CHAPITRE 4

Les structures conditionnelles

Thème : Algorithmique

Contenus	Capacités attendues
Mettre en évidence un corpus de constructions élémentaires	Séquences, affectation, conditionnelles, [...]

I. Généralités

Une structure conditionnelle permet d'exécuter une séquence d'instructions seulement dans le cas où une condition est vraie, ou dans le cas où une condition est fausse. Si par exemple, une variable est positive, deux textes sont identiques, un booléen est faux, ...

Une **condition** (on dit aussi prédicat) est une expression logique : le résultat de son évaluation est un booléen `True` ou `False`.
Les opérateurs de comparaison sont définis par : $=, \neq, <, >, \leqslant, \geqslant$.

Exemple 1

```
>>> a = 3
>>> b = 2
>>> a > b
True
>>> a == b
False
```

L'expression `a > b` est la condition ; `True` est le résultat de son évaluation.

On peut combiner plusieurs conditions avec les opérateurs logiques `and` et `or`.
On peut tester l'inverse d'une condition avec l'opérateur `not`.

Exemple 2

```
>>> a = 3
>>> b = 2
>>> a > b
True
>>> a not > b
False
```

On peut comparer des entiers entre eux, des réels entre eux ou des caractères entre eux. Dans ce cas, c'est l'ordre lexicographique (celui du dictionnaire) qui est pris en compte.

Exemple 3

```
>>> 'ane' < 'boa'
True
```

On peut aussi tester l'appartenance d'un élément à un autre avec le mot clé `in`.

Exemple 4

```
>>> 'a' in 'ane'
True
>>> 'b' in 'ane'
False
```

II. L'instruction conditionnelle simple

C'est la structure la plus simple. Elle définit un bloc d'instructions qui seront exécutées si la condition est vraie.

II. 1. Structure algorithmique

```
début
    si condition est vraie alors
        bloc a exécuter
        si la condition est vraie
    fin si
    Instructions qui seront TOUJOURS exécutées
fin
```

Exemple 5

Algorithme : est_positif

```
/* algorithme qui indique si un nombre est positif                    */
Entrées : x : un nombre
Sorties : y : un nombre
/* Variables                                                          */
/* x, y : des nombres                                                 */

1 début
2 |  si x > 0 alors
3 |  |  afficher "ce nombre est positif"
4 |  fin si
5 |  y ← 2 × x
6 |  renvoyer y
7 fin
```

La ligne 3 sera exécutée uniquement si $x > 0$.
Les lignes 5 et 6 seront exécutées dans tous les cas.

Lorsque x prend une valeur négative ou nulle, la ligne 3 n'est pas exécutée.

II. 2. Implémentation en python

Un instruction conditionnelle simple est implémentée en Python par l'instruction `if`.

```
1 if ma_condition :
2     bloc d'instructions
```

Remarques.

- La ligne commençant par le mot clef `if` doit être terminée par un `:` ;
- les instructions à exécuter si la condition est vraie doivent être indentées.

Exemple 6

```
1 def est_positif(x):
2     if x > 0 :
3         print("ce nombre est positif")
4     y = 2*x
5     return y
```

III. L'instruction conditionnelle avec une seule alternative

L'instruction conditionnelle avec alternative permet d'effectuer un bloc d'instructions si un test est vrai, et un autre bloc d'instructions si ce test est faux.

III. 1. Structure algorithmique

```
1  début
2  |  si condition est vraie alors
3  |  |  bloc a exécuter
4  |  |  si la condition est vraie
5  |  sinon
6  |  |  bloc à exécuter
7  |  |  si la condition est fausse
8  |  fin si
9  |  Instructions qui seront TOUJOURS exécutées
10 fin
```

Exemple 7

Algorithme : est_positif

```
/* algorithme qui indique si un nombre positif                    */
Entrées : x : un nombre
Sorties : y : un nombre
/* Variables                                                      */
/* x, y : des nombres                                             */

1  début
2  |  si x > 0 alors
3  |  |  afficher "ce nombre est positif"
4  |  sinon
5  |  |  afficher "ce nombre est négatif ou nul"
6  |  fin si
7  |  y ← 2 × x
8  |  renvoyer y
9  fin
```

La ligne 3 sera exécutée uniquement si $x > 0$.
La ligne 5 sera exécutée uniquement si $x \leqslant 0$.
Les lignes 7 et 8 seront exécutées dans tous les cas.

