Chapitre 2

Un peu plus d'algorithmes

« En informatique, le problème est souvent entre la chaise et le clavier ».

Une machine ne fait qu'exécuter ce que l'on lui demande, elle n'a pas d'intelligence intrinsèque. Si nous ne sommes pas assez précis dans nos demandes, ou si nous nous trompons, elle n'est pas en mesure de deviner nos attentes ou de corriger nos erreurs.

2.1 Affichage et lecture d'entrée

2.1.1 Affichage et sortie

Une machine possède en général un écran. L'écran est un **périphérique de sortie**. C'est grâce à lui que nous pouvons vérifier que la machine a bien exécuté ce que nous lui demandions. Puisque *« la machine est bête »*, lorsque nous voulons qu'elle affiche un résultat sur le périphérique de sortie, il faut le lui demander.

En algorithmique, nous utiliserons la commande Afficher(truc à afficher). Plus tard dans l'année, lorsque nous utiliserons un logiciel de programmation (Python), la commande sera `print` (imprimer en français).

Exemple :

Algorithme 1 :

1 $c \leftarrow 5$
2 $C \leftarrow c \times c$
3 Afficher(C)
4 Afficher(c)

La machine doit me retourner une information : à l'écran apparaît ..... puis ....

Remarques :

— On peut également directement afficher du texte (sans passer par une variable de type chaîne de caractères. Dans ce cas, il faut utiliser les guillemets " " ; par exemple Afficher("Hello !").

— Pour condenser l'écriture, on peut également afficher plusieurs variables, ainsi que du texte, en une seule ligne. Il faut alors séparer chaque élément par une virgule ; par exemple Afficher("La variable C vaut :",C).

Exemple :

Algorithme 2 :

```
1 c ← 5
2 C ← c × c
3 Afficher(.....)
```

Compléter l'algorithme suivant de sorte qu'en sortie (à l'écran) soit affiché La valeur de C est 25.

2.1.2 Lecture d'entrée

On peut programmer la machine qui nous demande des informations lors de l'exécution d'un programme ; une valeur ou un texte par exemple. En algorithmique, on utilisera la commande Demande(donnée à entrer). En programmation, il faut généralement préciser si nous demandons un entier, un flottant, etc... En Python, la commande en question est `input`.

Exemple :

Algorithme 3 :

```
1 A ← Demande(nombre flottant)
2 B ← A × A
3 Afficher(B)
```

1. Au début de l'algorithme, la machine demande une valeur à l'utilisateur. Si l'on rentre 5, quelle valeur sera affichée en sortie ?
2. Si l'on veut que l'algorithme nous réponde « le carré de A est B », comment écrire la dernière ligne ?

2.2 Test

Pour donner une instruction conditionnelle en algorithmique, on utilise les commandes Si condition vérifiée Alors et la commande Sinon. Comme précédemment, on respecte l'indentation pour signifier quelles sont les instructions à exécuter si la condition est remplie ou non.

Exemple : On considère l'algorithme ci-dessous qui commence par nous demander un entier.

Algorithme 4 :

```
1 N ←Demande(Nombre entier)
2 Si N > 5 Alors :
3         Afficher("Mouette !")
4 Sinon :
5         Afficher(N,"est trop petit")
```

1. Lorsque que l'on entre 7, la condition du Si est ..... et l'algorithme répond .....
2. Lorsque que l'on entre 5, la condition du Si est ..... et l'algorithme répond ..... .

Remarques :

— La condition du test est en fait une variable booléenne.

— Si on a plusieurs conditions à tester simultanément, on peut utiliser Et et Ou : Si condition1 Et condition2 Alors ... ; Si condition1 Ou condition2 Alors ...

2.3 Boucles

En programmation il est souvent intéressant de répéter une action de nombreuses fois. Pour éviter d'avoir à écrire plusieurs lignes de code identiques, on utilise une **boucle**. Celle-ci permet la répétition d'un bloc d'instructions déterminés. Il existe deux types de boucles, les boucles conditionnelles et les boucles inconditionnelles.

2.3.1 Boucle inconditionnelle

Une boucle inconditionnelle est une boucle dans laquelle est prédéterminé le nombre de fois où va se réaliser le bloc d'instructions qu'elle contient. En algorithmique, on pourra utiliser la commande Répéter n fois : où n est un entier naturel : le nombre de répétitions. En Python, les instructions à répéter doivent être décalées vers la droite. On parle **d'indentation**.

Exemple : Les deux programmes donnent le même résultat :

Algorithme 5 :

```
1 Afficher("Bonjour !")
2 Afficher("Comment ça va ?")
3 Afficher("Comment ça va ?")
4 Afficher("Comment ça va ?")
5 Afficher("Au revoir !")
```

Algorithme 6 :

```
1 Afficher("Bonjour !")
2 Répéter 3 fois :
3       Afficher("Comment ça va ?")
4 Afficher("Au revoir !")
```

Exemple : On considère l'algorithme ci-dessous. Il affiche en sortie .....
.....

