Abderrahmane Fadil

Algorithmique et développement Python

Cours et exemples d'applications



ellipses

Références sciences

Algorithmique et développement Python

Cours et exemples d'applications

Abderrahmane Fadil

ellipses

Collection Références sciences

dirigée par Paul de Laboulaye
paul.delaboulaye@editions-ellipses.fr

Retrouvez tous les livres de la collection et des extraits sur www.editions-ellipses.fr

ISBN 9782340-061125
©Ellipses Édition Marketing S.A., 2021
8/10 rue la Quintinie 75015 Paris

DANGER
LE PHOTOCOPILLAGE TUE LE LIVRE

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5.2° et 3°a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit constituerait une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

Avant-propos

L'apprentissage de l'informatique, méthodes et applications, est devenu une composante incontournable dans les programmes et syllabus de l'enseignement supérieur. A l'origine, le besoin de l'automatisation des traitements et notamment dans les domaines scientifiques, a permis la mise en place de méthodes structurantes de raisonnement. D'où l'essor de l'algorithmique moderne et des langages et environnements de développement informatiques.

Ce livre a pour objectif de présenter les bases de l'algorithmique permettant à un public d'étudiants de mieux structurer et programmer des algorithmes de traitement dans différents secteurs d'activité.

Cet objectif est également cohérent avec les recommandations des organismes de tutelle en enseignement supérieur et commissions d'accréditation comme la Commission des Titres d'Ingénieur (CTI). Ces recommandations concernent l'introduction de l'apprentissage de l'algorithmique, de la programmation, des bases de données et l'utilisation des approches numériques et outils informatiques dans tous les cursus ingénieurs.

La programmation des algorithmes est mise en œuvre via le langage Python. Le choix de Python est motivé par les facilités d'apprentissage et de mise en œuvre ainsi que la disponibilité et la variabilité de ses ressources. Ce langage est libre et il fait partie de la tête des classements dans le monde de la programmation et de l'enseignement.

Table des matières

Chapitre I. Représentation des données et algèbre de Boole 11

1 Généralités sur les algorithmes 11
1.1 Introduction 11
1.2 Définition 1 11
1.3 Exemples 11
1.4 Définition 2 12
1.5 Ecriture syntaxique 13
1.6 Remarques 14
1.7 Les instructions fondamentales 14

2 Algorithmes et programmation informatique 14
2.1 Rappels et définitions 15
2.2 Processus d'exécution d'un algorithme 16
2.3 Utilisation des langages de programmation 19

3 Représentation des données 20
3.1 Les systèmes de numérotation 20
3.2 Décomposition d'un nombre 22
3.3 Conversion du binaire à l'octal 23
3.4 La conversion du binaire à l'hexadécimal 23
3.5 Conversion du système décimal au système de base b 24
3.6 Utilité des changements de bases 24

4 Algèbre de Boole 25
4.1 Définitions 25
4.2 Propriétés 26
4.3 Théorèmes de MORGAN 26
4.4 Portes logiques 27

5 Fonctions booléennes 28
5.1 Forme disjonctive normale 28
5.2 Forme conjonctive normale 29

6 Exercices 30
6.1 Systèmes de numérotation 30
6.2 Exercices 1ère forme et 2ème forme canonique 30
6.3 Constatations puis modélisation 31
6.4 Simplification algébrique 31

7 Correction des exercices 31
7.1 Systèmes de numérotation 31
7.2 Exercices 1ère forme et 2ème forme canonique 32
7.3 Constatations puis modélisation 34
7.4 Simplification algébrique 34
7.5 Exercice libre 34

Chapitre II. Algorithmes : les bases 35

1 Ecriture d'un algorithme 35
1.1 Définitions et rappels 35
1.2 Approche ascendante 35
1.3 Approche descendante 38

2 Structure algorithmique 39
2.1 Rappel 39
2.2 Modèle d'écriture 40

3 Bloc de déclarations 41
3.1 Les variables 41
3.2 Les constantes 51

4 Bloc d'instructions 52
4.1 L'affectation 52
4.2 Les affectations multiples simultanées 53
4.3 Les entrées / sorties 54
4.4 Les tests et structures de contrôles conditionnelles 56
4.5 Les disjonctions de cas 'Sinon, Si' 58
4.6 Exemple d'un système de chauffage automatique 60
4.7 Exemple 61

