Bruno Hennart

DÉBUTER EN ALGORITHMIQUE ET EN PROGRAMMATION PYTHON

ellipses

Chapitre 1

DÉFINITION DE L'ALGORITHMIQUE

1. GÉNÉRALITÉS

Un algorithme est une suite d'opérations élémentaires ou instructions conduisant à un résultat donné lors de son exécution.

Si nous prenons l'exemple d'une recette de cuisine comme par exemple faire des frites.
Nous allons réaliser les opérations suivantes et dans cet ordre pour obtenir des frites :

- Éplucher des pommes de terre,
- Laver les pommes de terre épluchées,
- Égoutter les pommes de terre,
- Trancher les pommes de terre en lamelles,
- Mettre la friteuse en marche et attendre que l'huile chauffe,
- Mettre les lamelles de pomme de terre dans le panier de la friteuse,
- Plonger le panier dans l'huile chaude de la friteuse,
- Attendre et surveiller la cuisson,
- Sortir le panier de la friteuse quand les frites sont prêtes.

Sans le savoir, vous venez de créer un algorithme.
Il existe beaucoup d'autres exemples d'algorithmes identiques à la réalisation d'une recette de cuisine dans la vie courante comme indiquer le chemin à une personne, entretenir sa voiture ou rénover sa maison.

Pour résoudre un problème informatique, nous devons créer un programme qui sera exécuté par un ordinateur.
Il existe de multiples langages de programmation pour écrire un programme.
Le choix du langage adapté à la situation dépend de différents facteurs comme la technologie envisagée (web, mobile, sécurité ou réseau), l'objet du traitement, la thématique (IA, 3D, les statistiques), la maniabilité, la fiabilité, la portabilité ou encore la facilité d'utilisation de ce langage.

Chaque langage a sa propre syntaxe même si certains langages se ressemblent beaucoup. Malgré ces différences, il existe un point commun : la logique de programmation ou l'algorithmique.
L'algorithmique consiste, en effet, à diviser un problème en problèmes plus petits qui seront ensuite eux-mêmes découpés en instructions.
Un algorithme est le résultat de ce travail de découpage et il est indépendant des langages de programmation. Cet algorithme ne sera donc pas destiné à être exécuté directement sur un ordinateur.
Cette phase de travail constitue une ébauche de programme qui n'est pas directement opérationnelle et exécutable sur un ordinateur.

À l'issue de ce travail, cette ébauche sera traduite en un langage de programmation qui, lui, sera exécutable par un ordinateur.

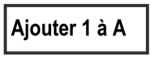
Bien qu'il n'existe pas vraiment de normes pour effectuer cette tâche d'écriture, cette étape peut être réalisée à l'aide d'outils comme l'ordinogramme ou le pseudo-code.

2. L'ORDINOGRAMME

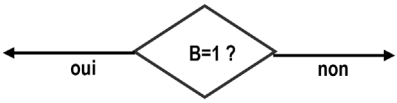
Il est constitué par un schéma du traitement des données dont les éléments sont les suivants :

- Le début et la fin d'un programme sont matérialisés par des figures ovales :

- Une action élémentaire ou instruction est représentée par un rectangle contenant le contenu de l'instruction :



- Un test est représenté par un losange avec des 2 issues possibles : vrai (ou oui ou encore 1) si le test est vrai ou faux (ou non ou 0) sinon :



- Une entrée comme la saisie au clavier par exemple ou une sortie comme l'affichage à l'écran ou encore une impression papier sont matérialisées par un parallélogramme contenant l'instruction d'entrée ou de sortie.

Exemple

Le problème à résoudre consiste à vérifier que le nombre saisi est divisible par 5.

Nous allons demander à l'utilisateur de saisir une valeur qui sera affectée à la variable A.

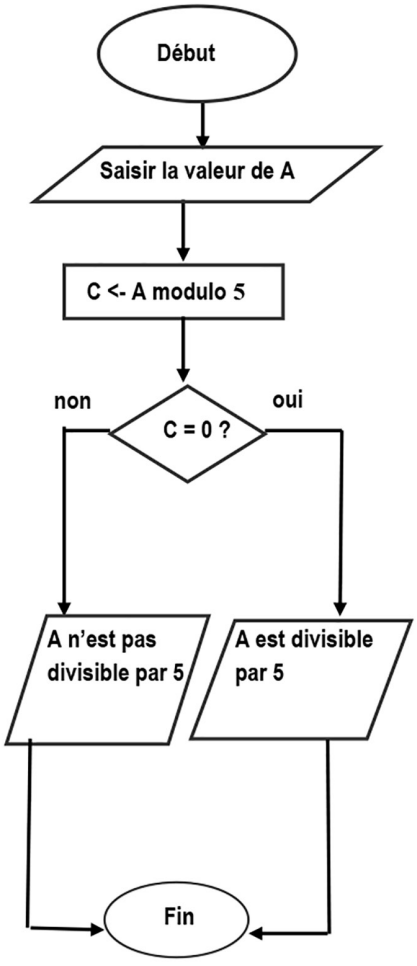
Nous allons utiliser la fonction modulo qui renvoie le reste de la division de A par 5.

