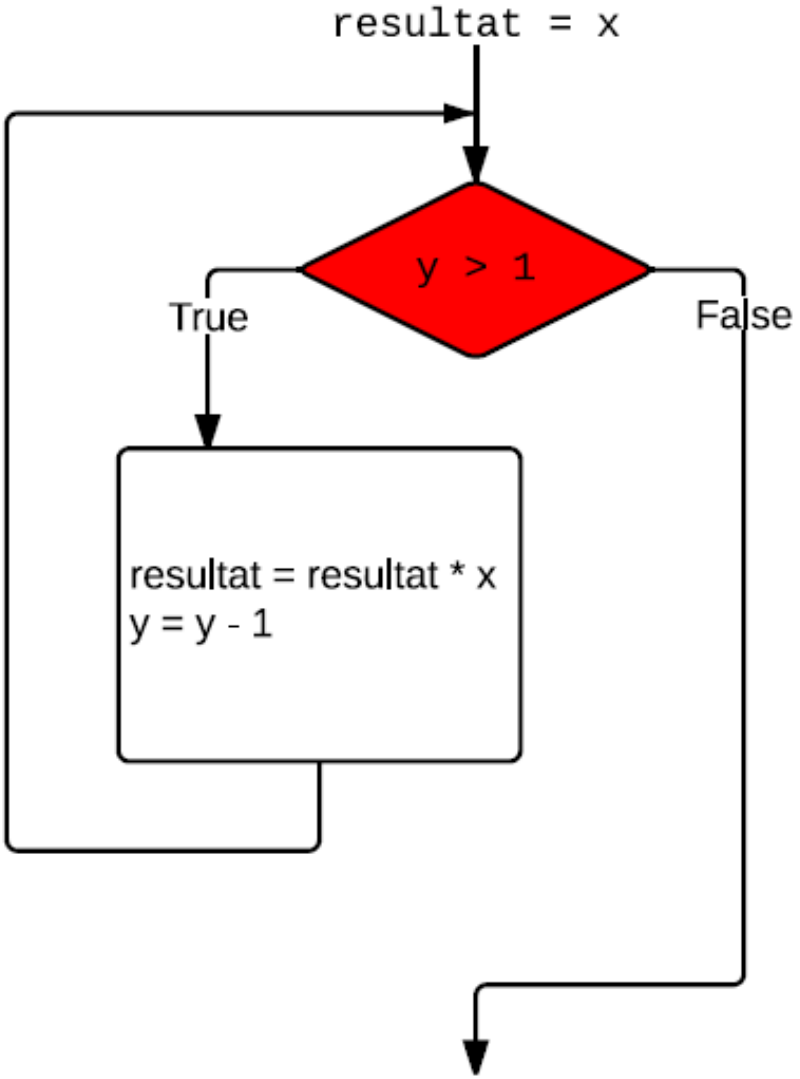
$$x^y = x \cdot x \cdot x \cdot \ldots \cdot x \quad (y - 1 \text{ fois})$$

```
resultat = x

while y > 1:
    resultat = resultat * x
    y = y - 1

print(resultat)
```

Est-ce que la boucle produit toujours le bon résultat ?



