Informatique – TP3 – Structures conditionnelles

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. — Préciser l'affichage renvoyé par l'ordinateur lorsque on exécute les programmes ci-dessous. On précisera les éventuels problèmes constatés.

```
if 2 ==3 :
        print('AAA')
        print('BBB')
```

```
if 2 !=3 :
        print('AAA')
print('BBB')
```

```
if 2 ==3 :
print('AAA')
        print('BBB')
```

Exercice 2. — **Programme à interpréter**

1/ Expliquer l'objectif du programme ci-dessous :

```
L =[2,3,5,7,11,13,17,19]
print('donner un entier compris entre 1 et 20')
N =input()
if N in L:
        print('OK')
```

2/ Quand on exécute ce programme, on ne constate aucun affichage et ce quel que soit le texte saisi par l'utilisateur. Justifier et rectifier ce programme.

Exercice 3. — **Grand-Petit**. Ecrire un programme demandant à l'utilisateur de saisir un entier au clavier, et de retourner "GRAND" si l'entier est supérieur ou égal à 1000, "PETIT" sinon.

Remarque : on pourra partir du principe que l'utilisateur est bien intentionné, et qu'il va effectivement bien saisir un entier (et pas n'importe quel caractère au clavier).

Exercice 4. — **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.

Exercice 5. — **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 6. — **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 7. — 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 8. — "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).