TP N 3 Structures conditionnelles et répétitives

Instructions conditionnelles

Avant de présenter les instructions conditionnelles, il faut d'abord connaître les opérateurs de comparaison. Voici un tableau résumant les opérateurs de comparaison et leur signification :

Opérateur	Signification	En python
==	Est égal à (*Equal*)	print(a==b)
!=	Est différent de (*Not equal*)	print(a !=b)
<	Est plus petit que (*less than*)	print(a<b)
≤	Est plus petit que ou égal à (*less or equal*)	print(a<=b)
>	Est plus grand que (*Greater than*)	print(a>b)
≥	Est plus grand que ou égal à (*Greater or equal*)	print(a>=b)

Ces opérateurs permettent de comparer deux valeurs et de renvoyer un résultat vrai ou faux selon la condition évaluée, ce qui est fondamental pour la logique des instructions conditionnelles.

Exemple

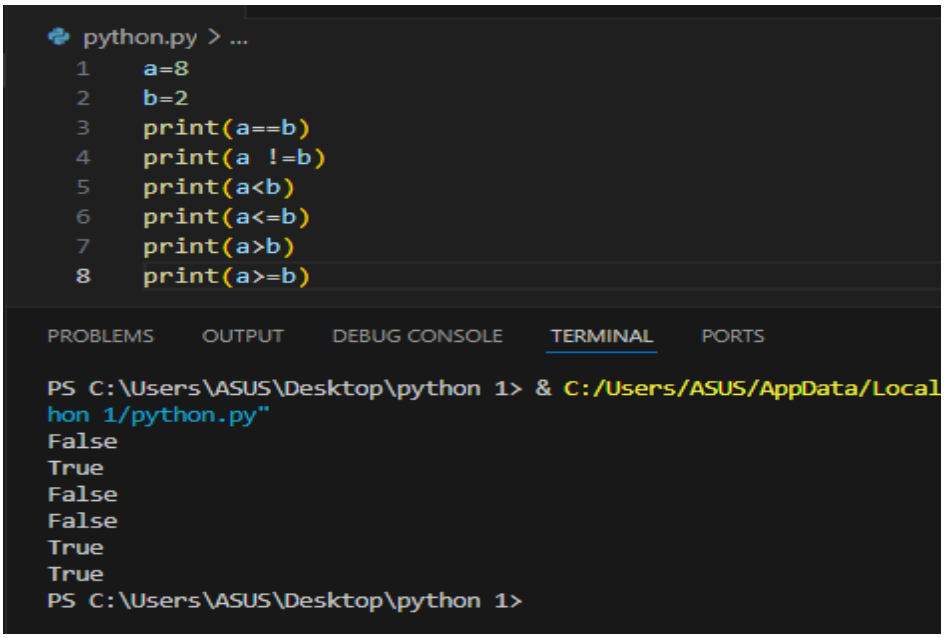


Figure 1 : les opérateurs de comparaison.

Instruction if , elif , else

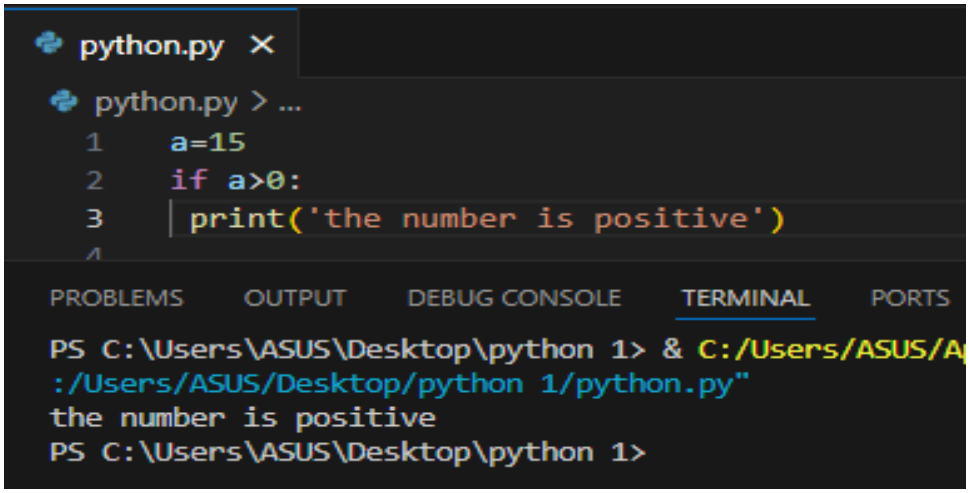
En python, la structure **if**

L'instruction if en Python est une instruction conditionnelle qui exécute un bloc de code uniquement lorsqu'une condition spécifique est satisfaite. Elle évalue une condition (une expression qui résulte en (True ou False), et si la condition est vraie, le bloc de code à l'intérieur de l'instruction if s'exécute.