La variable C contiendra le reste de la division de A par 5 :

- En saisissant 25, on donne la valeur 25 à A (A = 25) et, dans ce cas, le reste affecté à la variable C de la division sera égal à 0.
- $25 = 5 \times 5 + 0$ et C vaudra donc 0.
- Si on saisit le nombre 36 alors A = 36 et le reste C de la division sera 1.
- $36 = 5 \times 7 + 1$ et C vaudra donc 1.

En fonction de la valeur de C, on affichera alors soit « A est divisible par 5 » si C vaut 0 soit « A n'est pas divisible par 5 » si C ne vaut pas 0.

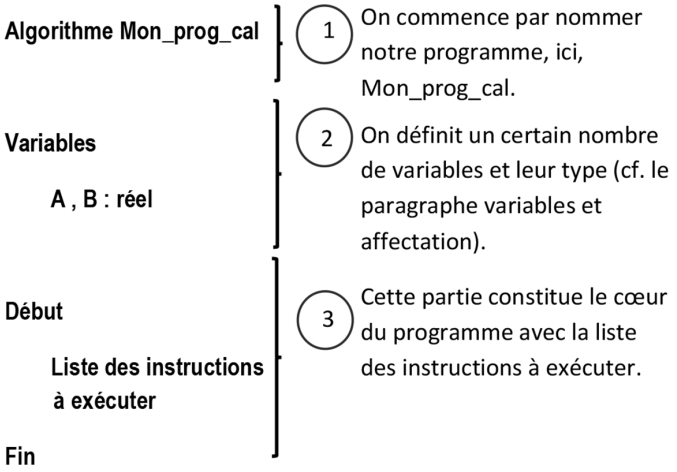
Ci-dessous, vous trouvez l'ordinogramme correspondant :



1. Etiquette marquant le début du programme
2. On commence par demander à l'utilisateur de saisir un nombre A.
3. On effectue l'opération modulo de A par 5 qui donne le reste de la division et on met le résultat dans la variable C.
4. On teste la valeur du reste de la division de A par 5 (affectée à C). Si C=0 alors A est divisible par 5 sinon A ne l'est pas.
5. En fonction du résultat du test précédent, soit on affiche le fait que A est divisible par 5 (à droite sur l'ordinogramme) soit on affiche le message inverse (à gauche sur l'ordinogramme).
6. Etiquette marquant la fin du programme

3. PSEUDO-CODE

La structure d'un algorithme en pseudo-code prend la forme suivante :



Si on reprend le problème de la divisibilité du nombre saisi par 5 et dont on nommera l'algorithme : **`Divisible_5`**, il comportera les étapes suivantes :

- Concernant les variables, on définit deux réels : A qui prendra la valeur du nombre saisi et C qui prendra le reste de la division de A par 5.
- Concernant les instructions, on demande à l'utilisateur de saisir un nombre que l'on affecte à la variable A.
- On effectue, ensuite, l'opération modulo entre A et 5 et on affecte le résultat à la variable C (C contient le reste de la division de A par 5 et vaut 0 si A est divisible par 5).
- Puis on teste la valeur de C et si C vaut 0 alors le nombre A est divisible par 5 sinon A ne l'est pas.
- Enfin, en fonction du résultat du test précédent, on affiche le message adéquat.

```
Algorithme Divisible_5

Variable
   A , C: réel
Début

   Ecrire ('enter un nombre A')
   Lire (A)

   C <- modulo(A,5)

   si C = 0 alors

      Ecrire (« A est divisible par
      5 »)

   sinon

      Ecrire (« A n'est pas
      divisible par 5 »)

   finsi
fin
```

1. On commence par donner un nom au programme
2. On définit 2 variables A et C de type réel
3. L'étiquette début marque le début des instructions qui vont être réalisées par le programme
4. On demande à l'utilisateur de saisir un nombre que l'on affecte à la variable A.
5. On affecte le reste de la division de A par 5 à la variable C à l'aide de l'opération modulo.
6. C contient le reste de la division de A par 5. On teste la valeur de C. Si C vaut 0 alors A est divisible par 5. Le test commence par un **si** et fini par un **alors** introduisant les instructions à réaliser si la condition est vraie.
7. On retrouve ici les instructions réalisées si le test est vrai. Dans notre cas, C vaut 0 donc A est divisible par 5 et on affiche le message : « A est divisible par 5 ».
8. Entre les mots **sinon** et **finsi,** on retrouve ici les instructions réalisées si le test est faux. Dans notre cas, C ne vaut pas 0 et on affiche le message : « A n'est pas divisible par 5 ».
9. Cette étiquette marque la fin du programme.