Algorithme 7 :

```
1 a ← 1
2 Répéter 5 fois :
3       a ← a + a
4 Afficher("La valeur de a est",a)
```

L'instruction Répéter est suffisante pour exécuter plusieurs fois les mêmes commandes mais ce n'est pas celle que l'on utilise dans la réalité car elle est contenue dans une instruction plus générale : Pour. L'instruction Pour permet de faire varier d'éventuelles variables présentes dans les commandes à répéter.

Exemple : Les deux programmes donnent le même résultat :

Algorithme 8 :

```
1 Afficher("Jour 1")
2 Afficher("Jour 2")
3 Afficher("Jour 3")
4 Afficher("Jour 4")
```

Algorithme 9 :

```
1 Pour NuméroDuJour Allant de 1 à 4 :
2       Afficher("Jour", NuméroDuJour)
```

Ici, NuméroDuJour est une variable prenait successivement les valeurs 1, 2, 3 et 4. À chaque fois que l'instruction de la boucle est exécutée, elle l'est avec la nouvelle valeur de NuméroDuJour.

Exemple : Le programme ci-dessous donne :

Algorithme 10 :

```
1 Pour Nombre Allant de 1 à 3 :
2       Afficher(2*Nombre)
```

1. ... ;
2. ... ;
3. ....

Remarque : si la variable du Pour n'apparaît pas dans les instructions répétées, alors on retrouve le même résultat que pour Répéter puisque rien de change.

Algorithme 11 :

```
1 Répéter 3 fois :
2       Afficher("Hello !")
```

Algorithme 12 :

```
1 Pour Numéro Allant de 1 à 3 :
2       Afficher("Hello !")
```

Ces deux algorithmes affichent trois fois de suite « Hello ! ».

2.3.2 Boucle conditionnelle

La boucle conditionnelle est une boucle pour laquelle on ne connaît pas au préalable le nombre de fois où elle va se répéter. Se pose alors un problème, comment faire pour que la boucle s'arrête et ne tourne pas indéfiniment ? Pour évite cela, on utilise une condition (ou un critère) d'arrêt. Tant que cette condition est respectée, la boucle continue ; dès que la condition n'est plus respectée, la boucle s'arrête. On utilisera donc l'instruction Tant que.

Exemple : Dans l'algorithme ci-dessous, tant que la valeur de N est plus petite que 10, on la multiplie par deux et on arrête dès qu'elle devient plus grande que 10. On peut résumer l'évolution de la boucle par le tableau ci-dessous.

Algorithme 13 :

```
1 N ← 1
2 Tant que N < 10 :
3       N ← 2 * N
4 Afficher(N)
```

N	1				
Condition					

.....
.....
.....
.....

Remarques :

— La condition du Tant que est en fait un booléen.

— Lorsqu'on manipule des boucles Tant que, il faut être vigilant à deux choses :

1. Que la boucle puisse bien démarrer. En effet, si la condition de la boucle n'est pas vérifiée au départ, elle ne démarre pas. Par exemple :

Algorithme 14 :

```
1 N ← 1
2 Tant que N > 10 :
3         N ← 2 * N
4 Afficher(N)
```

Ici, N n'est pas plus grand que 10 et donc la condition de la boucle étant fausse, elle ne démarre pas.

2. Que le boucle ait une fin. En effet, si la condition est toujours vérifiée, la boucle ne s'arrête jamais. Par exemple :

Algorithme 15 :

```
1 N ← 1
2 Tant que N < 10 :
3         N ← N − 1
4 Afficher(N)
```

Ici, N est toujours plus petit que 10 $(1, 0, -1, \dots)$ et donc la condition de la boucle étant toujours vraie, elle continue indéfiniment.

2.4 Exercices

Exercice 2.1.

Algorithme 16 :

```
1 M ←Demande(Nombre entier)
2 G ← 2 × M − 4
3 Afficher(G)
```

1. Qu'affiche la machine en sortie si on rentre 7 en entrée ?
2. Et si on rentre $2,5$?
3. Quelle valeur rentrer en entrée de l'algorithme pour que la machine affiche : 24 ?

Exercice 2.2.

Algorithme 17 :

```
1 w ← Demande(nombre entier)
2 Si w < 11,5 Alors :
3         w ← w − 0,5 × w
4 Sinon :
5         w ← w + 1
6 Afficher : (w)
```

1. Qu'affiche en sortie cet algorithme si l'on rentre 3 en entrée ?
2. Et si l'on rentre 12 ?
3. Et si l'on rentre $11,4$?

Exercice 2.3.

Dans un jeu de dés à six faces non truqués, on perd 1€ si on fait 3 ou moins, on ne gagne ni ne perd rien si on fait 4 ou 5 et on gagne 3€ si on fait 6. Recopier et compléter l'algorithme suivant afin qu'il affiche le gain après un lancer de dé virtuel.

