Structures conditionnelles et itératives

1) Structures de contrôle

a) Conditionnelle

Une **conditionnelle** est une **expression booléenne** dont la valeur est soit **True** soit **False**. C'est le résultat d'une comparaison.

b) Opérateurs de comparaison

Les **opérateurs de comparaison** entre deux termes sont :

`<`	`<=`	`>`	`>=`	`==`	`!=`
inférieur	**inférieur ou égal**	**supérieur**	**supérieur ou égal**	**égal**	**différent**

L'égalité entre deux termes se teste avec un "double égal". L'opérateur = est réservé aux affectations.
Les comparaisons s'effectuent entre deux nombres ou entre deux chaînes.

c) Opérateurs logiques

Un **opérateur logique** relie entre elles plusieurs expressions booléennes pour en former une nouvelle.

Il y a deux opérateurs binaires **or** et **and**, qui correspondent aux OU et ET de la logique mathématique, et un opérateur unaire **not**, qui lui correspond au NON.

Table de vérité des opérateurs binaires or et and :

A	B	A or B	A and B
False	False	False	False
False	True	True	False
True	False	True	False
True	True	True	True

2) Structures conditionnelles

a) Test simple : si … alors …

Syntaxe d'un test simple en Python :

```
if conditionnelle:
      bloc d'instructions à exécuter si la conditionnelle est vraie
```

- La ligne les précédant le bloc d'instructions se termine par « : »
- Le bloc d'instruction est indenté par rapport aux instructions qui le précèdent et qui le suivent.

Exemple : remplacer un nombre par sa valeur absolue

```
x = eval(input("Saisir un nombre x : "))
if x < 0:
    x = -x
print("la valeur absolue du nombre est :",x)
```

a) Test avec alternative(s) : si … alors … sinon si …. alors … sinon …

Syntaxe d'un test avec une alternative en Python :

```
if conditionnelle:
      bloc d'instructions à exécuter si la conditionnelle est vrai
else:
      bloc d'instructions à exécuter si la conditionnelle est fausse
```

exemple avec une alternative avec **else** et **if** :

```
n = eval(input("Saisir un nombre entier : "))
if n%2 == 0:
    print("le nombre saisi est pair")
else:
    print("le nombre saisi est impair")
```

exemple avec plusieurs alternatives (« **elif** » est une contraction de « else if ») :

```
x = eval(input("Saisir un nombre : "))
if x = 0:
    print("le nombre est nul")
elif x < 0:
    print("le nombre est strictement négatif")
else:
    print("le nombre est strictement positif")
```

b) Imbrication de tests

```
x = eval(input("Saisir un nombre : "))
if x >= 2:
    if x <= 8:
        print("x est dans l'intervalle [2 ;8].")
    else:
        print("x n'est pas dans l'intervalle [2 ;8].")
else:
    print("x n'est pas dans l'intervalle.")
```

c'est assez lourd, dans ce cas il est préférable d'écrire :

```
x = eval(input("Saisir un nombre : "))
if x >= 2 and x <= 8:
    print("x est dans l'intervalle [2 ;8].")
else:
    print("x n'est pas dans l'intervalle.")
```

3) Structures itératives

a) La structure "for"

La structure **for** réalise un nombre d'itérations fixe et connu. Elle utilise une variable dont la valeur va parcourir une certaine plage. Cette plage de valeurs est construite avec la fonction **range**.

Syntaxe de la fonction range :

```
for variable in range(debut,fin,pas):
      bloc d'instructions à exécuter
```

exemple : affichage des n premiers carrés parfaits:

```
n = eval(input("Saisir un entier positif : "))
for i in range(1,n+1):
    print(n**2)
```

affichage les nombres pairs compris entre 20 et 0 sur une seule ligne :

```
for i in range(20,-1,-2):
    print(i, end=" ")
```

b) La structure "while"

La structure **while** permet de répéter un bloc d'instructions tant qu'une condition est vraie.

La structure while réalise un nombre d'itérations non nécessairement connu.

Ce nombre dépend d'une conditionnelle qui est évaluée avant chaque itération.

Exemple 1 : afficher n fois coucou (version 1)

```
n = eval(input("Saisir un entier positif : "))
while n>0:
    print("coucou")
    n=n-1
```

Remarque : sans la ligne `n=n-1` la boucle ne s'arrête jamais.

Exemple 2 : afficher n fois coucou (version 2)

```
continuer = True
n = eval(input("Saisir un entier positif : "))
while continuer :
    if n > 0 :
        print("coucou")
        n = n-1
    else :
        continuer = False
```

Exemple 3 : calcul de la somme des n premiers entiers

```
n = -1
while n < 0:
    n = eval(input("Saisir un entier positif : "))
somme = 0
for i in range(n+1):
    somme += i
print("la somme des",n,"premiers entiers vaut",somme)
```