Chapitre 2

LES VARIABLES ET LES TYPES

1. QU'EST-CE QU'UNE VARIABLE ?

Un programme informatique a besoin de manipuler des valeurs comme les « j'aime » dans un réseau social ou le prix d'un article dans une application comptable.

Ces valeurs peuvent être saisies par un utilisateur, provenir d'un fichier ou être générées automatiquement par un programme. Le programme utilise des variables pour stocker ces valeurs qui peuvent changer au cours de l'exécution des instructions du programme.

Chaque variable a un nom unique utilisé par le programme. Les programmes manipulent les variables par leur nom.

Le nom de la variable ou identifiant ne doit contenir que des caractères alphanumériques ou le symbole _. Cet identifiant doit commencer par un caractère alphabétique.

Le nom d'une variable ne comporte pas d'accent ni d'espace.

Exemple

Identifiants valides

Les identifiants ci-dessous sont des identifiants valides :

Var1 | Mon_numero | mon_numero | prenom | Marque

Remarque : **`Mon_numero`** et **`mon_numero`** ne représentent pas la même variable compte tenu du M majuscule du premier identifiant. Ces variables sont donc distinctes.

Exemple

Identifiants non valides

Les identifiants suivants ne sont pas valides :

- **`1Var`** n'est pas valide car il commence par un nombre.
- **`A?123`** n'est pas valide car il contient un caractère interdit : ?
- **`m-12`** n'est pas valide car il contient un caractère interdit : –
- **`Numéro`** n'est pas valide car il contient un caractère accentué : é
- **`Mon prenom`** n'est pas valide car il contient un espace.

2. LE TYPE D'UNE VARIABLE

Une variable est également caractérisée par son type qui peut être :

- Le type entier correspond à un nombre entier,
- Le type réel correspond à un nombre à virgule (flottant),
- Le type booléen qui ne peut prendre que la valeur V (ou 1 ou oui) pour vrai ou F (ou 0 ou non) pour faux,
- Le type chaîne de caractère qui contient des caractères alphanumériques comme par exemple : 'abCdez?e'.

On parle de variables scalaires pour ces quatre types de variable.

Le programme ci-dessous : **`Mon_prog_cal`** comporte 2 variables A et B de type réel.

```
Algorithme Mon_prog_cal

Variables
    A , B: réel

Début
    Listes des instructions à
    exécuter

Fin
```

1. On commence par nommer notre programme, ici, Mon_prog_cal.
2. On définit un certain nombre de variables et leur type (cf. le paragraphe variables et affectation).
3. Cette partie constitue le cœur du programme avec la liste des instructions à exécuter.

3. NOTION D'INDENTATION

Dans l'algorithme ci-dessus, vous pouvez noter que la déclaration des variables et la liste des instructions subissent un retrait vers la gauche. Il s'agit d'une norme typographique appelée indentation. Elle consiste en l'ajout de tabulations ou d'espaces dans un fichier, pour une meilleure lecture et compréhension du code.

Dans certains langages comme Python, le défaut d'indentation provoque une erreur du code. Nous reviendrons sur ce sujet dans les chapitres suivants.

4. NOTION D'AFFECTATION

Les opérateurs d'assignation ou d'affectation permettent de remplir une variable avec une valeur.

L'affectation s'interprète de la façon suivante :

A ⟵ 6 signifie que la variable A prend la valeur 6.

On s'assurera de la cohérence de la variable avec la valeur affectée.
Un nombre entier ne peut, en effet, pas recevoir une chaîne de caractères comme valeur. Dans notre cas, A doit être une variable de type entier, réel ou chaîne de caractère pour accepter la valeur 6 mais pas un booléen.

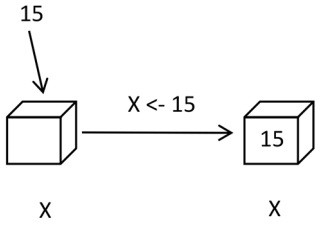
Quand on donne une valeur à une variable X, on écrit :

Affecter à X la valeur 15,
Ou
X prend la valeur 15,
Ou
X ⟵ 15.

Une variable est une sorte de boîte avec un nom et contenant une valeur.

Exemple

On affecte la valeur 15 à la variable nommée X.



5. LES OPÉRATEURS ARITHMÉTIQUES

Les opérateurs arithmétiques sont des opérateurs de calcul.