Algorithme 18 :

```
1 D ← (nombre entier aléatoire entre 1 et 6)
2 Si ... Alors :
3         ...
4 Si ... Alors :
5         ...
6 Sinon :
7         ...
```

Exercice 2.4. Qu'affichent les algorithmes ci-dessous en sortie ?

Algorithme 19 :

```
1 B ←"Ah !"
2 Répéter 3 fois :
3         B ← B + " Ah ! "
4 Afficher(B)
```

Algorithme 20 :

```
1 B ←"Ah !"
2 Répéter 3 fois :
3         Afficher(B)
4 B ← B + " Ah ! "
```

Algorithme 21 :

```
1 B ←"Ah !"
2 Répéter 3 fois :
3         Afficher(B)
4         B ← B + " Ah ! "
```

Algorithme 22 :

```
1 B ←"Ah !"
2 Répéter 3 fois :
3         B ← B + " Ah ! "
4         Afficher(B)
```

Algorithme 23 :

```
1 B ←"Ah !"
2 Répéter 3 fois :
3         B ← B + B
4         Afficher(B)
```

Exercice 2.5. Écrire un algorithme qui affiche 2356 fois le mot "mouette".

Exercice 2.6. Qu'affichent les algorithmes ci-dessous en sortie ?

Algorithme 24 :

```
1 Pour k Allant de 0 à 10 :
2         Afficher(2k)
```

Algorithme 25 :

```
1 Pour k Allant de 0 à 10 :
2         Afficher(2k + 1)
```

Exercice 2.7.

Algorithme 26 :

```
1 S ← ...
2 Pour k Allant de ... à ... :
3         S ← ...
4 Afficher(...)
```

Recopier et compléter cet algorithme de sorte qu'il affiche la somme des 10 premiers entiers naturels.

Exercice 2.8. Écrire un algorithme affichant les 101 premiers multiples de 11.

Exercice 2.9. Qu'affiche cet algorithme en sortie si on entre 0 pour P et 6 pour A ? Et si on entre 6 pour P et 0 pour A ?

Algorithme 27 :

```
1 P ← Demande(nombre flottant)
2 A ← Demande(nombre flottant)
3 Répéter 4 fois :
4         Si P < A Alors :
5                 P ← P + 3
6         Sinon :
7                 P ← P − 0,5
8 Afficher(P)
```

Exercice 2.10. Qu'affiche cet algorithme en sortie si on rentre 1 pour P et 18 pour A ? Et si on rentre 2 pour P et 29 pour A ?

Algorithme 28 :

```
1 P ←Demande(nombre flottant)
2 A ←Demande(nombre flottant)
3 Pour N Allant de 1 à 4 :
4         P ← P × 2
5 Si P < A Alors :
6         Afficher(P)
7 Sinon :
8         Afficher("C'est nul ! !")
```

Exercice 2.11. Qu'affichent les algorithmes ci-dessous en sortie ?

Algorithme 29 :

```
1 k ← 1
2 Tant que k < 10 :
3         k ← 3 ∗ k
4 Afficher(k)
```

Algorithme 30 :

```
1 k ← 16
2 Tant que k > 1 :
3         k ← k/4
4 Afficher(k)
```

Exercice 2.12. Déterminer si la condition de l'instruction Tant que des algorithmes suivants est bien définie ou non (boucle infinie ou ne démarrant pas).

Algorithme 31 :

```
1 k ← 5
2 Tant que k < 10 :
3         Afficher(k)
4         k ← (k + 1)/2
```

Algorithme 32 :

```
1 k ← 10
2 Tant que k ≤ 1 :
3         k ← k + 1
4 Afficher(k)
```

Algorithme 33 :

```
1 k ← 0
2 Tant que k < 10 :
3         Afficher(k)
4 k ← k + 1
```

Algorithme 34 :

```
1 T ←"Ah"
2 Tant que Longueur(T) ≤ 10 :
3         T ← T + T
4 Afficher(T)
```

Exercice 2.13.

Algorithme 35 :

```
1 Somme ← 0
2 Entier ← 0
3 Tant que ... :
4         Entier ← ...
5         Somme ← ...
6 Afficher(...)
```

Recopier et compléter cet algorithme de sorte qu'il affiche le dernier entier N tel que la somme des N premiers entiers, hormis 0, soit plus grande que 100 (par exemple, N n'est pas égal à 5 puisque $1+2+3+4+5=15<100$).

Exercice 2.14. Écrire un algorithme permettant de déterminer P la dernière puissance de 2 telle que la somme de celle-ci et des précédents soit plus grande que 1000 ; par exemple, P n'est pas 16 puisque $1+2+4+8+16=31<1000$.

2.5 Attendus et savoirs-faire

— Comprendre et utiliser les instructions d'entrée et de sortie d'une valeur dans un algorithme.

— Comprendre, utiliser, compléter et écrire un algorithme comportant un test, une boucle Pour / Répéter, une boucle Tant que.

— Identifier si une boucle Tant que est bien définie.