Exemple 2 : calculer x^y

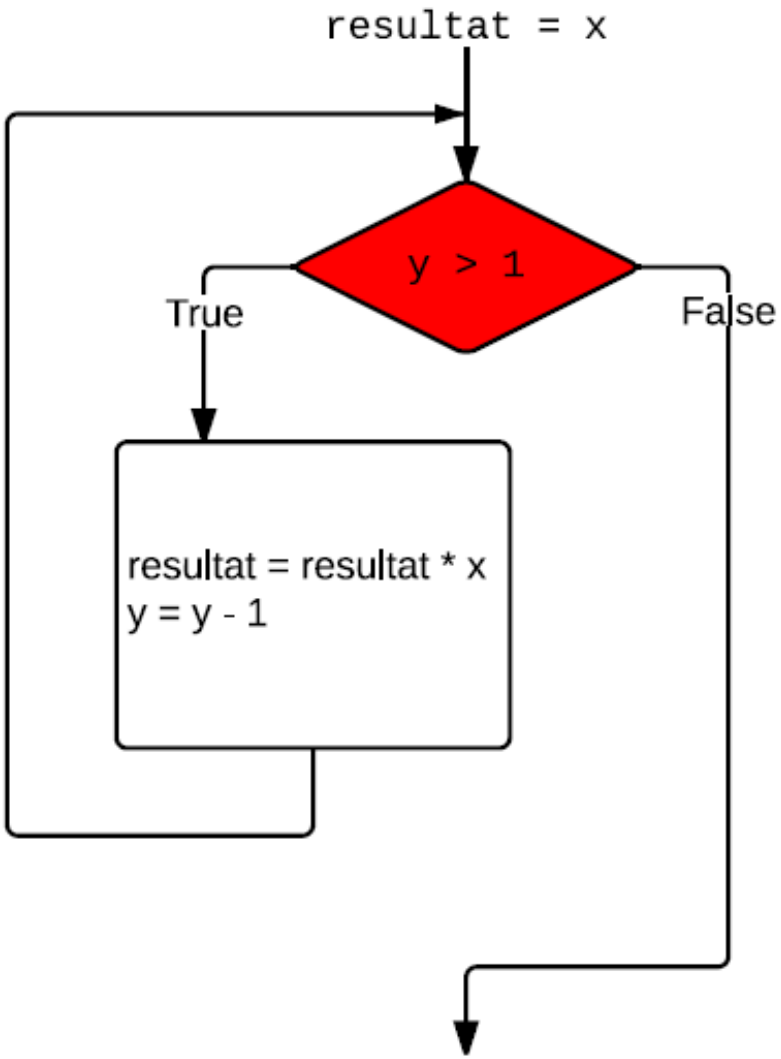
$$x^y = x \cdot x \cdot x \cdot \ldots \cdot x \quad (y - 1 \text{ fois})$$

```
resultat = x

while y > 1:
    resultat = resultat * x
    y = y - 1

print(resultat)
```

Et si y est négatif ou nul ?



Exemple 2 : calculer x^y

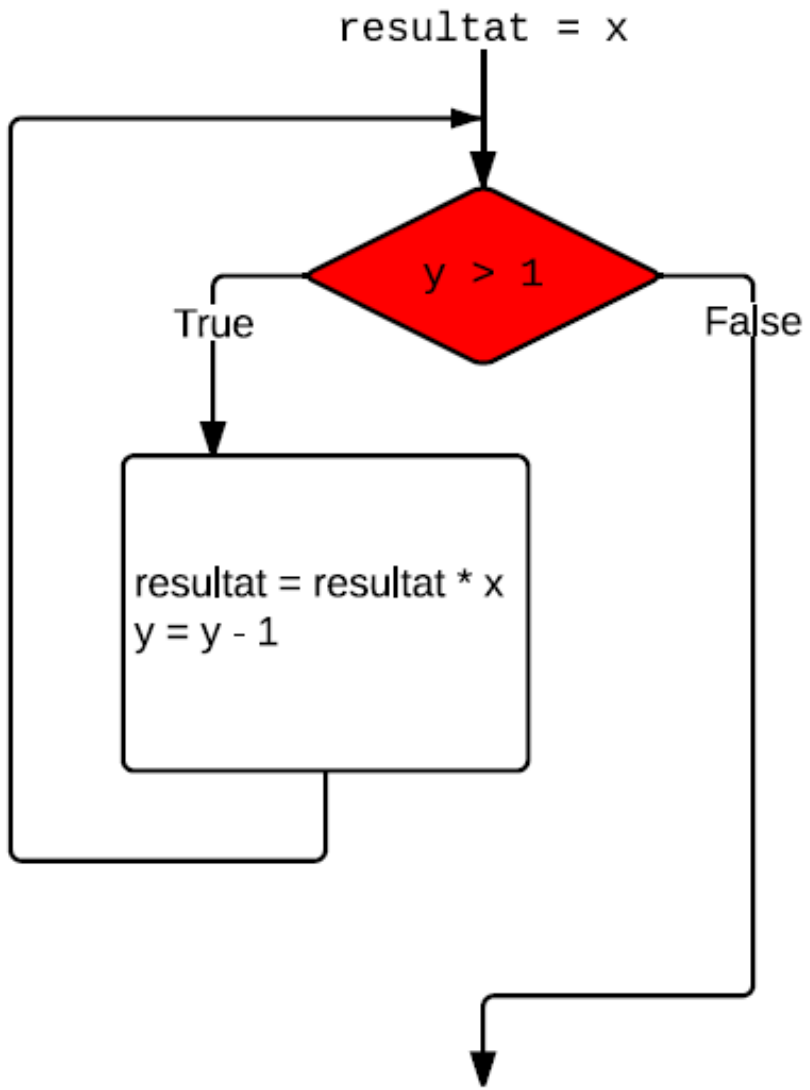
$$x^y = x \cdot x \cdot x \cdot \ldots \cdot x \quad (y - 1 \text{ fois})$$

```
resultat = x

while y > 1:
    resultat = resultat * x
    y = y - 1

print(resultat)
```

Comment modifier le code pour inclure le cas de `y = 0` ?



