PYTHON AU LYCÉE (2) : LES INSTRUCTIONS CONDITIONNELLES

COURS

Python au lycée (2) : Les instructions conditionnelles

DURÉE ESTIMÉE

15 minutes

OBJECTIFS DU CHAPITRE

- Écrire un test simple avec if
- Traiter deux cas avec if ... else
- Enchaîner plusieurs cas avec if ... elif ... else
- Combiner des conditions avec and, or, not
- Programmer une fonction définie par morceaux

1. Type booléen et conditions

Type booléen

Dans le chapitre précédent, nous avons vu trois types de variables : entiers, décimaux, chaînes de caractères. Nous allons maintenant étudier un quatrième type de variable : le type **booléen**.

Les variables de type booléen ne peuvent prendre que l'une ou l'autre des 2 valeurs suivantes : *True* (vrai) ou *False* (faux).

> ⚠ **Attention**
>
> Les valeurs *True* et *False* doivent être écrites avec une **initiale en majuscule** pour être reconnues par Python. Par ailleurs, il ne faut pas écrire ces valeurs entre apostrophes car elles seraient alors interprétées comme des chaînes de caractères et non comme des booléens.

Par exemple :

```
>>> type(True)     # affiche class 'bool'
>>> type(true)     # affiche une erreur
>>> type("True")   # affiche class 'str'
```

Rappel :
Les chevrons >>> indiquent que les commandes ont été saisies en mode interactif - sous IDLE par exemple.
Les # indiquent le début d'un commentaire.

Conditions

Le type booléen est généralement utilisé pour indiquer le résultat d'une condition, par exemple pour comparer deux valeurs :

```
>>> a = 3 # la valeur 3 est affectée à la variable a
>>> a > 5 # affiche False (car 3 n'est pas supérieur à 5)
>>> a < 5 # affiche True (car 3 est inférieur à 5)
```

Les principaux opérateurs de comparaison que l'on peut utiliser en Python sont les suivants :

Op.	Description
==	égal à
!=	différent de
<	strictement inférieur à
>	strictement supérieur à
<=	inférieur ou égal à
>=	supérieur ou égal à

> ⚠ **Attention**
>
> Pour tester l'égalité de deux valeurs, il faut utiliser l'opérateur == (double signe égal) et non =. En effet, le symbole = simple est l'opérateur d'affectation ; son utilisation à l'intérieur d'une condition provoquera une erreur de Python.

Ces opérateurs permettent de comparer deux variables de type numérique (entier décimal) mais également deux variables de type « chaîne de caractères ». Dans ce cas, les variables seront classées par ordre alphabétique.

Par exemple :

```
>>> 1 <= 3 # affiche True
>>> 1!= 3 # affiche True
>>> 1 == 3 # affiche False
```

```
>>> 1 = 1 # affiche une erreur (il faut utiliser ==)
>>> "a" < "b" # affiche True
```

Bien sûr, il est généralement plus intéressant d'utiliser ces opérateurs avec des variables :

```
>>> x = 1 # affecte la valeur 1 à la variable x
>>> y = 6 # affecte la valeur 6 à la variable y
>>> x!= y # affiche True
>>> x == y # affiche False
>>> x <= y # affiche True
>>> x+5 == y # affiche True
```

Conditions composées

Il est possible de modifier ou de relier différentes conditions à l'aide des opérateurs suivants : *not*, *or*, *and*.

- **not** : inverse le résultat de la condition ; *not condition* est vraie si et seulement si *condition* est fausse.
- **or** : *condition1 or condition2* est vraie si et seulement si *condition1* est vraie ou si *condition2* est vraie ou si les 2 conditions sont vraies simultanément.
- **and** : *condition1 and condition2* est vraie si et seulement si les 2 conditions sont vraies simultanément.

Par exemple :

```
>>> x=1 # affecte la valeur 1 à la variable x
>>> y=2 # affecte la valeur 2 à la variable y
>>> not x==1 # affiche False
>>> not y==1 # affiche True
>>> x==1 or y==1 # affiche True (car la première condition est
```

```
>>> x==2 or y==3 # affiche False
>>> x==1 and y==1 # affiche False (car la seconde condition n'es
>>> x==1 and y==2 # affiche True
```

Remarque

Contrairement à d'autres langages, en Python, il n'est pas nécessaire de placer les conditions entre parenthèses.

2. Instructions conditionnelles

Test « if »

En Python, un test est introduit par le mot-clé « *if* » (si) suivi d'une condition qui peut être vraie ou fausse. Le bloc d'instructions suivant (appelé bloc d'*instructions conditionnelles*) sera exécuté si et seulement si la condition est vraie :

```
if condition :
    # ces instructions (indentées) seront exécutées
    # si et seulement si la condition est vraie
## les instructions placées ensuite, non indentées,
## seront exécutées dans tous les cas
```

Remarque

Deux choses sont importantes à noter :

- La condition doit toujours être suivie du caractère « : » qui introduit le bloc d'instructions conditionnelles.