Outre les opérateurs élémentaires de calcul que sont l'addition, la soustraction, la multiplication (ayant pour symbole : *) et la division (ayant pour symbole : /), le tableau ci-dessous reprend la liste des autres principaux opérateurs :

Opérateurs	Symboles	Exemples
Division entière	DIV	DIV (27,5) = 5 car 27 = 5 × 5 + 2
Modulo (reste de la division)	%	27 % 5 = 2 car 27 = 5 × 5 + 2
Puissance	^	3 ^ 3 = 27

6. UTILISATION DE CES OPÉRATEURS DANS UN TRAITEMENT

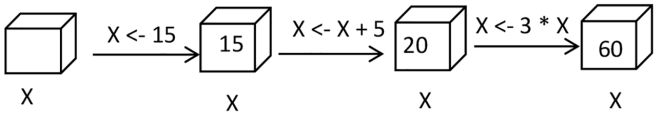
Ce type d'opérateur permet d'écraser la valeur précédente prise par la variable par une nouvelle valeur affectée à cette variable lors de la succession des instructions du traitement.

Exemple

Soit les instructions suivantes affectant la variable X :

- La variable X prend la valeur 15 (X ← 15).
- La variable X est augmentée de 5 (X ← X + 5).
- La variable X est multipliée par 3 (X ← X * 3).

Cette suite d'instructions exerce une influence sur la valeur de X et peut être schématisée de la manière suivante :

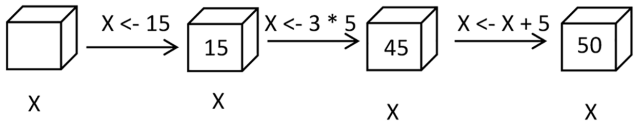


À la fin, la variable X contient la valeur 60.

L'ordre des instructions influence le résultat final, en effet, si on reprend les instructions précédentes dans un ordre différent en inversant la deuxième opération avec la troisième, on peut obtenir un résultat différent.

On a donc les instructions suivantes affectant la variable X :

- La variable X prend la valeur 15 (X ← 15).
- La variable X est multipliée par 3 (X ← X * 3).
- La variable X est augmentée de 5 (X ← X + 5).



À la fin, la variable X contient la valeur 50.

On constate que l'on n'obtient pas la même valeur finale et l'ordre des instructions influence donc le résultat. On parle d'opérations séquentielles qui s'exécutent les unes après les autres.

EXERCICES

EXERCICE 1

Soit la suite d'instructions suivantes concernant 2 variables numériques A et B :

1. A prend la valeur 7.

2. B prend la valeur A + 1.

3. A prend la valeur A + B.

Quelles sont les valeurs finales de A et B ?

EXERCICE 2

1. Que vaut X si on exécute les instructions suivantes :

X prend la valeur « Pluie ».

X prend la valeur « Soleil ».

2. Que vaut X si on exécute les instructions suivantes :

X prend la valeur « Soleil ».

X prend la valeur « Pluie ».

3. L'ordre des instructions a-t-il une importance ?

EXERCICE 3

Arthur a simulé l'exécution des instructions qui modifient la valeur de A mais, lors de ce travail, il a commis quelques erreurs, pouvez-vous l'aider et corriger les erreurs ?

Listes des instructions	Valeur de A
A ⟵ 4	4
A ⟵ 3 * A	16
A ⟵ A + 6	24
A ⟵ A / 6	4

EXERCICE 4

Soit la liste d'instructions suivantes :

1. On affecte 2 à X.
2. On affecte 7 à Y.
3. On affecte Y $-$ X à X.
4. On affecte Y $+$ X à Z.

Quelles sont les valeurs de X, Y et Z à la fin de cette liste d'instructions ?

EXERCICE 5

Si on considère un pavé de longueur 5 cm, de largeur 3 cm et de hauteur 4 cm, que représente la variable R avec la liste d'instructions ci-dessous ?

```
L ← 5
l ← 3
p ← 4
R ← L * l * p
```

EXERCICE 6

On souhaite permuter les valeurs des variables X et Y, quel algorithme ci-dessous permet cet échange ?

Algorithme 1
X prend la valeur 2
Y prend la valeur 5
X prend la valeur de Y
Y prend la valeur de X

Algorithme 2
X prend la valeur 2
Y prend la valeur 5
Z prend la valeur de X
X prend la valeur de Y
Y prend la valeur de Z

Algorithme 3
X prend la valeur 2
Y prend la valeur 5
Y prend la valeur de X
X prend la valeur de Y

CORRIGÉ DES EXERCICES

EXERCICE 1

La simulation des instructions avec les valeurs de A et B à chaque étape est :

1. Après la première instruction : **A** ← **7**, A vaut 7 et B n'a pas de valeur.
2. Après la deuxième instruction **B** ← **A** + **1**, A vaut 7 et B vaut 8.
3. Après la troisième instruction **A** ← **A** + **B**, A vaut 15 et B vaut 8.

