Algorithmes et Python, Structures

SI → if

```
1 si condition alors
2 |   instruction 1
3 |   instruction 2
4 |   ...
5 fin
6 suite...
```

```
1 if condition:
2     instruction 1
3     instruction 2
4     ...
5 suite...
```

En Python **tous** les blocs de structure ont une déclaration avec **:** en fin de ligne.

En Python, on n'écrit pas **fin** ou **end** pour indiquer la fin d'un bloc. Le début du bloc est annoncé par **:** qui est forcément suivi d'une **indentation** *marge à gauche*. C'est l'indentation qui permet de savoir quand le bloc se termine.

Il n'y a pas de contraintes pour l'indentation : 1 espace, 2, 3...
Vous ferez l'effort de toujours mettre 4 espaces pour un code plus propre.

SI... SINON → if... else

```
1 si condition alors
2 |   instruction 1
3 |   instruction 2
4 |   ...
5 sinon
6 |   instruction a
7 |   instruction b
8 |   ...
9 fin
10 suite...
```

```
1 if condition:
2     instruction 1
3     instruction 2
4     ...
5 else:
6     instruction a
7     instruction b
8     ...
9 suite...
```

TANT QUE... RÉPÉTER → while

```
1 tant que condition répéter
2 |   instruction 1
3 |   instruction 2
4 |   ...
5 fin
6 suite...
```

```
1 while condition:
2     instruction 1
3     instruction 2
4     ...
5 suite...
```

POUR CHAQUE... FAIRE → for... in

```
1 pour chaque élément dans
    ensemble faire
2 |   instruction 1
3 |   instruction 2
4 |   ...
5 fin
6 suite...
```

```
1 for élément in ensemble:
2     instruction 1
3     instruction 2
4     ...
5 suite...
```

Python permet l'utilisation d'accents. `élément` est donc permis. Mais en général on préfère éviter et il est aussi courant d'utiliser des mots en anglais, donc sans accents.

RÉPÉTER n FOIS → for i in range(n)

Il n'y a pas de structure exprès pour `RÉPÉTER n FOIS` mais on peut y arriver avec la boucle `for`.

```
1 répéter n fois
2 |   instruction 1
3 |   instruction 2
4 |   ...
5 fin
6 suite...
```

```
1 for i in range(n):
2     instruction 1
3     instruction 2
4     ...
5 suite...
```

`range` est un outil qui reviendra souvent. Il permet de créer une sorte d'ensemble : `range(n)` crée un ensemble $\{0, 1, 2...n-1\}$. La variable i est utilisée comme un simple compteur.

POUR i ALLANT DE a À b → for i in range(a,b+1)

C'est la forme classique de la boucle `POUR` et dans beaucoup de langages, ce `POUR` se traduirait par un `for`. Mais le `for` de Python en fait un `POUR CHAQUE` traduit par `foreach` dans certains langages.

```
1 pour i allant de a à b faire
2 |   instruction 1
3 |   instruction 2
4 |   ...
5 fin
6 suite...
```

```
1 for i in range(a,b+1):
2     instruction 1
3     instruction 2
4     ...
5 suite...
```

`range(a,b+1)` crée un ensemble $\{a, a+1, ...b\}$.

En Python, quand une commande exprime l'idée « de a à b », b est exclu. Ainsi `range(a,b)` énumère les entiers de a à b, b exclu. Si on veut inclure b, il faut écrire `range(a,b+1)`.