5 Les algorigrammes 62
5.1 Définition 62
5.2 Avantages et contextes d'utilisation des algorigrammes 62
5.3 Objets graphiques 63
5.4 Exemples 63

Chapitre III. Traitements répétitifs 67

1 La répétition 67
1.1 Contexte 67
1.2 L'itération 68
1.3 Conditions de répétition 68

2 Type de répétition 69
2.1 Les boucles 69
2.2 Exemples 70

3 La boucle *'Pour'*.....71
3.1 Sens incrémental.....71
3.2 Sens décrémental.....74
3.3 Syntaxe de la boucle *'Pour'* sous Python.....77

4 La boucle *'Tant Que'*.....79
4.1 Syntaxe algorithmique de la boucle *'Tant Que'*.....79
4.2 Syntaxe de la boucle *'Tant Que'* sous Python.....79

5 Exemple.....80

6 Algorigrammes des boucles répétitives.....81
6.1 Algorigramme de la boucle *'Pour'*.....81
6.2 Algorigramme de la boucle *'Tant Que'*.....82

7 Exemples sous Python.....83
7.1 Manipulation des nombres complexes.....83
7.2 Intégration numérique utilisant la méthode des rectangles.....84
7.3 Nombres parfaits.....85
7.4 Fonctions trigonométriques.....88
7.5 Résolution d'une équation de second degré.....90
7.6 Fonction exponentielle.....94

Chapitre IV. Structures de données listes et tableaux.....97

1 Tableaux et listes en algorithmique.....97
1.1 Exemple 1.....97
1.2 Exemple 2.....97
1.3 Les tableaux en algorithmique.....98
1.4 Les listes en algorithmique.....99

2 Les tableaux sous Python.....104
2.1 Les modules.....104
2.2 Les tableaux avec le module numpy.....106

3 Tableaux unidimensionnels.....106
3.1 Création d'un tableau.....106
3.2 Utilisation d'une portion d'un tableau en absolu ou en copie.....109
3.3 Création d'un tableau de réels dans l'espace linéaire.....111

4 Tableaux multidimensionnels.....112
4.1 Représentation matricielle.....112
4.2 Représentation linéaire.....113
4.3 Le nombre d'éléments d'un tableau.....113
4.4 La taille d'un tableau.....114
4.5 Création d'un tableau avec *zeros()* ou avec *ones()*.....114
4.6 Complément initialisation d'un tableau et/ou une matrice.....116

4.7 Les matrices par la méthode *mat()* de numpy 116
4.8 Conversion d'un tableau en matrice 118
4.9 Quelques opérations et fonctions matricielles 118
4.10 Script de résolution d'un système d'équations linéaires 121
4.11 Valeurs et vecteurs propres 123

5 Exemples supplémentaires avec algorithmes et scripts Python 124
5.1 Exemple 1 124
5.2 Exemple 2 125
5.3 Exemple 3 126

6 Exercices de recherche dans un tableau 128
6.1 Recherche séquentielle dans un tableau non trié 128
6.2 Recherche exhaustive version 2 130
6.3 Recherche exhaustive version 3 132
6.4 Existence d'un élément dans un tableau de données 135

7 Vérification mot palindrome en tant que liste de caractères 138
7.1 Script palindrome 138
7.2 Test mot palindrome 138

Chapitre V. Tri et fouille de données 139

1 Tri et fouille de données 139
1.1 Tri des données 139
1.2 Fouille de données 139

2 Méthodes de tri 140
2.1 Tri par échange 140
2.2 Tri à bulles 144

3 Fouille de données 147
3.1 Recherche séquentielle dans un tableau trié 148
3.2 Recherche dichotomique dans un tableau trié 151

Chapitre VI. Procédures, fonctions et récursivité 157

1 Procédures et fonctions 157
1.1 Introduction 157
1.2 Algorithme appelant et algorithme appelé 158

2 Procédures 159
2.1 Définition 159
2.2 Exemple de fusion de deux tableaux 159
2.3 Syntaxe algorithmique d'une procédure 159
2.4 Etapes d'exécution d'une procédure 160
2.5 Procédure de fusion de deux tableaux triés 161

2.6 Syntaxe d'écriture d'une procédure sous Python 165
2.7 Script Python de la procédure de fusion de deux tableaux 165
2.8 Cas de deux tableaux non triés 167

