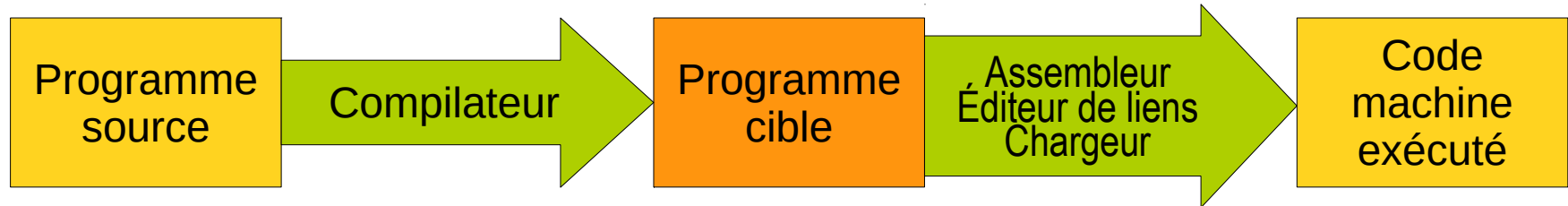


# Traduction et exécution de programmes

- Langages **compilés** (p. ex. Fortran, C)

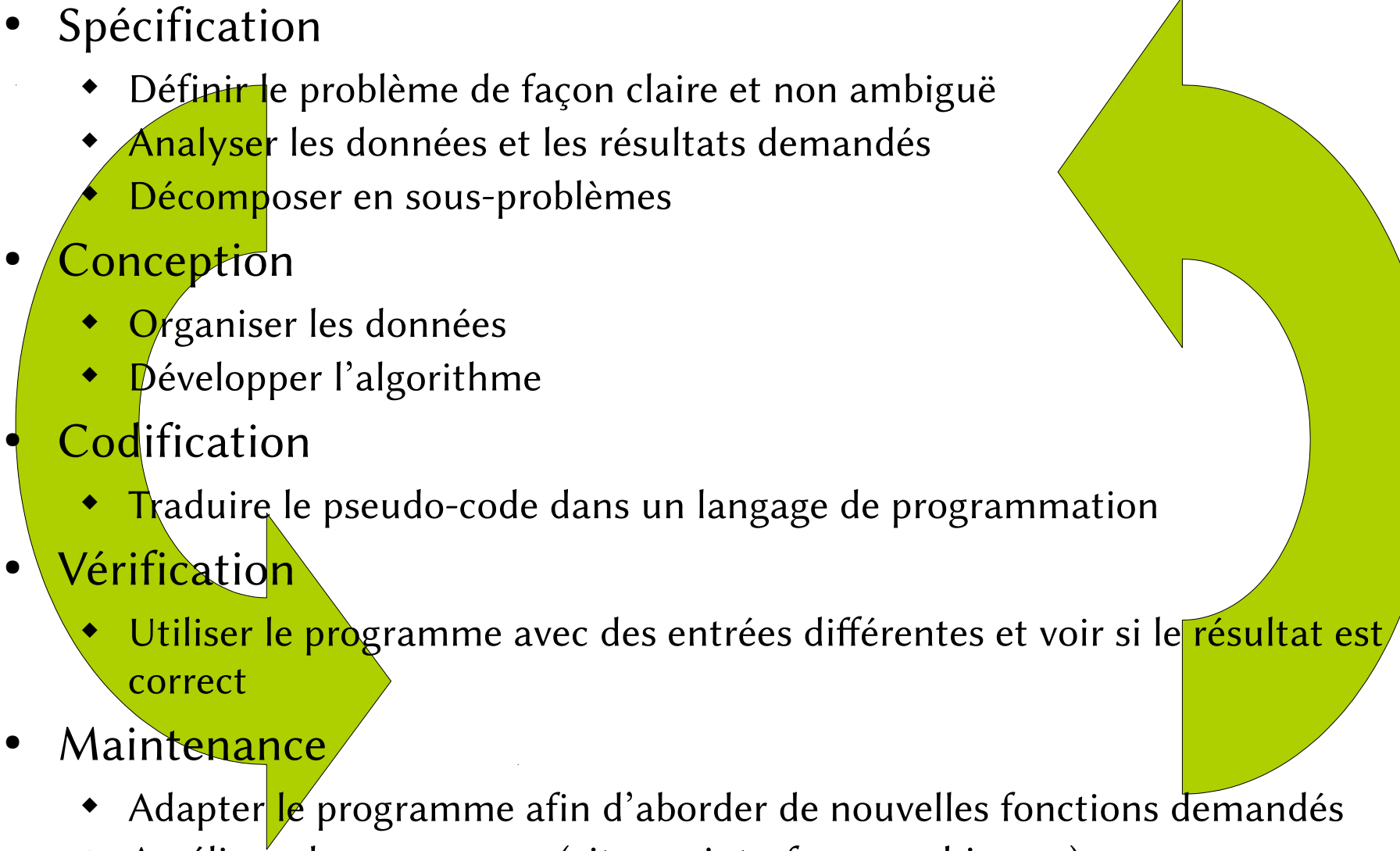


- Langages **interprétés** (p. ex. Basic, Matlab)



- ♦ Les commandes peuvent être entrées en ligne (mais ça dépend de l'environnement de programmation), au fur et à mesure de l'exécution
- ♦ C'est plus simple du côté utilisateur
- ♦ L'exécution paraît immédiate
- ♦ En fait le code machine n'est pas optimal et l'interprétation peut être compliquée alors l'exécution prends plus de temps, de mémoire etc.

# Cycle de vie d'un programme

- **Spécification**
    - ◆ Définir le problème de façon claire et non ambiguë
    - ◆ Analyser les données et les résultats demandés
    - ◆ Décomposer en sous-problèmes
  - **Conception**
    - ◆ Organiser les données
    - ◆ Développer l'algorithme
  - **Codification**
    - ◆ Traduire le pseudo-code dans un langage de programmation
  - **Vérification**
    - ◆ Utiliser le programme avec des entrées différentes et voir si le résultat est correct
  - **Maintenance**
    - ◆ Adapter le programme afin d'aborder de nouvelles fonctions demandés
    - ◆ Améliorer le programme (vitesse, interface graphique...)
- 

# Documentation

- Programmer, c'est communiquer