Exemple 2 : calculer x^y

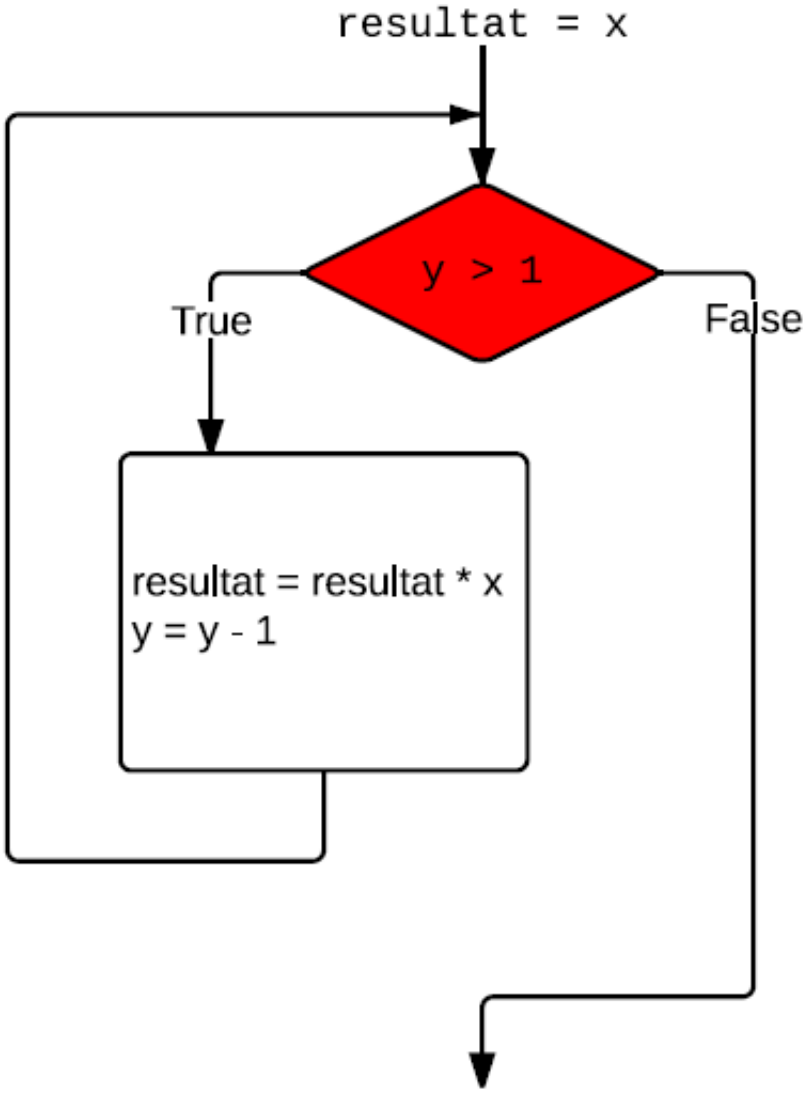
$$x^y = x \cdot x \cdot x \cdot \ldots \cdot x \quad (y - 1 \text{ fois})$$

```
# On suppose y >= 0

if y == 0:
  resultat = 1

else:
  resultat = x
  while y > 1:
    resultat = resultat * x
    y = y - 1

print(resultat)
```



Exemple 2: calculer x^y

Exemplewhile_xpuissancey.py - C:/Exemples/Exemplewhile_xpuissancey.py (3.7.0)

File Edit Format Run Options Window Help

```
x = int(input("Entrer la valeur de x: "))
y = int(input("Entrer la valeur de y: "))
# on suppose y >= 0

if y == 0:
        resultat = 1
else:
        resultat = x
        while y > 1:
                resultat = resultat * x
                y = y -1
print("le résultat de x puissance y est:", resultat)
```