3 Fonctions 168
3.1 Définition 168
3.2 Exemple de calcul des coefficients d'un polynôme 168
3.3 Syntaxe algorithmique d'une fonction 169
3.4 Etapes d'exécution d'une fonction 169
3.5 Fonctions de calcul des coefficients d'un polynôme 170
3.6 Syntaxe d'écriture d'une fonction sous Python 174
3.7 Script Python de calcul des coefficients polynômiaux 175
3.8 Produit, somme et transposée de matrices 177

4 Récursivité 184
4.1 Principe de la récursivité 184
4.2 Avantage de la récursivité 185
4.3 Mise en œuvre de la récursivité 185
4.4 Procédures et fonctions récursives 186

5 Autres exemples 193
5.1 Mots palindromes 194
5.2 Suite mystère 196
5.3 Série des inverses carrés 198
5.4 Fonction de couplage de Cantor 201
5.5 Suite de Syracuse 203
5.6 Liste des nombres premiers 207
5.7 Les nombres parfaits 213
5.8 Série alternée 216
5.9 La puissance k-ième d'une matrice carrée 220

Bibliographie 223

Index 225

Chapitre I. Représentation des données et algèbre de Boole

1 Généralités sur les algorithmes

1.1 Introduction

Depuis la machine de Turing et la procédure de déchiffrement des messages codés, le mot algorithme est associé aux traitements automatiques, aux stratégies d'optimisation et à l'informatique moderne. Actuellement on l'associe aux méthodes d'analyse et de traitement des données, à l'analyse prédictive, au datamining et encore à l'apprentissage, au *deep learning* et à l'Intelligence Artificielle.

Aujourd'hui et demain, chaque fois qu'il s'agit de mettre en place un traitement, automatique, autonome, qui évolue intelligemment dans des contextes connus ou nouveaux, on fera appel aux algorithmes et à l'Intelligence Artificielle.

L'algorithme est un terme d'origine arabe qui fait référence au mathématicien Al Khawarizmi (780 - 850) auteur d'un ouvrage décrivant des méthodes de calculs algébriques. A l'origine du mot **algorithme** en français (*algorithm* en anglais, *algorithmus* en allemand, *algorithmo* en espagnol, *algorittmul* en roumain, …) on trouve le mot Algorismus qui est la forme latine du nom Khwarizmi.

1.2 Définition 1

Un algorithme est une méthode de résolution de problèmes énoncée sous la forme d'une série d'instructions ou de commandes, c'est-à-dire des opérations à effectuer.

1.3 Exemples

Voici une liste, évidemment non exhaustive, de situations ou problèmes à résoudre et dont la résolution consiste en amont à procéder par méthode et à effectuer une séquence d'opérations dans un ordre cohérent menant ainsi à la ou à une solution du problème posé :

- préparer une recette de cuisine ;
- calculer puis préparer une ration, équilibrée, nutritionnelle équine ;
- récompenser la fidélité d'un client dans le cadre d'une gestion de la relation client GRC ;

- stocker et diffuser les données d'un réseau de capteurs météorologiques implantés sur une parcelle ou un territoire ;
- déterminer la meilleure implantation géomarketing d'une enseigne marchande ;
- déterminer la meilleure implantation géoéconomique et environnementale d'un silo à grain ;
- déterminer en temps réel la présence d'un individu dans une foule d'individus ;
- déterminer la probabilité et le niveau de risque d'irruption d'un volcan.
- En analyse de données, calculer l'inverse, lorsqu'il existe, d'une matrice carrée d'ordre 100 ;
- décoder l'ADN d'un être vivant (algorithmes génétiques) ;
- caractériser des services de l'Internet des objet connectés (Internet of Things-IoT) ;
- déterminer le chemin le plus court ou le moins coûteux dans un réseau de relais d'information ;
- reconnaître et comptabiliser des ravageurs en agriculture connectée en utilisant des algorithmes d'apprentissage automatique que l'on appelle le Deep Learning ;
- etc.

La mise en œuvre de l'algorithme consiste en l'écriture de ces instructions dans un langage appelé « langage de programmation » et constitue alors la brique de base d'un programme informatique.