- c'est l'**indentation** (décalage des instructions vers la droite) qui définit le début et la fin du bloc d'instructions conditionnelles

Exemple

```
nom = input("Saisir votre nom : ")
if nom=="Henri" :
    print("Bonjour Henri")
    print("Comment allez-vous?")
print("Bienvenue sur maths-cours.fr!")
```

Le programme ci-dessus demandera un nom d'utilisateur.
Puis il affichera :
Bonjour Henri
Comment allez-vous ?
Bienvenue sur maths-cours.fr ! si le nom entré est Henri ;

tandis qu'il affichera uniquement :
Bienvenue sur maths-cours.fr ! si un autre nom est saisi.

(pour les instructions *input* et *print* voir le chapitre précédent).

Test « if - else »

Il est possible de faire suivre une condition « if » par une condition « else » (sinon) ; le bloc d'instructions qui suit *else* sera alors exécuté si et seulement si la condition suivant le « if » est fausse :

```
if condition :
    # ces instructions seront exécutées
    # si la condition est vraie
else :
    # ces instructions seront exécutées
```

```
    # si la condition est fausse
## les instructions placées ensuite, non indentées,
## seront exécutées dans tous les cas
```

Remarque

Notez, là aussi, la présence du symbole « : » après le mot *else* ainsi que l'**indentation**.

Exemple

```
mdp = input("Saisir votre mot de passe : ")
if mdp=="12345678" :
    print("Bienvenue!")
else :
    print("Mot de passe incorrect!")
```

Ce programme demandera un mot de passe.
Puis il affichera :
Bienvenue ! si le mot de passe entré est 12345678 ;

alors qu'il affichera :
Mot de passe incorrect ! si un autre mot de passe a été entré.

Test « if - elif - else »

Lorsque le test comporte plusieurs conditions, on peut utiliser l'instruction « *elif* » qui est l'abréviation de *else if*. Cette instruction s'utilise de la manière suivante :

```
if condition1 :
    # instructions exécutées si condition1 est vraie
elif condition2 :
```

```
    # instructions exécutées si condition1 est fausse
    # et condition2 est vraie
elif condition3 :
    # instructions exécutées si condition1 et condition2
    # sont fausses et condition3 est vraie
else :
    # instructions exécutées si toutes les conditions
    # précédentes sont fausses
## les instructions placées ensuite, non indentées,
## seront exécutées dans tous les cas
```

Dans ce cas, on peut placer autant de conditions que l'on veut.
Par exemple :

```
erreurs = int(input(" Entrez le nombre d'erreurs :  "))
if erreurs == 0 :
    print(" Il n'y a aucune erreur!")
elif erreurs == 1 :
    print(" Il y a une seule erreur!")
else :
    print(" Il y a ", erreurs, " erreurs!")
```

Ce programme demande puis affiche le nombre d'erreurs présentes dans un texte.

Application : fonction définie par morceaux

Les instructions conditionnelles sont particulièrement utiles pour programmer une fonction mathématique définie par morceaux. Par exemple, soit *f* définie par :

$$f(x) = -x + 1 \text{ si } x \leq 0, \text{ et } f(x) = 3x + 1 \text{ si } x > 0$$

On peut calculer les images par *f* grâce au programme suivant :

```
x = float(input("Saisir un nombre x : "))
if x <= 0 :
    y = -x + 1
else :
    y = 3*x + 1
print("f(", x, ") =", y)
```

Condition stockée dans une variable booléenne

Le résultat d'une comparaison est une valeur booléenne (True ou False) que l'on peut stocker dans une variable. On peut ensuite utiliser cette variable comme condition d'un test :

```
x = 7
est_positif = (x > 0)     # est_positif vaut True
print(type(est_positif))  # affiche class 'bool'
if est_positif :
    print("x est strictement positif")
else :
    print("x est négatif ou nul")
```

Les questions essentielles

1. Comment écrire un test simple avec if ?

On écrit *if condition :* suivi d'un bloc indenté. Les instructions du bloc ne sont exécutées que si la condition est vraie. Ne pas oublier les deux-points et l'indentation.

Voir la fiche méthode : Écrire un test simple avec if

2. Comment traiter deux cas avec if ... else ?

Après le bloc *if*, on ajoute *else :* suivi d'un bloc indenté. Ce second bloc est exécuté si la condition du *if* est fausse.

Voir la fiche méthode : Traiter deux cas avec if ... else

3. Comment enchaîner plusieurs cas avec if ... elif ... else ?

On utilise *elif condition :* après le *if* initial, autant de fois que nécessaire, puis un *else* final. Python teste les conditions dans l'ordre et s'arrête à la première vraie.

Voir la fiche méthode : Enchaîner plusieurs cas avec if ... elif ... else

4. Comment combiner plusieurs conditions avec and, or et not ?

and (« et ») exige que les deux conditions soient vraies. *or* (« ou ») accepte que l'une au moins soit vraie. *not* (« non ») inverse une condition.

Voir la fiche méthode : Combiner des conditions avec and, or, not

5. Comment programmer une fonction définie par morceaux ?

On utilise une structure *if ... elif ... else* avec une branche par morceau, en reprenant les conditions du plus restrictif au plus général.

Voir la fiche méthode : Programmer une fonction définie par morceaux

Télécharger en PDF