

PRÉSENTATION DES INSTRUCTIONS CONDITIONNELLES

► **L'instruction if.** L'instruction **if** permet d'exécuter une suite, ou **bloc**, d'instructions seulement si (en anglais if) une condition est vérifiée. La syntaxe est la suivante

```
if condition :
    bloc_d_instructions
```

Exemple

```
# Les deux lignes ci-dessous sont écrites dans l'éditeur de pyzo
if 2 == 6 - 4 :
    print('toto')
```

Le résultat obtenu est :

```
# Exécution dans le shell par clique droit sur l'onglet de l'éditeur en choisissant <<Exécuter le fichier>>
>>>(executing lines 1 to 2 of "<tmp 1>")
toto
```

Quelques commentaires

- Si la condition est vérifiée alors on réalise le bloc_d_instructions ; sinon, l'ordinateur passe à la suite du programme sans effectuer le bloc_d_instructions.

- Vous remarquez qu'après la condition le bloc_d_instructions à effectuer a été décalée on parle d'**indentation**, ceci est pour faciliter la lecture du programme et identifier dès la première lecture les instructions situées dans la condition et hors la condition. Ce décalage n'a pas simplement un objectif esthétique, il influe également sur l'exécution des instructions.

Exemple 1

```
if 2 == 6 - 3 :
    print('toto') ; print('titi')
```

Le résultat obtenu est vide comme prévu, la condition n'étant pas vérifiée.

Remarque. Pour séparer des instructions dans un bloc d'instructions à effectuer on a deux possibilités : soit on les sépare avec des points-virgule soit on va à la ligne.

Exemple 2

```
if 2 == 6 - 3 :
    print('toto')
print('titi')
```

Le résultat obtenu est :

```
titi    # La condition n'est pas vérifiée mais l'instruction print('titi') est effectuée
        # car elle est hors de la structure conditionnelle comme en témoigne l'indentation
```

Remarque. La condition à tester peut prendre différentes formes :

```
a == b           # Est-ce que a est égal à b ?
a != b           # Est-ce que a est différent de b ?
a > b            # Est-ce que a est strictement supérieur à b ?
a >= b           # Est-ce que a est supérieur ou égal à b ?
a < b            # Est-ce que a est strictement inférieur à b ?
a <= b           # Est-ce que a est inférieur à b ?
a in objet       # Est-ce que a appartient à objet (une chaîne de caractères, une liste)
```

► **L'instruction `if ... else`.** On peut enrichir la syntaxe des tests pour permettre des tests avec alternative. On utilise L'instruction `if ... else` qui permet d'exécuter un bloc d'instructions seulement si (en anglais `if`) une condition est vérifiée et autrement (en anglais `else`) un autre bloc d'instructions si la condition n'est pas vérifiée.

La syntaxe est la suivante

```
if condition :
    bloc_d_instructions
else :
    autre_bloc_d_instructions
```

Quelques commentaires

- Si la condition est vérifiée alors on réalise uniquement toutes les instructions placées avant `else`, notées `bloc_d_instructions`, et si la condition n'est pas vérifiée, on réalise toutes les instructions placées après `else`, notées `autre_bloc_d_instructions`.

- Remarquez l'indentation de l'instruction `else`.

► **L'instruction `if ... elif`.** On peut enrichir la syntaxe des tests pour permettre des tests qui se décompose en plus de deux cas. On utilise L'instruction `if ... elif` qui permet d'exécuter un bloc d'instructions seulement si une `condition_1` est vérifiée, un autre bloc d'instructions si une `condition_2` est vérifiée et encore un autre bloc d'instructions si aucune des deux conditions n'est vérifiée.

La syntaxe est la suivante

```
if condition_1 :
    bloc_d_instructions
elif condition_2 :
    autre_bloc_d_instructions
else :
    encore_une_autre_bloc_d_instructions
```

Quelques commentaires.