1.4 Définition 2

Un algorithme est une suite finie d'opérations élémentaires, à appliquer dans un ordre déterminé, à des données. Sa réalisation permet de résoudre un problème donné.

La *figure 1.* est une représentation systémique d'un algorithme. En effet, une séquence d'instructions définit et met en œuvre des fonctions de traitement de données. Ces fonctions agissent sur des données en entrée e_1, e_2, …, e_n où n est le nombre de ces données en entrée, c'est-à-dire les données qui alimentent l'algorithme. Le traitement utilise ces données pour en produire d'autres.

A la fin du traitement, les données produites sont les données en sortie de l'algorithme s_1, s_2, …, s_m où m est le nombre de ces données en sortie et correspondent aux résultats escomptés du traitement.

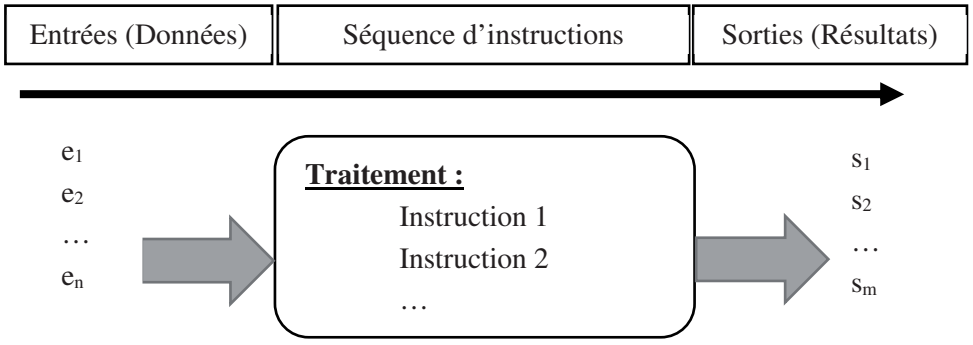


Figure 1. *Modèle algorithmique*

Prenons comme exemple (*figure 2.*), le calcul du prix toutes taxes comprises (pttc) d'un article étant donnés son prix hors taxes (pht) et sa tva :

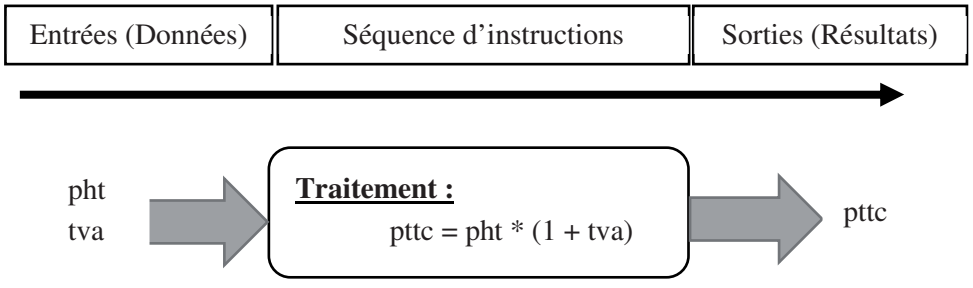


Figure 2. *Algorithme calcul pttc*

1.5 Ecriture syntaxique

Il y a plusieurs conventions d'écriture d'un algorithme. Le but d'une convention algorithmique est de :

- écrire un algorithme dans un langage proche du langage courant en adoptant une syntaxe d'écriture (une grammaire) ;
- permettre une sorte de standardisation d'écriture, ainsi toutes les personnes avisées puissent faire la même lecture de l'algorithme ;
- faciliter la traduction du langage algorithmique en langage de programmation ;
- de prouver formellement la performance et la convergence d'un algorithme.

Un algorithme peut être considéré comme l'écriture d'un ***modèle abstrait de traitement*** (c'est-à-dire modèle + méthode de résolution) d'une ou de plusieurs instances d'une situation réelle (*figure 3.*).

Voici un exemple d'écriture conventionnelle de l'algorithme précédent de calcul du prix pttc :

```
Algorithme Calcul_PTTC
    Variables
        pht, tva, pttc : réels
    Traitement
        Saisir(pht)        # Utiliser le # pour un commentaire
        Saisir(tva)        # comprise strictement entre 0 et 1
        pttc = pht * (1 + tva)
        Imprimer(pttc)
Fin Algorithme
```

1.6 Remarques

Pour fonctionner, un algorithme doit donc contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter : *Homme* ou *Machine*.

