Une forme alternative du `if`

Complément - niveau basique

Expressions et instructions

Même si cela n'a pas été "sous-titré" jusqu'à présent, vous aurez peut-être remarqué que les constructions python que nous avons vues jusqu'ici peuvent se ranger en deux familles.

D'une part les **expressions** sont les fragments de code qui **retournent une valeur**; c'est le cas lorsqu'on invoque n'importe quel opérateur numérique ou pour les appels de fonctions.

D'autre part les **instructions**; dans cette famille, jusqu'ici nous avons vu `print`, l'affectation, et `if`, et nous en verrons bien d'autres.

La différence essentielle est que les expressions peuvent être combinées entre elles pour faire des expressions arbitrairement grosses. Aussi si vous avez un doute pour savoir si vous avez affaire à une expression ou à une instruction, demandez vous si vous pourriez utiliser ce code **comme membre droit d'une affectation**. Si oui, vous avez une expression.

`if` est une instruction

La forme du `if` qui vous a été présentée pendant la vidéo ne peut donc pas servir à renvoyer une valeur. Nous allons voir ce que cela implique pour écrire quelque chose d'aussi simple que *"affecter à y la valeur 12 ou 35, selon que x est vrai ou non"*.

En python donc, `if` n'est pas une expression, donc on ne peut pas l'utiliser pour écrire quelque chose comme :

```
# ceci n'est pas du python
```

y = (if x : 12 ; else 35)

Avec les notions introduites jusqu'ici, il nous faudrait écrire ceci:

```
x = True
if x:
    y = 12
else:
    y = 35
print y
```

Expression conditionnelle

On trouve dans de nombreux langages l'évaluation conditionnelle comme une expression; par exemple en C, pour ceux qui connaissent, avec la forme

```
y = ( x ? 12 : 35 ) /* version C */
```

Aussi, avec la version 2.5 de python a été introduite la forme dite d'expression conditionnelle, qui est une **expression à part entière**, avec la syntaxe

```
<si_vrai> if <condition> else <si_faux> # version python
```

Ainsi on pourrait écrire l'exemple ci-dessus de manière plus simple et plus concise comme ceci

```
y = 12 if x else 35
print y
```

Nous aurons l'occasion de voir de nombreux exemples où cette construction rend le style de programmation plus fonctionnel et plus fluide.

Complément - niveau intermédiaire

Pour en savoir plus

- La section sur les [expressions conditionnelles] (https://docs.python.org/2/reference/expressions.html (https://docs.python.org/2/reference/expressions.html)

conditional-expressions) de la documentation python

- Le PEP308 (http://legacy.python.org/dev/peps/pep-0308/) qui résume les discussions ayant donné lieu au choix de la syntaxe adoptée

De manière générale, les PEP rassemblent les discussions préalables à toutes les évolutions majeures du langage python.