Traduction en Python en complément

Note : la commande Python **`print`** permet d'imprimer les valeurs de A et B.

```
A=7
B=A+1
A=A+B
print(«A=»,A,«B=»,B)
```

EXERCICE 2

1. À la fin de l'exécution des instructions, X vaut « Soleil ».
2. À la fin de l'exécution des instructions, X vaut « Pluie ».
3. Avec les questions 1 et 2 de cet exercice, on s'aperçoit que X vaut « Soleil » dans le premier cas, puis, si on échange les instructions, X vaut « Pluie » dans le second cas.

   L'ordre des instructions à une importance prépondérante sur la valeur des variables.

EXERCICE 3

Listes des instructions	Valeur de A		Correction
A ← 4	4		A = 4
A ← 3 * A	16	FAUX	A = 3 × 4 = 12
A ← A + 6	24	FAUX	A = 12 + 6 = 18
A ← A / 6	4	FAUX	A = 18 ÷ 3 = 3

Traduction en Python en complément

```
A=4
A=3*A
A=A+6
A=A/6
print("A=",A)
```

EXERCICE 4

La simulation des instructions avec les valeurs de X, Y, Z est :

1. On affecte 2 à X — on a donc : X = 2, Y et Z n'ont pas de valeur.

2. On affecte 7 à Y — on a donc : X = 2, Y = 7 et Z n'a pas de valeur.

3. On affecte Y–X à X — on a donc : X = 5, Y = 7 et Z n'a pas de valeur.

4. On affecte Y+X à Z — au final, on a donc : X = 5, Y = 7 et Z = 12.

Traduction en Python en complément

```
X=2
Y=7
X=Y-X
Z=Y+X
print("X=",X,"Y=",Y,"Z=",Z)
```

EXERCICE 5

La variable R représente le volume du pavé soit : $5 \times 3 \times 4 = 60$ cm^3.

Avec $L = 5$, $l = 3$, $p = 4$ et $R = L \times l \times p$.

EXERCICE 6

Considérons chacun des algorithmes et leur simulation :

Algorithme 1	Simulation
X prend la valeur 2	X = 2
Y prend la valeur 5	X = 2 et Y = 5
X prend la valeur de Y	X = 5 et Y = 5
Y prend la valeur de X	X = 5 et Y = 5

Cet algorithme n'échange donc pas les valeurs de X et Y.

Algorithme 3	Simulation
X prend la valeur 2	X = 2
Y prend la valeur 5	X = 2 et Y = 5
Y prend la valeur de X	X = 2 et Y = 2
X prend la valeur de Y	X = 2 et Y = 2

Cet algorithme n'échange donc pas les valeurs de X et Y.

Algorithme 2	Simulation
X prend la valeur 2	X = 2
Y prend la valeur 5	X = 2 et Y = 5
Z prend la valeur de X	X = 2, Y = 5 et Z = 2
X prend la valeur de Y	X = 5, Y = 5 et Z = 2
Y prend la valeur de Z	X = 5, Y = 2 et Z = 2

Seul, l'algorithme 2 permet de permuter les valeurs de X et Y avec l'introduction d'une variable tampon : Z.

Traduction en Python en complément

```
X=2
Y=5
Z=X
X=Y
Y=Z
print("X=",X,"Y=",Y)
```

Chapitre 3

ÉCRIRE UN ALGORITHMIQUE

1. STRUCTURE D'UN ALGORITHME

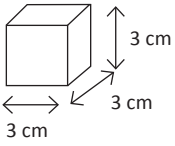
Dans ce chapitre, nous allons voir la structuration d'un algorithme ainsi que les interactions d'un programme avec son utilisateur.

Après avoir analysé le problème, on commence l'écriture de l'algorithme qui comprend 3 parties :

1. Une phase de déclaration et d'initialisation des variables nécessaires.
2. Une phase de traitement du problème constituée par une liste d'instructions.
3. Une phase de sortie des résultats.

Exemple

On souhaite calculer le volume d'un pavé carré de côté C = 3 cm.



Pour cela, nous aurons les 3 parties suivantes dans notre programme :

- **Déclarations et initialisations**
  Nous allons déclarer 2 variables : C qui représentera la longueur du côté du pavé et sera initialisé à 3 cm et V qui représentera le volume du pavé.
- **Traitement**
  Cette partie du programme ne contiendra qu'une instruction : l'affectation de C^3 à V ($V = C^3$).
- **La sortie des résultats**
  On affichera la valeur de V (soit 27).

L'algorithme que l'on appellera **Vol_Cube** sera :

```
Algorithme Vol_Cube
      Variables
            C, V : réel
      Début
            V ← C ^ 3
            Écrire (C)
      Fin
```

Remarque : On applique une indentation de 4 caractères sur les étiquettes Variable, Début et Fin de cet algorithme. On applique également une indentation de 4 caractères sur le contenu de ces blocs (le bloc de déclaration des variables et celui du traitement entre l'étiquette début et fin).