La taille d'un algorithme en nombre d'instructions ne conditionne pas en soi sa complexité : de longs algorithmes peuvent être finalement assez simples, et de petits algorithmes peuvent être très 'complexes'

1.7 Les instructions fondamentales

Les instructions fondamentales utilisées dans un algorithme sont :

- l'affectation ;
- la lecture / écriture ;
- les tests ;
- les boucles ou répétition.

2 Algorithmes et programmation informatique

L'informatique est la science du traitement automatique de l'information. Pour cela il faut :

- modéliser cette information ;
- définir à l'aide d'un formalisme strict les traitements dont elle fera l'objet ;
- et enfin traduire ces traitements dans un langage informatique compréhensible par un ordinateur et/ou son système d'exploitation.

Les deux premiers points aboutissent à un modèle abstrait de traitement en algorithmique, alors que le dernier le dernier point relève de ce que l'on nomme la programmation informatique.

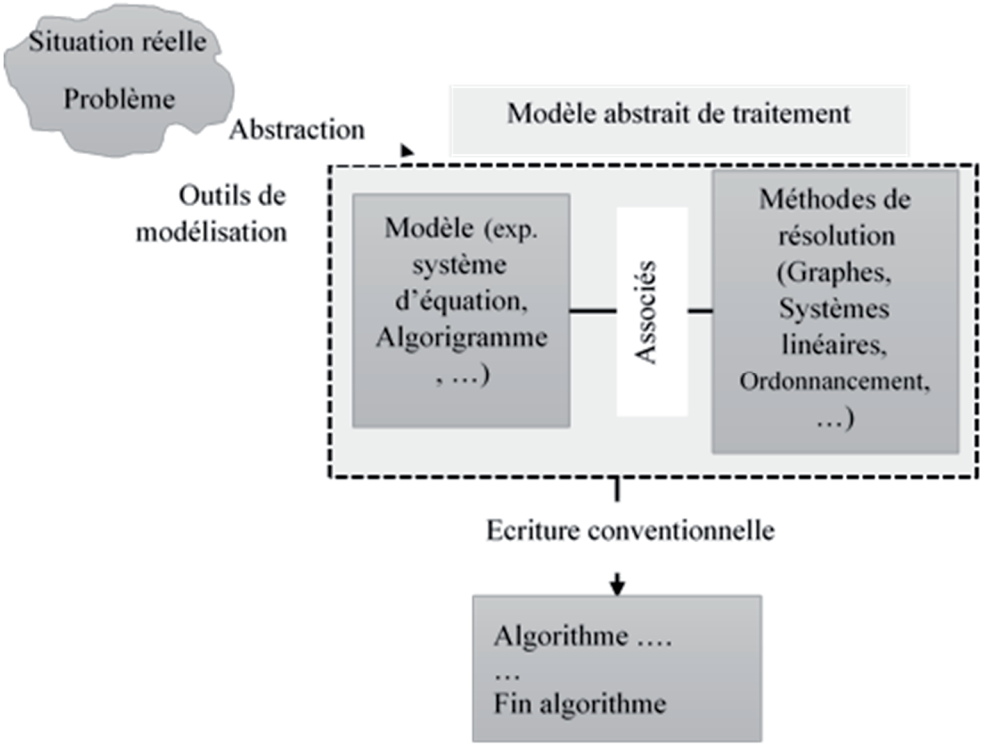


Figure 3. *Modèle abstrait de traitement*

2.1 Rappels et définitions

L'informatique désigne les sciences et techniques du traitement automatisé de l'information, c'est à dire des données (informations structurées).

Proposer une application ou une solution informatique pour la résolution d'un problème qu'il soit ponctuel ou répétitif, dans un contexte constant ou changeant, revient à :

1. définir quelle information à traiter : ce qui implique une maitrise de l'encodage et de la représentation des données (systèmes de numérotation de données). Nous abordons ce thème dans la suite de ce chapitre ;
2. définir comment traiter l'information : ce qui implique la définition d'un algorithme de traitement performant c'est-à-dire le meilleur (optimisation du temps de réponse, du temps de résolution, de la mémoire intermédiaire

requise, …) et le plus adéquat possible (réponse attendue). Les algorithmes en particulier et l'informatique en général sont adossés à des bases mathématiques à savoir l'algèbre de Boole. Nous abordons ces thèmes (c'est-à-dire Algèbre de Boole et Algorithmique) de façon plus étendue dans la suite de ce chapitre ;

3. faire exécuter un algorithme par une machine : ceci relève de la programmation informatique. Nous aborderons également ce thème les chapitres suivants en vue d'applications algorithmiques. A ce titre nous utiliserons le langage de programmation Python qui est un langage de script.