Syntaxe de l'instruction if

```
If condition :
Les instructions
```

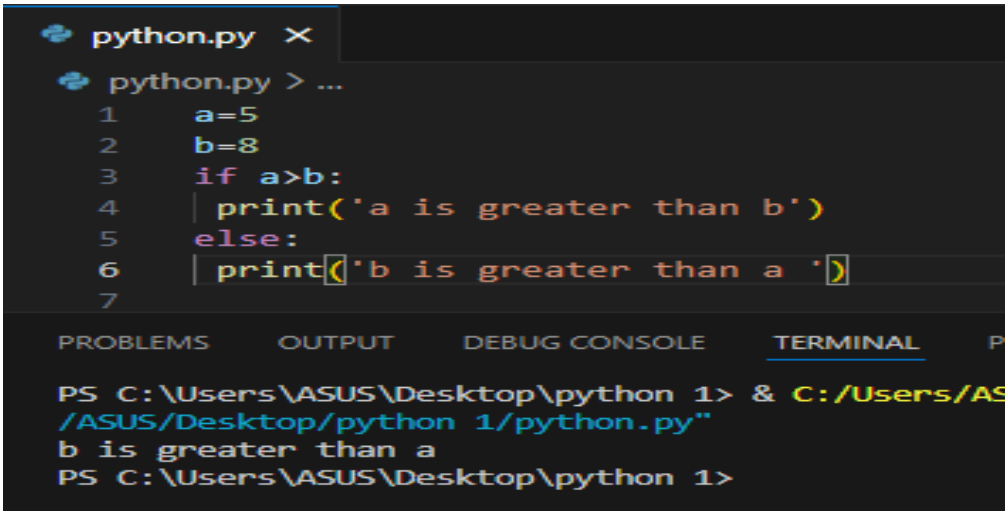
Exemple



L'instruction **if ….else** permet de définir deux chemins d'exécution : un quand la condition est vraie, et un autre quand elle est fausse

Syntaxes de instruction if-else:

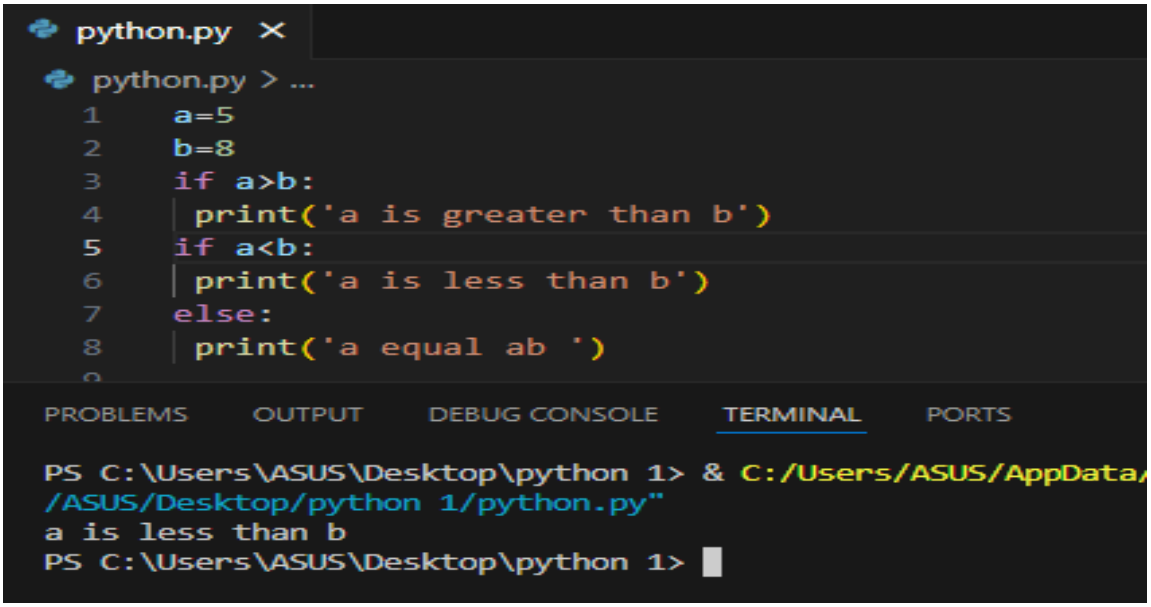
```
If condition:
Les instructions 1 # condition est True
Else :
Les instructions 2 # condition est False
```



Lorsque vous avez plus de deux alternatives, vous utilisez l'instruction **if…elif…else.** L'instruction **elif** permet de vérifier plusieurs conditions successives.