2. LECTURE ET ÉCRITURE

Un programme informatique pour être utile et efficace doit pouvoir interagir avec ses utilisateurs et cette interaction se traduit notamment par l'affichage de messages à l'écran pour que l'utilisateur les lise et la saisie d'informations par l'utilisateur qui sera prise en compte par le programme pour être traitée par lui.

Soit l'algorithme suivant :

```
Algorithme multiplie
Variables
   A, B, C: Entier
Début

   A <- 15
   B <- 3
   C <- A * B

    Ecrire(C)
Fin
```

A chaque exécution de ce programme A prendra toujours la valeur 15 et B, la valeur 3. C sera toujours égal à 45

Cet algorithme réalise la multiplication des entiers A par B qui ont toujours les valeurs 15 et 3 respectivement et dont le résultat sera affecté à la variable C qui prendra toujours la valeur 45 mais cet algorithme ne présente aucun intérêt, puisqu'à chaque exécution, il donnera toujours le même résultat : 45.

Pour améliorer cet algorithme et le rendre utile et efficace, nous allons demander à l'utilisateur de fournir des valeurs pour les variables A et B. L'algorithme va afficher à l'écran un texte : 'Entrez un nombre'. Cet algorithme attendra alors que l'utilisateur saisisse une valeur qu'il affectera à la variable A puis réaffichera le même texte et affectera la valeur saisie à la variable B. Pour cela, le programme va récupérer ces valeurs à l'aide de la fonction **`Lire`**.

Le programme effectuera ensuite le produit de A par B et l'affectera à la variable C. Le programme affichera avec l'instruction **`Écrire`** la valeur de C qui sera le résultat du produit des deux nombres saisis.

Ce dialogue constitué par la saisie de l'utilisateur des valeurs A et B et l'affichage du résultat, C, est une interaction entre le programme et l'utilisateur. Cette interaction permettra de faire la multiplication de deux nombres choisis et fournis au programme afin d'obtenir le résultat de ce produit.

Voici, ci-dessous, l'algorithme correspondant :

```
Algorithme multiplie
  Variables
    A, B, C: Entier
  Début
    Ecrire ('Entrez un nombre ')
    Lire (A)
    Ecrire ('Entrez un nombre ')
    Lire (B)

    C <- A * B

    Ecrire (C)

  Fin
```

Ces instructions constituent la phase de saisie et d'interaction de l'utilisateur avec le programme

Cette instruction constitue la phase d'affichage et d'interaction du programme avec l'utilisateur

Remarque : Certains langages et programmes permettent d'interagir avec l'utilisateur grâce à d'autres types d'interaction comme le clic de la souris par exemple.

3. LES CHAÎNES DE CARACTÈRES

Une chaîne de caractères est une suite de caractères. Cette chaîne de caractères constitue un texte qui peut représenter un nom, un prénom, une adresse mail ou tout autre type de données alphanumériques.

Dans le bloc de déclaration des variables, nous aurons la déclaration suivante pour la variable **`Mon_texte`** de type chaîne de caractères :

Variables

```
Mon_texte : chaîne de caractères
```

4. LA CONCATÉNATION DES CHAÎNES DE CARACTÈRES

Une opération importante sur les chaînes de caractères s'appelle la concaténation. Elle permet de former une nouvelle chaîne de caractères à partir de variables et de chaînes de caractères en les juxtaposant. Pour cela on utilise le symbole de concaténation : **&**.

Exemple

Soit les variables chaînes de caractères :

```
A ← 'Il fait beau',
B ← 'et'
C ← 'je vais sortir'
```

À partir de ces chaînes de caractères, on peut créer la nouvelle chaîne de caractères **Phrase** dont le contenu sera : **'Il fait beau et je vais sortir'**.

L'instruction pour créer cette nouvelle chaîne est la suivante :

```
Phrase=A&' '&B&' '&C
```

On juxtapose le contenu de A suivi d'un espace puis le contenu de B suivi d'un espace et enfin le contenu de C.

Traduction en Python

```
Phrase=A+''+B+''+C
```

5. LA FONCTION ÉCRIRE

La fonction Écrire permet également de concaténer directement différentes chaînes de caractères.

Les paramètres passés à cette fonction seront séparés par des virgules.

Pour afficher cette même phrase : **'Il fait beau et je vais sortir'** à partir des mêmes variables A, B et C que précédemment, on rédige l'instruction suivante :

```
Écrire (A,'',B,'',C)
```

Remarque : La plupart des langages de programmation offrent de nombreuses autres fonctions permettant de traiter et manipuler des chaînes de caractères.

Traduction en Python

```
print(A,‘‘,B,‘‘,C)
```

EXERCICES

EXERCICE 1

Rédiger l'algorithme permettant de calculer la moyenne de la taille des 5 enfants de la famille Martin.