Dans une démarche d'informatisation (automatisation d'une résolution de problème), il y a donc trois phases à caractériser :

1. représentation des données ;
2. algorithme (de traitement) ;
3. programmation (automatisation).

Mais d'ores et déjà, et pour une maitrise avisée de l'environnement algorithmique, nous expliquons ci-dessous le processus d'exécution d'un algorithme par une machine (un processeur).

2.2 Processus d'exécution d'un algorithme

Dans ce paragraphe (2.2), nous utiliserons exclusivement en texte et bibliographie les productions de *France-IOI* comme référence.

2.2.1 Définition

Un langage informatique est destiné à décrire l'ensemble des actions consécutives qu'un ordinateur doit exécuter. Ceci implique, l'apprentissage, la mise en œuvre puis la pratique du langage choisi, sa syntaxe, son écriture, son débogage et son exécution machine.

2.2.2 Langage machine

Le langage machine, le seul compréhensible par la machine, est le langage utilisé par le processeur (qui utilise le code binaire, suite de 0 et de 1, peu compréhensible par un humain). Ceci implique la connaissance avisée du codage en langage machine (alphabet {0, 1}), son algèbre (Algèbre de Boole) puis les modes de raisonnement (Algèbre des propositions).

2.2.3 Langage assembleur

Le langage assembleur est un langage intermédiaire, proche du langage machine, mais plus « lisible » par un humain. Le langage assembleur est dépendant du processeur et donc il est donc non portable.

Cette acquisition n'est pas vraiment utile pour les non informaticiens.

2.2.4 Langage informatique

Un langage informatique est par définition différent du langage machine. Un langage informatique dispose et est régit par une grammaire syntaxique.

Un programme informatique ou encore un code source correspond à la traduction d'un ou plusieurs algorithmes par un programmeur ou développeur (qu'il soit Homme et/ou Machine) en utilisant comme traducteur un éditeur de langage informatique.

Les mots d'un langage machine sont écrits à l'aide de l'alphabet {0 ; 1} et sont codés dans le système de numérotation binaire et leur codage est conforme à l'algèbre de Boole.

2.2.5 Langage interprété

Il faut donc le traduire pour le rendre intelligible du point de vue du processeur. Un programme écrit dans un langage interprété a besoin d'un programme auxiliaire (l'interpréteur) pour traduire au fur et à mesure les instructions du programme.

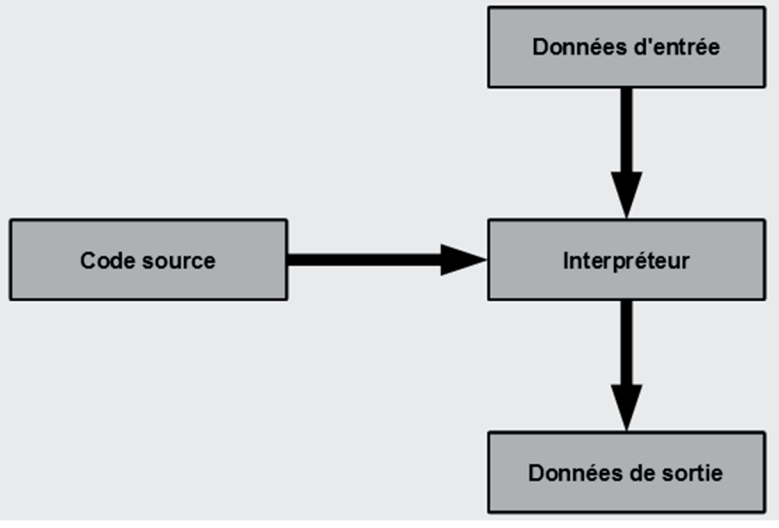


Figure 4. *Langage Interprété*

En effet, le code source (celui que vous écrivez) est interprété, par un logiciel qu'on appelle *interpréteur*. Celui-ci va utiliser le code source et les données d'entrée pour calculer les données de sortie. L'interprétation du code source est un processus « pas à pas » : l'interpréteur va exécuter les lignes du code une par une, en décidant à chaque étape ce qu'il va faire ensuite.

