Algorithmique et Programmation

1 Variables et Affectation

Définition : On peut considérer une variable comme une boîte dans laquelle on peut mettre quelque chose : un nombre, une phrase, une liste de mots, etc.
La boîte est ou bien "vide" ou bien elle contient une seule valeur. Si on mets une nouvelle valeur dans la boîte, celle-ci remplace la valeur précédente.

Définition : affecter une valeur à une variable, c'est mettre cette valeur dans la boîte.
On utilise le symbole $\leftarrow$ pour désigner l'affectation

Exemple :

$A \leftarrow 3$	"j'affecte la valeur 3 à la variable A"
$mot \leftarrow$ jean	"j'affecte la valeur "jean" à la variable mot"
$A \leftarrow 7$	"j'affecte la valeur 7 à la variable A"
$B \leftarrow A + 3$	"j'affecte la valeur A+3 à la variable B"

Définition : Un algorithme est une suite **d'instructions** effectuées l'une après l'autre pour aboutir à un résultat.

Exemple : Que fait cet algorithme lorsqu'on saisit la valeur 5 pour x ?

```
Saisir x
y ← x + 1
y ← 2y
y ← y − x
y ← y − 2
Afficher y
```

. . . . . .

En python :

L'affectation en langage Python est effectuée par le symbole "="

Exemple : L'algorithme précédent, écrit en langage Python est

```
input(x)
y = x + 1
y = 2y
y = y − x
y = y − 2
print(y)
```

NB : Pour demander à l'utilisateur de rentrer une valeur, on utilise la commande "**input()**"
Pour afficher quelque chose à l'écran, on utilise la commande "**print()**"

2 Instructions conditionnelles

Définition : Une instruction conditionnelle est une instruction dans un algorithme effectuée seulement si une certaine condition est vérifiée.
Une machine ou une personne qui effectue cet algorithme doit d'abord vérifier si la condition est vraie. Si elle est vraie alors il effectue l'instruction (ou les instructions), sinon il saute la (ou les) ligne(s). On utilise les connecteurs logique **SI** et **ALORS**.

SI *condition*
ALORS *instruction*

Exemple : Que fait cet algorithme ?

```
Saisir x
SI x est pair :
ALORS y ← x/2
SI x est impair :
ALORS  y ← (x + 1)/2
Afficher y
```

On peut utiliser le connecteur logique **SINON** pour donner une (ou plusieurs) instructions à effectuer dans le cas où la condition n'est pas respectée. Par exemple l'algorithme ci-dessus peut être aussi écrit :

```
Saisir x
SI x est pair :
ALORS y ← x/2
SINON y ← (x + 1)/2
Afficher y
```

. . . . . .

En python :

Les instructions conditionnelles en langage Python sont effectuées grâce au connecteurs logiques "**if**" (**SI**) et "**else**" (**SINON**). Il n'y a pas de connecteur logique "**ALORS**". Les instructions à effectuer sont placées selon une *indentation* particulière : il faut placer un espace devant l'instruction à effectuer.

Exemple : L'algorithme précédent écrit en Python est :

```
input(x)
if x%2 == 0:
    y = x/2
else :
    y = (x + 1)/2
print(y)
```

NB: Pour savoir si un nombre est pair, on teste le résultat de sa division euclidienne par 2. En Python, la division euclidienne est effectuée par le symbole % et le test d'égalité par le symbole ==. Dans l'algorithme précédent, la première ligne se traduit donc par "Si le résultat de la division euclidienne de x par 2 est 0 alors :"

3 Boucles

Définition : Une boucle d'instruction est une liste d'instructions à effectuer en boucle selon une certaine règle.
On distingue les boucles bornées ou boucle "for" qui sont des listes d'instructions à effectuer jusqu'à ce que la boucle ait été faite un certain nombre de fois, et les boucles non-bornées, ou boucle "while" qui sont des listes d'instructions à effectuer tant qu'une certaine condition est respectée.

Une boucle "for" doit donc avoir un compteur, qui va compter le nombre de fois que la boucle a été effectuée. Une boucle "while" doit avoir une condition qui va être testée à chaque début de boucle.
On utilise les connecteurs logiques **POUR** / **FIN POUR** et **TANT QUE** / **FIN TANT QUE** pour indiquer le début et la fin d'une boucle "for" (pour) et "while" (tant que).

Exemples : Que font ces deux algorithmes ?

Boucle "for" :

POUR i allant de 1 à 10:
afficher $i * 5$
FIN POUR

Boucle "while" :

$A \leftarrow 0$
TANT QUE $A < 10$:
$A \leftarrow A + 1$
afficher A
FIN TANT QUE

. . . . . .

En python :

En langage Python, les boucles "for" sont effectuées grâce à l'opérateur **for** et les boucles "while" grâce à l'opérateur **while**. Les instructions à effectuer sont indiquées par un alignement après un espace (indentation) donc il n'y a pas "**FIN POUR**" ou "**FIN TANT QUE**"

Exemple: Les algorithmes précédents en Python :

```
for i in range(1,10):
    print(i * 5)
```

```
A = 0
while A < 10 :
    A = A + 1
    print(A)
```

NB: Pour indiquer que le compteur compte de 1 à 10, on écrit "i in *range*(1,10)"

Remarque : Quelle est la différence entre ces deux algorithmes écrit en Python?

```
A = 0
while A < 10 :
    A = A + 1
print(A)
```

```
A = 0
while A < 10 :
    A = A + 1
    print(A)
```