Ln: 4 Col: 0

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
============== RESTART: C:/Exemples/Exemplewhile_xpuissancey.py ==============
Entrer la valeur de x: 4
Entrer la valeur de y: 2
le résultat de x puissance y est: 16
>>>
```

Ln: 8 Col: 4

Exemple 3 : Boucles while imbriquées

- Exemple d'utilisation : calculer une table de multiplication [1 à x] par [1 à y], où x et y sont deux paramètres tels que x >=1 et y >= 1.

1	1	2	3	4	5	...	y
1	1	2	3	4	5	...	y
2	2	4	6	8	10	...	2 · y
3	3	6	9	12	15	...	3 · y
...	...	...	...	...	...	...	...
x	x	x · 2	x · 3	x · 4	x · 5	...	x · y

Exemple 3 : Boucles while imbriquées

- Exemple d'utilisation : calculer une table de multiplication [1 à x] par [1 à y], où x et y sont deux paramètres tels que x >=1 et y >= 1.

```
# On suppose que x >= 1 et y >= 1

multiplicateur = 1
while multiplicateur <= x:  # création d'une ligne
    print("Les multiples de ", multiplicateur,
          " sont : ", end="")

    ...

    multiplicateur = multiplicateur + 1  # on passe à la
                                         # ligne suivante
                                         # du tableau
```

Exemple 3 : Boucles while imbriquées

- Exemple d'utilisation : calculer une table de multiplication [1 à x] par [1 à y], où x et y sont deux paramètres tels que x >=1 et y >= 1.

```
# On suppose que x >= 1 et x >= 1

multiplicateur = 1
while multiplicateur <= x:  # création d'une ligne
    print("Les multiples de ", multiplicateur,
          " sont : ", end="")
    multiplicande = 1

    while multiplicande <= y:
        print(multiplicateur * multiplicande, end=" ")
        multiplicande = multiplicande + 1

    print("")
    multiplicateur = multiplicateur + 1  # on passe à la
                                         # ligne suivante
                                         # du tableau
```

Exemple 3 : Boucles while imbriquées

Exemple_WhileMultiplicateurs.py - C:\Exemples\Exemple_WhileMultiplicateurs.py (3.7.0)

File Edit Format Run Options Window Help

```
x = 10
y = 8
multiplicateur = 1
while multiplicateur <= x:
    print("Les multiples de ", multiplicateur, " sont : ", end="")
    multiplicande = 1

    while multiplicande <= y:
        print(multiplicateur * multiplicande, end=" ")
        multiplicande = multiplicande + 1

    print("")
    multiplicateur = multiplicateur + 1
```

Ln: 14 Col: 0

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
========== RESTART: C:\Exemples\Exemple_WhileMultiplicateurs.py ========
====
Les multiples de  1  sont : 1 2 3 4 5 6 7 8
Les multiples de  2  sont : 2 4 6 8 10 12 14 16
Les multiples de  3  sont : 3 6 9 12 15 18 21 24
Les multiples de  4  sont : 4 8 12 16 20 24 28 32
Les multiples de  5  sont : 5 10 15 20 25 30 35 40
Les multiples de  6  sont : 6 12 18 24 30 36 42 48
Les multiples de  7  sont : 7 14 21 28 35 42 49 56
Les multiples de  8  sont : 8 16 24 32 40 48 56 64
Les multiples de  9  sont : 9 18 27 36 45 54 63 72
Les multiples de  10  sont : 10 20 30 40 50 60 70 80
>>>
```

Ln: 15 Col: 4

Simuler la boucle do..while (∄ en Python)

Exemple_SimulateDoWhile.py - C:/Exemples/Exemple_SimulateDoWhile.py (3.7.0)

File Edit Format Run Options Window Help

```
#condition = True
#while condition:
#     # loop body here
#     condition = test_loop_condition()
## end of loop

#Exemple:

test = True
while test:
    nombre = int(input("Entrer un nombre > 0: "))
    test = nombre <= 0
    if test:
        print("Nombre invalide, veuillez recommecer")
print("Nombre correct: ", nombre)
```