- Si la condition 1 est vérifiée alors on réalise uniquement toutes les instructions placées avant `elif`; si la condition 1 n'est pas vérifiée et la condition 2 est vérifiée, on réalise toutes les instructions placées après `elif` et avant `else`; si enfin la condition 2 n'est pas vérifiée, alors on réalise toutes les instructions placées après `else`.

- On peut bien entendu ajouter des instructions `elif` suivant le nombre de cas à étudier.

Exemple

```
primaire =[ 'CP', 'CE1', 'CE2', 'CM1', 'CM2' ]
college =[ '6eme', '5eme', '4eme', '3eme' ]
lycee =[ '2nde', 'Premiere', 'Terminale' ]
if classe in primaire :
    print( 'l élève est en primaire' )
elif classe in college :
    print( 'l élève est au collège' )
elif classe in lycee :
    print( 'l élève est au lycée' )
else :
    print( 'Ce niveau n est pas pris en compte' )
```

EXERCICES

EXERCICE 1. — Le B-A-BA. Ecrire un programme demandant à l'utilisateur de saisir une lettre au clavier, et de retourner "VOYELLE" si la lettre saisie en est une, et "CONSONNE" si c'est une consonne.

On pourra partir du principe que l'utilisateur est bien intentionné, et qu'il va effectivement bien saisir une lettre (pas n'importe quel caractère au clavier).

EXERCICE 2. — Solide, liquide, gazeux

Ecrire un programme demandant à l'utilisateur de saisir une température T en degrés Celsius, et qui affiche l'état de l'eau à la température T (et à pression normale).

EXERCICE 3. — if, elif, else...

1/ Coder en Python le programme ci-dessous écrit en langage naturel :

Demander à l'utilisateur de saisir sa moyenne au bac et affecter cette note à la variable NOTE.
si la note est strictement inférieure à 10, afficher un message d'échec au bac.
sinon, si la note est strictement inférieure à 12, afficher que le bac est obtenu sans mention.
sinon, si la note est strictement inférieure à 14, afficher que la mention obtenue est AB.
sinon, si la note est strictement inférieure à 16, afficher que la mention obtenue est B.
sinon, afficher que la mention est TB.

2/ Quel sera l'affichage renvoyé par l'ordinateur lors de l'exécution du programme si la note entrée est 8 ? 13 ? 17 ?

3/ Dans le programme précédent, si on choisit de remplacer tous les **elif** par des **if**, quel serait l'affichage renvoyé par l'ordinateur lors de l'exécution du programme si la note entrée est 8 ? 13 ? 17 ?

EXERCICE 4. — Division euclidienne

1/ Proposer un programme qui demande à l'utilisateur de saisir un entier N et un entier naturel non nul k , puis qui précise si N est divisible par k .

2/ Proposer un programme qui demande à l'utilisateur de saisir un entier N et qui précise si cet entier est pair ou impair.

EXERCICE 5. — Triangles

Ecrire un programme demandant à l'utilisateur de donner les longueurs des trois côtés AB, BC et CA d'un triangle ABC. Le programme doit d'abord préciser si le triangle ABC existe ; il précise enfin si ce triangle est rectangle.

EXERCICE 6. — Un peu d’optique

On considère un rayon lumineux incident atteignant un dioptre séparant un milieu d’indice n_1 et un milieu d’indice n_2 .

Proposer un programme réalisant les actions suivantes :

Demander à l’utilisateur de saisir la valeur de l’angle d’incidence en radian.

Préciser à l’utilisateur si on est dans un cas où il peut y avoir réflexion totale ou non.

Dans le cas où il ne peut y avoir réflexion totale, l’ordinateur doit indiquer la valeur de l’angle de réfraction.

Dans le cas contraire, l’ordinateur doit préciser si le rayon réfracté existe et, si c’est le cas, il doit donner la valeur de l’angle de réfraction.

EXERCICE 7. — “Delta”

Ecrire un programme qui demande à l’utilisateur de saisir les coefficients réels a , b et c d’un polynôme du second degré (“ $ax^2 + bx + c$ ”), et qui renvoie les racines de ce polynôme (ou la racine lorsque le discriminant est nul).