Bruno Hennart

DÉBUTER EN ALGORITHMIQUE ET EN PROGRAMMATION PYTHON



ellipses

DÉBUTER EN ALGORITHMIQUE ET EN PROGRAMMATION PYTHON

Bruno Hennart

ellipses

Conception graphique couverture : Nathalie FOULLOY
Conception graphique intérieur : Charline PINTO
Mise en pages : Nord Compo

ISBN 9782340-104761
Dépôt légal : mai 2025
©Ellipses Édition Marketing S.A.
8/10 rue la Quintinie 75015 Paris



Le Code de la propriété intellectuelle et artistique n'autorisant, aux termes des alinéas 2 et 3 de l'article L. 122-5, 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 » (alinéa 1er de l'article L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

www.editions-ellipses.fr

AVANT-PROPOS

Python est un langage dont le code est accessible au public, dit langage « open source ». Il tire avantage de cette propriété car il est l'un des langages les plus répandus et les plus populaires. Il est, en outre enseigné au lycée.

Vous êtes lycéenne, lycéen, ou vous débutez en programmation, cet ouvrage va vous permettre d'appréhender l'étude et la résolution de problèmes par l'informatique en utilisant l'algorithmique. Il vous permettra d'analyser le problème et trouver une solution à celui-ci. La programmation du résultat de cette analyse en Python donnera vie à la solution du problème.

Ce livre propose un éventail des connaissances essentielles en algorithmique et en programmation Python au fur et à mesure de la lecture des différents chapitres.

Cet objectif d'apprentissage est facilité par des mises en application des concepts étudiés au travers d'exemples et d'exercices pratiques, cours, corrigés et commentés où les algorithmes proposés sont traduits en Python. Certains de ces exercices font également le lien avec des connaissances en mathématiques vues au lycée.

Ces différentes ressources sont non seulement des outils d'apprentissage mais aussi des modèles et des sources d'inspiration pour vous aider à aborder les enseignements au lycée ou pour débuter la programmation.

TABLE DES MATIÈRES

1 Définition de l'algorithmique 7

2 Les variables et les types 13

3 Écrire un algorithmique 25

4 Programmation d'un algorithme 41

5 Les instructions conditionnelles 49

6 Les instructions itératives (partie 1) 69

7 Les instructions itératives (partie 2) 83

8 Les tableaux 95

9 Fonctions et procédures 111

10 La récursivité 133

11 Debugging 143

12 Notion de classes et d'héritage 159

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.

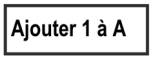
2. L'ORDINOGRAMME

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

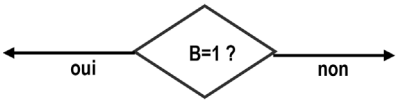
- Le début et la fin d'un programme sont matérialisés par des figures ovales :



- 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.

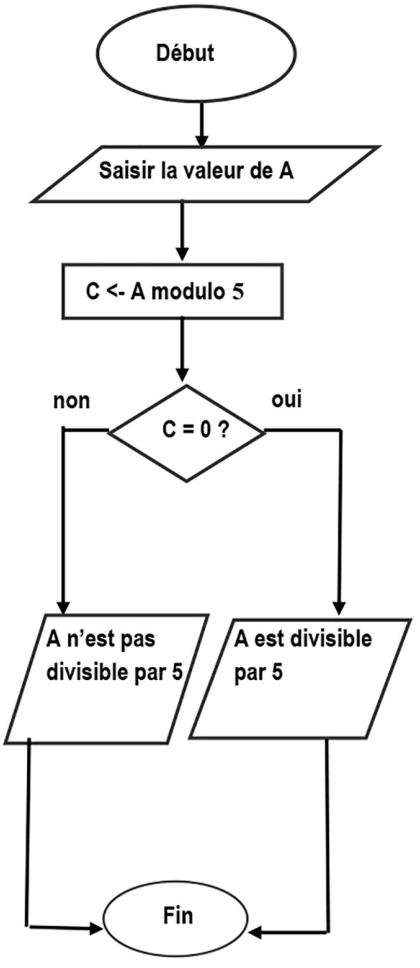
Nous allons utiliser la fonction modulo qui renvoie le reste de la division de A par 5.

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 :

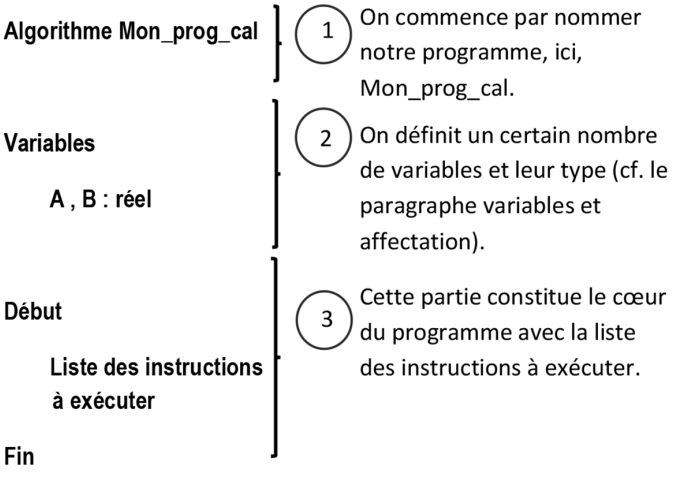


1. Etiquette marquant le début du programme
2. On commence par demander à l'utilisateur de saisir un nombre A.
3. On effectue l'opération modulo de A par 5 qui donne le reste de la division et on met le résultat dans la variable C.
4. On teste la valeur du reste de la division de A par 5 (affectée à 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.

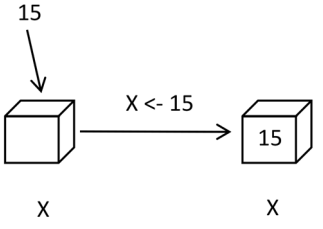
Quand on donne une valeur à une variable X, on écrit :

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 :