Ln: 15 Col: 33

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Entrer un nombre > 0: -1
Nombre invalide, veuillez recommecer
Entrer un nombre > 0: 0
Nombre invalide, veuillez recommecer
Entrer un nombre > 0: 2
Nombre correct:  2
>>>
```

Ln: 362 Col: 4

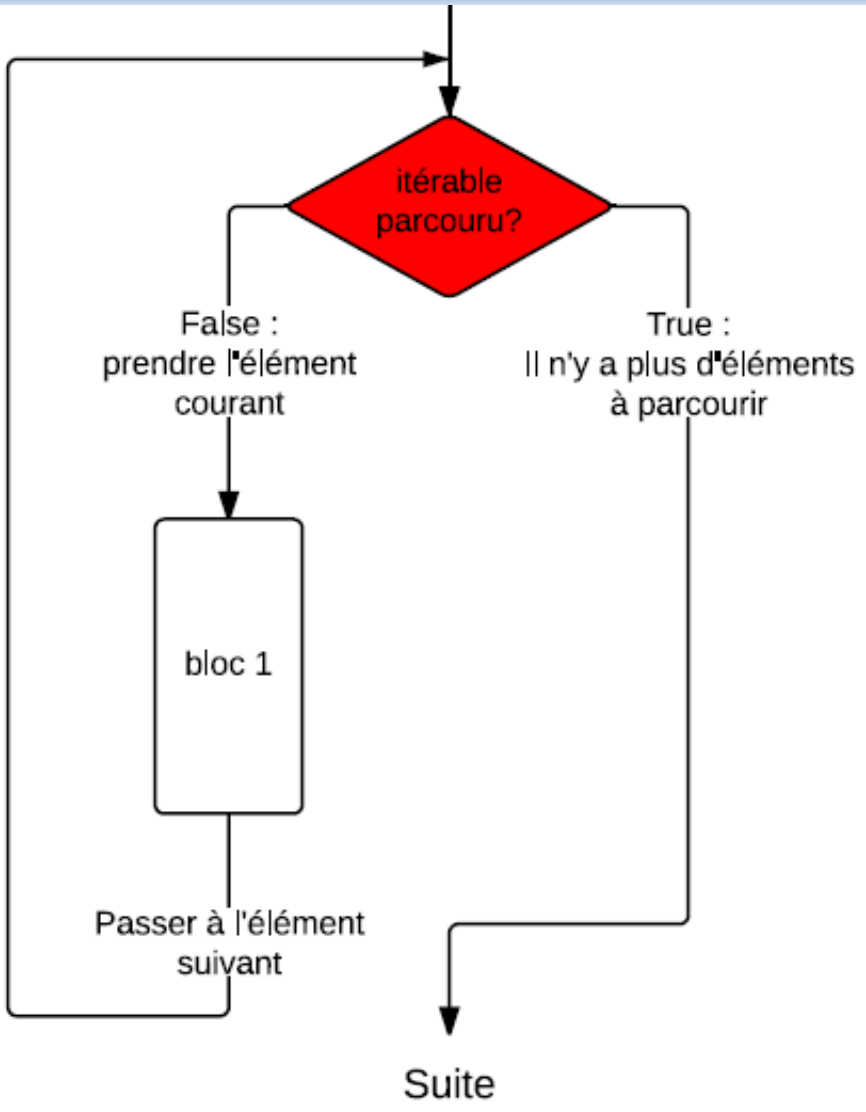
La boucle for

- Pour *cible* dans *itérable* :
  - On exécute le bloc 1
- Suite

Une *liste* est un objet itérable

```
for nb in [1, 2, 3]:
    print(nb)
print("terminé")
```

On y reviendra !



Exemple 1 : for

Exemple_ForMultiplicateurs.py - C:\Exemples\Exemple_ForMultiplicateurs.py (3.7.0)

File Edit Format Run Options Window Help

```
for multiplicateur in [1,2,3,4,5,6,7,8,9,10]:
    print("Les multiples de ", multiplicateur, " sont : ", end="")

    for multiplicande in [1,2,3,4,5,6,7,8]:
        print(multiplicateur * multiplicande, end=" ")

    print("")
```

Ln: 7 Col: 13

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
>>>
============= RESTART: C:\Exemples\Exemple_ForMultiplicateurs.py =============
Les multiples de  1  sont : 1 2 3 4 5 6 7 8
Les multiples de  2  sont : 2 4 6 8 10 12 14 16
Les multiples de  3  sont : 3 6 9 12 15 18 21 24
Les multiples de  4  sont : 4 8 12 16 20 24 28 32
Les multiples de  5  sont : 5 10 15 20 25 30 35 40
Les multiples de  6  sont : 6 12 18 24 30 36 42 48
Les multiples de  7  sont : 7 14 21 28 35 42 49 56
Les multiples de  8  sont : 8 16 24 32 40 48 56 64
Les multiples de  9  sont : 9 18 27 36 45 54 63 72
Les multiples de  10  sont : 10 20 30 40 50 60 70 80
>>>
```