Lorsque x prend une valeur négative ou nulle, la ligne 3 n'est pas exécutée.
Lorsque x prend une valeur positive non nulle, la ligne 5 n'est pas exécutée.

III. 2. Implémentation en python

Un instruction conditionnelle avec alternative est implémentée en Python par l'instruction `if ... else`.

```
1  if ma_condition :
2      bloc d'instructions
3  else :
4      bloc d'instructions
```

Remarques.

- La ligne commençant par le mot clef `else` doit être terminée par un : ;
- il est conseillé de ne pas placer de test à la suite du `else`.

Activité 1.
Implémentez l'algorithme précédent

```
def est_positif(x):
    if ...
        ..............



    return y
```

IV. L'instruction conditionnelle avec plusieurs alternatives

L'instruction conditionnelle avec plusiers alternatives permet d'effectuer un bloc d'instructions si le test1 est vrai, un autre bloc d'instructions si le test2 est vrai, et/ou un autre bloc d'instructions si les deux tests précédents sont faux. On peut ajouter autant de conditions que l'on souhaite.

Cette structure permet de minimiser le temps d'exécution.

IV. 1. Structure algorithmique

```
début
    si condition1 est vraie alors
        bloc a exécuter
        si la condition1 est vraie
    sinon si condition2 est vraie alors
        bloc à exécuter
        si la condition2 est vraie
    sinon
        bloc à exécuter
        si la condition1 ET la condition2 sont fausses
    fin si
    Instructions qui seront TOUJOURS exécutées
fin
```

Exemple 8

Algorithme : est_positif

```
/* algorithme qui indique si un nombre positif                    */
Entrées : x : un nombre
Sorties : y : un nombre
/* Variables                                                      */
/* x, y : des nombres                                             */
```

```
1  début
2      si x > 0 alors
3          afficher "ce nombre est positif"
4      sinon si x < 0 alors
5          afficher "ce nombre est négatif"
6      sinon
7          afficher "ce nombre est nul"
8      fin si
9      y ← 2 × x
10     renvoyer y
11 fin
```

Lorsque x prend une valeur positive, les lignes 4 à 8 ne sont pas exécutées.
Lorsque x prend une valeur négative, les lignes 3, 6, 7 et 8 ne sont pas exécutée.
Lorsque x prend une valeur nulle, les lignes 3 et 5 ne sont pas exécutée.
Les lignes 9 et 10 seront exécutées dans tous les cas.

IV. 2. Implémentation en python

Un instruction conditionnelle avec alternative est implémentée en Python par l'instruction `if ... elif... else`.

```
1  if ma_condition1 :
2      bloc d'instructions
3  elif ma_condition2 :
4      bloc d'instructions
5  else :
6      bloc d'instructions
```

Activité 2.
Implémentez l'algorithme précédent

```
1  def est_positif(x):
2      if ...
3          ..............
4
5
6
7
8
9  return y
```

Je retiens

- ▷ Une **structure conditionnelle** permet d'exécuter une séquence d'instructions seulement dans le cas où une condition est vraie ;
- ▷ une **condition** est une expression logique. Le résultat de son évaluation est un booléen `True` ou `False` ;
- ▷ l'implémentation en Python d'une structure conditionnelle est `if ... elif... else` ;
- ▷ les lignes commençant par `if`, `elif` ou `else` se terminent toujours par un double point : ;
- ▷ les blocs d'instructions conditionnelles doivent être indentés (touche tabulation) ;
- ▷ lors de l'exécution, certaines lignes de code ne sont pas exécutées, suivant le résultat de l'évaluation de la condition ;
- ▷ les opérateurs de comparaison sont implémentés en Python par : `==` , `!=` , `<` , `<=` , `>` , `>=` ;
- ▷ on peut combiner plusieurs conditions avec les opérateurs logiques `and` et `or` ;
- ▷ on peut tester l'appartenance avec le mot-clé `in` ;
- ▷ on peut tester l'inverse d'une condition avec l'opérateur `not` ;

V. Exercices

Exercice 1 (QCM).