Ce programme se nommera **`Moyenne_Martin`**. Il faudra définir les variables A, B, C, D, E pour désigner la taille des 5 enfants et **`Moy`** sera la variable correspondant à la moyenne.

Cet algorithme devra interagir avec l'utilisateur pour pouvoir saisir les tailles, faire le calcul et afficher le résultat.

EXERCICE 2

Soit l'algorithme suivant :

```
Algorithme Question
  Variables
    A, B, C, D : chaîne de caractères
  Début
    A ← 'Bonjour,'
    B ← 'Comment'
    C ← 'Allez-vous ?'
    D ← A&B&C
    Écrire (D)
    Écrire (A,'',B,'',C)
  Fin
```

Qu'affiche cet algorithme ?

EXERCICE 3

Soit l'algorithme suivant :

```
Algorithme Invers
  Variables
    A, B, B1, B2, B3 : chaîne de caractères
  Début
    Écrire ('saisir un mot')
    Lire (A)
    B1 ← premier caractère de A
    B2 ← deuxième caractère de A
    B3 ← troisième caractère de A
```

```
    B ← B3&B2&B1
    Écrire (B)
    Écrire (B3,''',B2,''',B1)
  Fin
```

Qu'affiche cet algorithme si on affecte la chaîne : 'Soleil' à A ?

EXERCICE 4

Lors des soldes, un commerçant en prêt à porter doit modifier ses étiquettes.

Il souhaite faire une remise de 10 % sur une chemise affichée au prix de 25 € et une remise de 30 % sur un pull de 45 €.

1. Quel prix figurera sur la nouvelle étiquette de la chemise ?

2. Soit l'algorithme suivant :

```
Algorithme RRR
  Variables
    P, R : réel
  Début
    Écrire ('entrer P')
    Lire (P)
    Écrire ('entrer R')
    Lire (R)
    P ← P - P * T / 100
    Écrire (P)
  Fin
```

a. Quelle est la valeur afficher si P = 25 et R = 10 ?

b. Quel est le rôle de cet algorithme ?

c. Quelle valeur faut-il attribuer à P et R pour calculer le nouveau prix du pull ?

EXERCICE 5

Pour fabriquer 3 m^3 de béton il faut mélanger : 1 050 kg de ciment, 2 100 kg de sable, 3 150 kg de graviers et 375 litres d'eau.

1. Quelle quantité de ciment faut-il pour fabriquer 5 m^3 ?

2. Soit l'algorithme suivant :

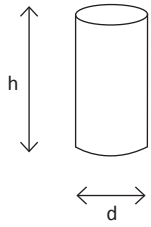
```
Algorithme Beton
  Variable
    Qté : réel
  Début
    Écrire ('Donner une Quantité : ')
    Lire (Qté)
```

```
    Qté ← Qté * 1 050 / 3
    Écrire (Qté)
  Fin
```

a. Quel est le rôle de cet algorithme ?
b. Que doit-on modifier à l'algorithme si on veut connaître la quantité de sable nécessaire ?

EXERCICE 6

Soit un cylindre de hauteur h et de diamètre d :



1. Compléter l'algorithme suivant qui calcule le volume de ce cylindre.

On approxime π à la valeur de 3,14.

```
Algorithme Cal_Vol
  Variables
    V, d, h, Pi : réel
  Début
    Écrire ('donner le diamètre de la base du cylindre : ')
    Lire (d)
      ........
  Fin
```

2. Compléter l'algorithme suivant qui calcule l'aire et le périmètre de la base de ce cylindre.

```
Algorithme PetM_base_cyl
  Variables
    P, A, d, Pi : réel
  Début
    Écrire ('donner le diamètre de la base du cylindre : ')
    Lire (d)
      ........
  Fin
```

EXERCICE 7

1. José est un adepte de la course à pied. Ce matin, il a parcouru 15 km à 12 km/h. Pendant combien de temps a-t-il couru ?
2. Compléter l'algorithme ci-dessous pour effectuer ce type de calcul.

```
Algorithme cal_tps
  Variables
    ........
  Début
    ........
  Fin
```

3. Hier, il avait couru pendant une 1 heure 15 à 11 km/h, quelle distance a-t-il parcourue ?
4. Compléter l'algorithme ci-dessous pour effectuer ce type de calcul.

```
Algorithme cal_dist
  Variables
    ........
  Début
    ........
  Fin
```

CORRIGÉ DES EXERCICES

EXERCICE 1

L'algorithme permettant de calculer la moyenne de la taille des 5 enfants de la famille Martin est :