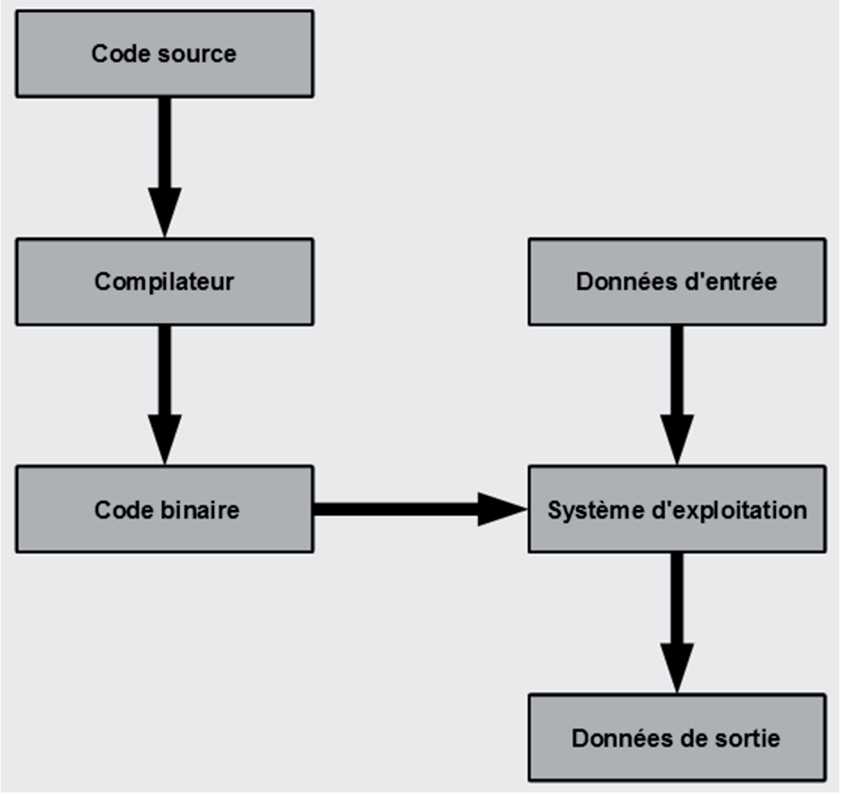


Figure 5. *Langage Compilé*

2.2.6 Langage compilé

Un programme écrit dans un langage dit compilé va être traduit une fois pour toutes par un programme annexe, appelé compilateur, afin de générer un nouveau fichier qui sera autonome, c'est-à-dire qui n'aura plus besoin d'un programme autre que lui pour s'exécuter ; on dit d'ailleurs que ce fichier est exécutable.

Un programme écrit dans un langage compilé a comme avantage de ne plus avoir besoin, une fois compilé, de programme annexe pour s'exécuter. De plus, la traduction étant faite une fois pour toute, il est plus rapide à l'exécution.

Toutefois il est moins souple qu'un programme écrit avec un langage interprété car à chaque modification du fichier source (fichier intelligible par l'*Homme* : celui qui va être compilé) il faudra le recompiler pour que les modifications prennent effet.

D'autre part, un programme compilé a pour avantage de garantir la sécurité du code source. En effet, un langage interprété, étant directement intelligible (lisible), permet à n'importe qui de connaître les secrets de fabrication d'un programme et donc de copier le code voire de le modifier. Il y a donc risque de non-respect des droits d'auteur.

D'autre part, certaines applications sécurisées nécessitent la confidentialité du code pour éviter le piratage (transaction bancaire, paiement en ligne, communications sécurisées, etc.).

2.3 Utilisation des langages de programmation

Voici ci-dessous une liste non exhaustive des langages de programmation les plus utilisés ainsi que leurs domaines d'application.