Ln: 27 Col: 4

La boucle for

range([start], stop[, step])

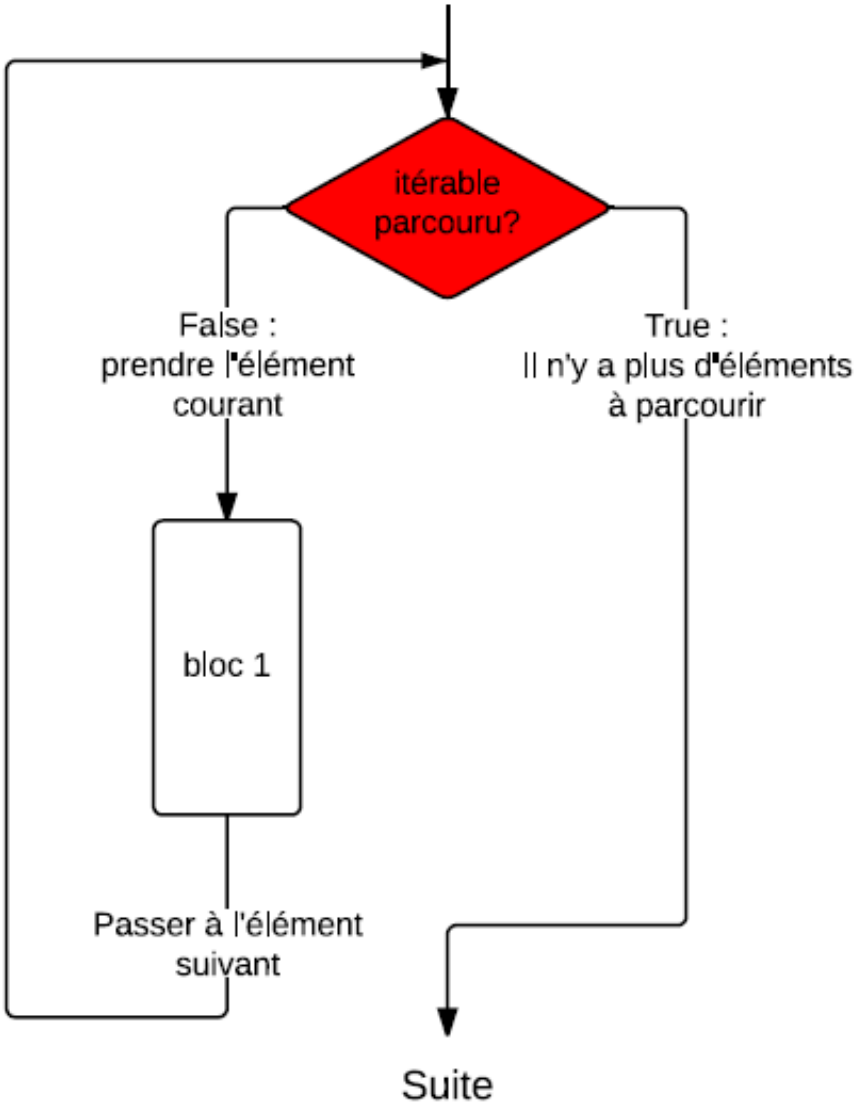
Par défaut:

start = 0

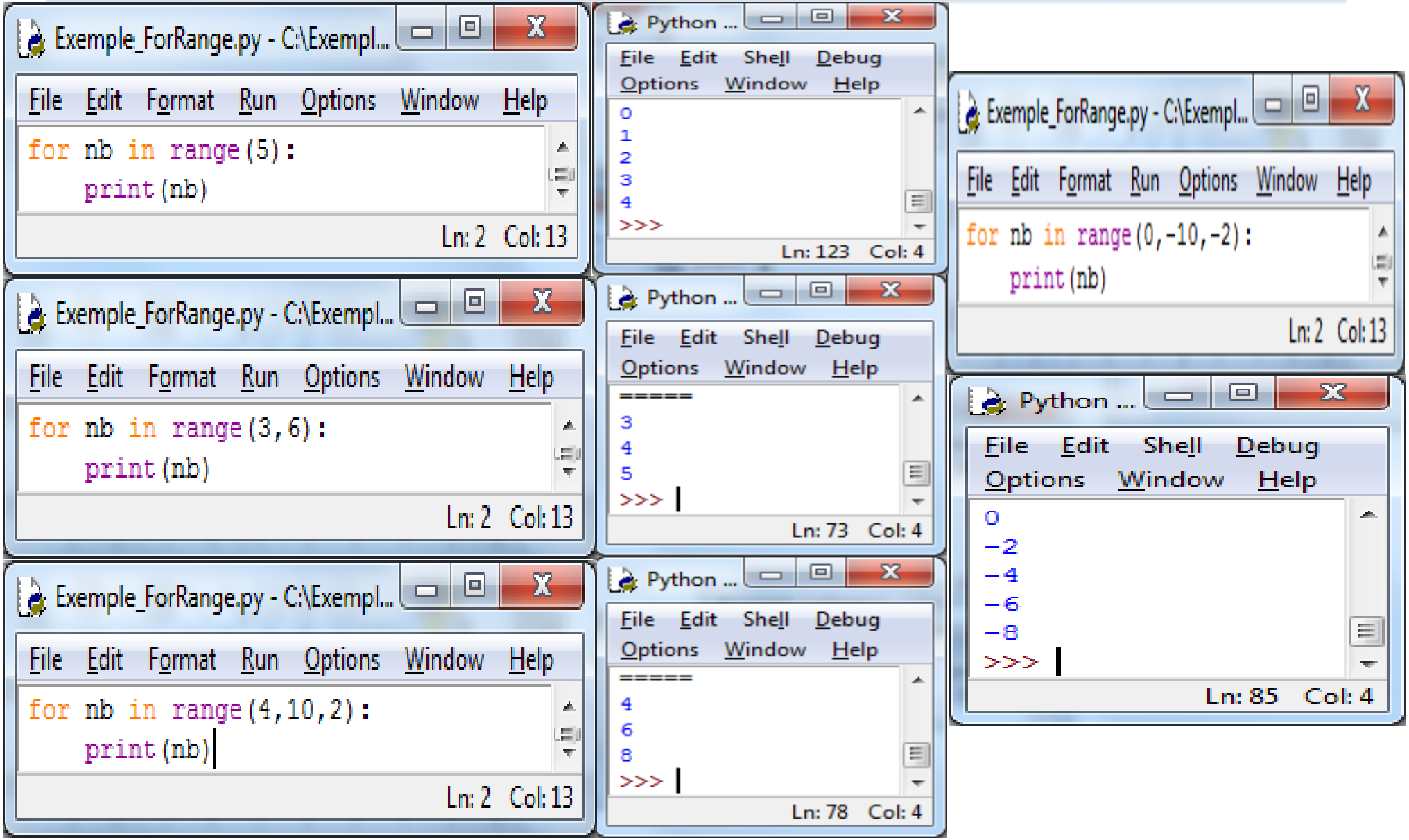
step = 1

```
for nb in range(5):
    print(nb)
```

On y reviendra !



Exemple 2: for



Exemple 3 : for

Exemple_ForMultiplicateursRange.py - C:\Exemples\Exemple_ForMultiplicateursRange.py (3.7.0)

File Edit Format Run Options Window Help

```
x = 10
y =8
for multiplicateur in range(1,x+1):
    print("Les multiples de ", multiplicateur, " sont : ", end="")

    for multiplicande in range(1,y+1):
        print(multiplicateur * multiplicande, end=" ")

    print("")
```

Ln: 9 Col: 13

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
>>>
========== RESTART: C:\Exemples\Exemple_ForMultiplicateursRange.py ==========
Les multiples de  1  sont : 1 2 3 4 5 6 7 8
Les multiples de  2  sont : 2 4 6 8 10 12 14 16
Les multiples de  3  sont : 3 6 9 12 15 18 21 24
Les multiples de  4  sont : 4 8 12 16 20 24 28 32
Les multiples de  5  sont : 5 10 15 20 25 30 35 40
Les multiples de  6  sont : 6 12 18 24 30 36 42 48
Les multiples de  7  sont : 7 14 21 28 35 42 49 56
Les multiples de  8  sont : 8 16 24 32 40 48 56 64
Les multiples de  9  sont : 9 18 27 36 45 54 63 72
Les multiples de  10  sont : 10 20 30 40 50 60 70 80
>>>
```

Ln: 138 Col: 0