Syntaxes de instruction if-else:

If condition1:
Les instructions 1
If condition 2:
Les instructions 2
If condition 3:
Les instructions 3
Else:
Les instructions 4



Boucle (for, while)

Les boucles répétitives en Python permettent d'exécuter plusieurs fois un bloc d'instructions.

– Boucle **for** est utilisée lorsque le nombre de répétition est connu en avance.

Leur syntaxe est :

For X in ensemble:
Les instructions 1
.
.
Les instructions n

```
python.py
1   for i in range(10):
2       print(i)
3

PROBLEMS  OUTPUT  DEBUG CONSOLE  TER

PS C:\Users\ASUS\Desktop\python 1> & C:
/ASUS/Desktop/python 1/python.py"
0
1
2
3
4
5
6
7
8
9
PS C:\Users\ASUS\Desktop\python 1>
```

```
python.py
1   for i in range(2,10):
2       print(i)
3

PROBLEMS  OUTPUT  DEBUG CONSOLE  TE

PS C:\Users\ASUS\Desktop\python 1> &
/ASUS/Desktop/python 1/python.py"
2
3
4
5
6
7
8
9
PS C:\Users\ASUS\Desktop\python 1>
```

Débit

Fin

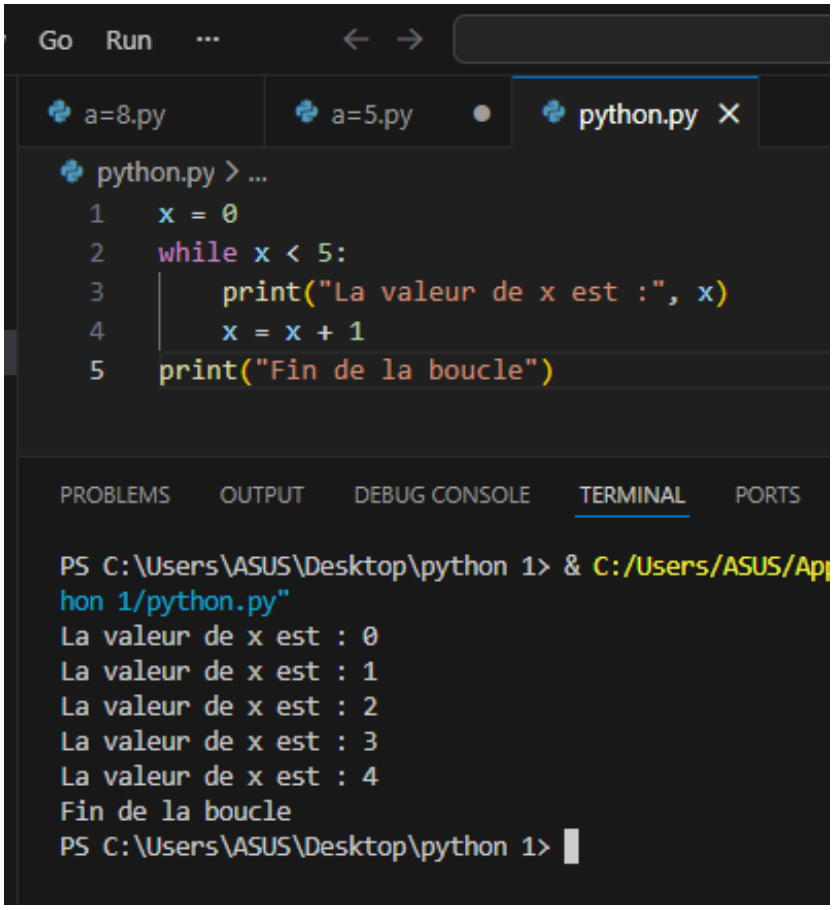
```
python.py
1   for i in range(0,10,2):
2       print(i)
3

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\ASUS\Desktop\python 1> & C:/Users/
/ASUS/Desktop/python 1/python.py"
0
2
4
6
8
PS C:\Users\ASUS\Desktop\python 1>
```

Pas

– La boucle **while** est adaptée lorsque le nombre d'itération n'est pas connu à l'avance et dépend d'une condition, la boucle while répète que la condition est vraie.


Go Run
a=8.py
a=5.py
python.py
python.py > ...
1 x = 0
2 while x < 5:
3 print("La valeur de x est :", x)
4 x = x + 1
5 print("Fin de la boucle")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ASUS\Desktop\python 1> & C:/Users/ASUS/App
hon 1/python.py"
La valeur de x est : 0
La valeur de x est : 1
La valeur de x est : 2
La valeur de x est : 3
La valeur de x est : 4
Fin de la boucle
PS C:\Users\ASUS\Desktop\python 1>