Pour calculer la moyenne, on demande à l'utilisateur de saisir les 5 tailles en cm, on ajoute ensuite les 5 tailles des enfants et on divise par 5. Il s'agit d'une moyenne simple.

```
Algorithme Moyenne_Martin
  Variables
    A, B, C, D, E, Moy : réel
  Début
    Écrire ('Entrer la 1re Taille : ')
    Lire (A)
    Écrire ('Entrer la 2e Taille : ')
    Lire (B)
    Écrire ('Entrer la 3e Taille : ')
    Lire (C)
    Écrire ('Entrer la 4e Taille : ')
    Lire (D)
    Écrire ('Entrer la 5e Taille : ')
    Lire (E)
    Moy ← (A+B+C+D+E)/5
    Écrire (Moy)
  Fin
```

Traduction en Python en complément

```
A=float(input('Entrer la 1ière taille:'))
B=float(input('Entrer la 2ième taille:'))
C=float(input('Entrer la 3ième taille:'))
D=float(input('Entrer la 4ième taille:'))
E=float(input('Entrer la 5ième taille:'))
Moy=(A+B+C+D+E)/5
print(Moy)
```

EXERCICE 2

La commande **Écrire (D)** de l'algorithme **Question** affiche :

Bonjour,CommentAllez-vous ?

Cette instruction ne contient pas d'espace entre les variables A, B et C.

La commande **Écrire (A,'‘,B,'‘,C)** contient des espaces entre les variables et affiche :

Bonjour, Comment Allez-vous ?

EXERCICE 3

Si on affecte la chaîne : 'Soleil' à A, la commande **Écrire (B)** de l'algorithme **Invers** affiche : loS.

Cette instruction ne contient pas d'espace entre les variables B1, B2 et B3.

La commande **Écrire (B3,'‘,B2,'‘,B1)** affiche : l o S.

Dans chacune des instructions d'écriture, on a inversé les 3 premières lettres du mot Soleil.

EXERCICE 4

Lors des soldes, un commerçant doit modifier ses étiquettes.

Il souhaite faire une remise de 10 % sur une chemise affichée au prix de 25 € et une remise de 30 % sur un pull de 45 €.

1. Le prix de la nouvelle étiquette de la chemise affichera : 22,5 €.

$P = 25 - 25 \times 10 / 100 = 25 - 2{,}5 = 22{,}5$ €.

P représente le prix et R la remise.

On calcule la remise de 10 % de 25 € soit : $25 \times 10 / 100 = 2{,}5$ €.

On soustrait la remise ci-dessus au prix d'origine : $25 - 2{,}5 = 22{,}5$ €

2. Concernant l'algorithme RRR,

a. Si $P = 25$ et $R = 10$, l'algorithme affichera : 22,5 €. On obtient ainsi le prix après remise de la chemise.
$25 - 25 \times 10 / 100 = 22{,}5$ €.

b. Cet algorithme calcule le nouveau prix d'un article en fonction de son prix de départ et de la remise appliquée.

c. Pour le pull, le prix initial est de 45 € et on affectera 45 à P et la remise est de 30 % et on affectera 30 à R. L'étiquette affichera 31,5 €.
$45 - 45 \times 30 / 100 = 31{,}5$ €.

EXERCICE 5

1. Pour fabriquer 5 m^3, il faut : 1 750 kg de ciment.

1 050 kg de ciment sont nécessaires pour fabriquer 3 m^3 de béton.

Il faut donc $1\,050 \div 3 = 350$ kg de ciment pour 1 m^3 de béton.
Pour fabriquer 5 m^3, il faut : $350 \times 5 = 1\,750$ kg de ciment ou encore :

$1\,050 \div 3 \times 5$.

2. Concernant l'algorithme,

a. Le rôle de cet algorithme est de calculer le poids de ciment nécessaire pour fabriquer Qté m^3 de béton.

b. Pour calculer le sable nécessaire en fonction de la quantité de béton désiré, il faut remplacer l'instruction **`Qté=Qté*1 050/3`** par **`Qté=Qté*2 100/3`**. Il faut, en effet, 2 100 kg de sable pour 3 m^3 de béton.

L'algorithme sera donc :

```
Algorithme Beton
  Variable
    Qté : réel
  Début
    Écrire ('Donner une quantité : ')
    Lire (Qté)
    Qté ← Qté * 2 100 / 3
    Écrire (Qté)
  Fin
```

EXERCICE 6

1. L'algorithme qui calcule le volume du cylindre est :

On approxime π à la valeur de 3,14.

```
Algorithme Cal_Vol
  Variables
    V, d, h, Pi : réel
  Début
    Écrire ('donner le diamètre de la base du cylindre : ')
    Lire (d)
    Écrire ('donner la hauteur du cylindre : ')
    Lire (h)
    Pi ← 3,14
    d ← d / 2
    V ← Pi * h * d ^ 2
    Écrire ('Le volume du cylindre est : ', V)
  Fin
```