Que peut-on dire du code suivant ?

```
a = 5
if a < 2
    print('oui')
else
    print('non')
```

- ☐ Ce code affiche 'oui'
- ☐ Ce code est invalide
- ☐ Ce code affiche 'non'
- ☐ Ce code n'affiche aucun message

On a saisi le code suivant :

```
if a < 1 :
    print("TIC")
elif a > 8.5 :
    print("TAC")
else :
    print("TOE")
```

Quel message affiche l'ordinateur lorsque a prend la valeur 8.5 ?

- ☐ TIC
- ☐ TAC
- ☐ TOE
- ☐ Le code n'affiche aucun message

On a saisit le code suivant :

```
if n % 3 == 0 or n % 5 == 0 :
    if n % 3 == 0 :
        resultat = 'A'
    else :
        resultat = 'B'
else :
    if n % 5 == 0 :
        resultat = 'C'
    else :
        resultat = 'D'
```

Quelle valeur prend la variable resultat si n prend la valeur 10 ?

- ☐ 'A'
- ☐ 'B'
- ☐ 'C'
- ☐ 'D'

Exercice 2.

Complétez la fonction *est_entier* qui renvoie `True` si le nombre entré est entier, et `False` sinon.

```
def est_entier(x) :
    ..........
    ..........
    ..........
```

Exercice 3.
Complétez la fonction *est_pair* qui renvoie `True` si le nombre entré est pair, et `False` sinon.

```
def est_pair(x) :
    ..........
    ..........
    ..........
```

Exercice 4.
Complétez la fonction *appartient* qui renvoie `True` si le caractère appartient à la chaîne de caractères, et `False` sinon.

```
def appartient(car, chaine) :
    ..........
    ..........
    ..........
```

Exercice 5.
Un thermostat permet de maintenir à température constante une enceinte. L'utilisateur fixe une consigne de température $Tc = 21°C$.

Lorsque la température passe en dessous de cette consigne moins un écart constant de valeur $dT = 1.5°C$, un dispositif de chauffage se met en marche : Cela correspond à `C = True`.
Lorsque la température passe au dessus de cette consigne plus l'écart constant de valeur dT, le dispositif de chauffage s'arrête : `C = False`.

Complétez la fonction ci-dessous, permettant d'afficher l'état C du dispositif de chauffage, selon cette valeur de température.

```
def controle(T) :
    ..........
    ..........
    return C
```

Exercice 6.
Complétez la fonction *pH* qui renvoie `"acide"` si le pH est inférieur à 7, `"neutre"` si le pH est égal à 7, et `"basique"` si le pH est supérieur à 7.

```
def pH(x) :
    ..........
    ..........
    ..........
```

Exercice 7.

1. Complétez la fonction Python *intervalle* qui retourne `True` si ce réel est dans l'intervalle [0,5] et `False` sinon.

```
def intervalle(x):
    ........
```

2. Complétez la fonction Python *appartient_bis* qui retourne `True` si le caractère est dans chacune des deux chaines de caractères et `False` sinon.

```
def appartient_bis(chaine1, chaine2, caractere):
    ........
```

3. Complétez la fonction Python *appartient_ter* qui retourne `True` si le caractère est dans l'une ou l'autre des deux chaines de caractères et `False` sinon.

```
def appartient_ter(chaine1, chaine2, caractere):
    ........
```

4. On considère la fonction suivante :

```
def delta(A, B, x):
    return (x in A or x in B) and not (x in A and  x in B)
```

On considère que A prend la valeur 'ABCDEFGHIJKLM' et que B prend la valeur 'OKLM'

a. Que retourne la fonction lorsque x prend la valeur 'O' ?

b. Trouver une valeur de x pour que la valeur renvoyée soit `False`.