Langage	**Application**	**Compilé / Interprété**
Java	Programmation Orientée Objet, Internet	Compilé
C#	Programmation Système Orientée Objet	Compilé
C++		
Python	Domaine scientifique, Langage de script, Enseignement	Compilé, Interprété

Tableau 1. *Liste non exhaustive des langages de programmation*

Langage de programmation	**Classement en février 2020**	**Classement en février 2019**	**Évolution**
Java	1	1	=
C	2	2	=
Python	***3***	***3***	**=**
C++	4	4	=
C#	5	7	2
Visual Basic .NET	6	5	-1
JavaScript	7	6	-1
PHP	8	8	=
SQL	9	9	=

Langage de programmation	Classement en février 2020	Classement en février 2019	Évolution
Swift	10	20	10
Go	11	18	7
Assembly language	12	11	-1
R	13	15	2
D	14	23	9
Ruby	15	16	1
MATLAB	16	12	-4
PL/SQL	17	21	4
Delphi/Object Pascal	18	14	-4
Perl	19	13	-6
Objective-C	20	10	-10

Tableau 2. *Les langages les plus populaires d'après l'index TIOBE - 2020*

3 Représentation des données

3.1 Les systèmes de numérotation

Usuellement, les nombres sont représentés en base 10 (dix), c'est-à-dire avec les dix chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.

Il existe d'autres systèmes de représentation (base 2, base 3, base 8, ….). On dit systèmes de numérotation.

Le système de numérotation binaire ne représente que deux états possibles : le circuit est allumé (1) ou éteint (0). On peut aussi les représenter avec les variables logiques VRAI ou FAUX. Dans le système binaire, les tables d'addition et de multiplication sont très simples.

Système	Base	Symboles utilisés
Binaire	2	0 1
Ternaire	3	0 1 2
Octal	8	0 1 2 3 4 5 6 7
Décimal	10	0 1 2 3 4 5 6 7 8 9
Hexadécimal	16	0 1 2 3 4 5 6 7 8 9 A B C D E F

Tableau 3. *Quelques bases de numérotation*

Nous reviendrons plus en détail sur le système binaire dans la suite consacrée à l'algèbre de Boole. Voici ci-dessous (*Tableau 4.*) les valeurs équivalentes exprimées dans les principales bases : base 2 (binaire), base 8 (octale), base 10 (décimale) et base 16 (hexadécimale).

Exemple :

$11110_2 = 30_{10} = 36_8 = 1E_{16}$

Binaire	Décimale	Octale	Hexadécimale
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	10	8
1001	9	11	9
1010	10	12	A
1011	11	13	B
1100	12	14	C
1101	13	15	D
1110	14	16	E
1111	15	17	F
10000	16	20	10
10001	17	21	11
10010	18	22	12
10011	19	23	13
10100	20	24	14
…	…	…	…
11001	25	31	19
11010	26	32	1A
11011	27	33	1B

Binaire	Décimale	Octale	Hexadécimale
11100	28	34	1C
11101	29	35	1D
11110	30	36	1E
1111	31	37	1F
100000	32	40	20

Tableau 4. *Valeurs équivalentes exprimées dans les principales bases*

3.2 Décomposition d'un nombre

La décomposition d'un nombre met en évidence les facteurs multiplicatifs et les puissances de la base de décomposition.

3.2.1 Exemple 1 de décomposition d'un nombre

Décomposition de 4321 en base 10 :

$4321_{10} = 4 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$

3.2.2 Exemple 2 de décomposition d'un nombre

Décomposition de 0.123 en base 10 :

$0.123_{10} = 1 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3}$

De façon générale, un nombre N écrit dans la base b s'écrit sous forme décomposée :

$$N_b = (C_{n-1} \times b^{n-1}) + (C_{n-2} \times b^{n-2}) + \ldots + (C_1 \times b^1) + (C_0 \times b^0) + (C_{-1} \times b^{-1}) + (C_{-2} \times b^{-2}) + \ldots + (C_{-m} \times b^{-m})$$

où :

N : est un nombre en base b

C : un chiffre ou un symbole composant le nombre N

b : la base

n : le rang du symbole le plus à gauche du point

-m : le rang du symbole le plus à droite du point

3.2.3 Exercice d'application

Trouver la valeur en base 10 de $E7AB3_{16}$.

$E7AB3_{16} = (14 \times 16^4 + 7 \times 16^3 + 10 \times 16^2 + 11 \times 16^1 + 3 \times 16^0)_{10} = 948915_{10}$