

L'instruction pass

Complément - niveau basique

Nous avons vu qu'en python les blocs de code sont définis par leur indentation.


Une fonction vide

Cette convention a une limitation lorsqu'on essaie de définir un bloc vide. Voyons par exemple comment on définirait en C une fonction qui ne fait rien

```
 /* une fonction C qui ne fait rien */
```


```
void foo() {}
```

Comme en python on n'a pas d'accolade pour délimiter les blocs de code, il existe en python une instruction `pass`, qui ne fait rien. À l'aide de cette instruction on peut à présent définir une fonction vide comme ceci

```
 # une fonction python qui ne fait rien
def foo():
    pass
```

Une boucle vide

Pour prendre un second exemple un peu plus pratique, et pour anticiper un peu sur l'instruction `while` que nous verrons très bientôt, voici un exemple d'une boucle vide, c'est à dire sans corps, qui permet de "dépiler" dans une liste jusqu'à l'obtention d'une certaine valeur:

```
 liste = range(10)
print 'avant', liste
while liste.pop() != 5:
    pass
print 'avant', liste
```

On voit qu'ici encore l'instruction `pass` a toute son utilité

Complément - niveau intermédiaire

Un if sans then

```
from math import sin
```

Imaginons qu'on parte d'un code hypothétique qui fasse ceci

```
# la version initiale
if sin(0) != 0.:
    print "non"
else:
    print "bingo"
```

et que l'on veuille modifier ce code pour simplement supprimer l'impression de non. La syntaxe du langage ne permet pas de simplement commenter le premier `print`:

```
# la version initiale
if sin(0) != 0.:
    # print "non"
else:
    print "bingo"
```

Évidemment ceci pourrait être réécrit autrement en inversant la condition, mais parfois on s'efforce de limiter au maximum l'impact d'une modification sur le code. Dans ce genre de situation on préférera écrire plutôt

```
# la version initiale
if sin(0) != 0.:
    # print "non"
    pass
else:
    print "bingo"
```

Complément - niveau avancé

Une classe vide

Pour anticiper encore un peu plus sur les classes que nous verrons en cinquième semaine, on peut aussi utiliser `pass` pour définir une classe vide comme ceci

```
class Foo:
    pass

foo = Foo()
```