  - ◆ avec l'interpréteur, compilateur...
- mais aussi
  - ◆ avec les autres
  - ◆ avec soi-même (dans le futur)
- **Important : bien documenter son travail**
  - ◆ désignations évocatrices qui décrivent le contenu ou le rôle d'une variable, fonction... : *hauteur* et *calculer\_longueur* au lieu de *a* et *fonction1*
  - ◆ algorithmes en pseudo-code ou en diagrammes
  - ◆ indentation du code
    - ▶ si  $a = 2$  alors  
    afficher « a est égal 2 »  
sinon  
    afficher « a n'est pas égal 2 »
  - ◆ commentaires dans le code
    - ▶ si  $b=0$ , l'equation n'a pas de solution
    - ▶ formule a consulter avec Mme la comptable

## 2. Éléments d'un langage de programmation ; introduction au langage Matlab et à l'environnement Octave

Syntaxe et instructions

Données et types simples

Types composés : tableaux, chaînes de caractères

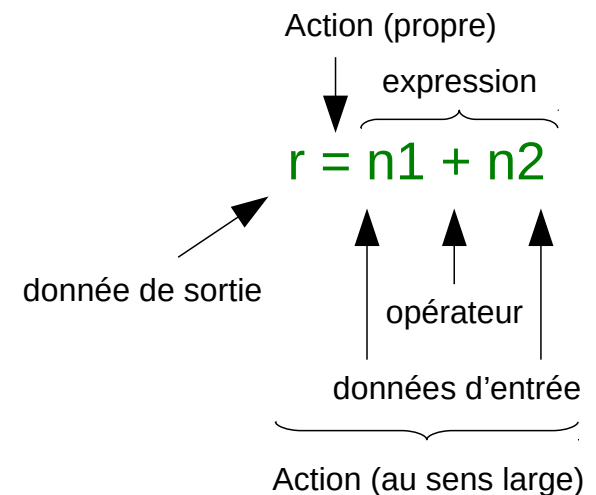
Affectation

Opérateurs arithmétiques

Obtention et affichage de données

# Éléments d'un langage de programmation

- **Données** ; classifiées selon leur rôle comme :
  - ◆ données d'entrée (entrées, arguments, paramètres [d'entrée], données)
  - ◆ données internes (intermédiaires, auxiliaires)
  - ◆ données de sortie (sorties, résultats, valeurs de retour, paramètres de sortie)
- **Expressions**
  - ◆ décrivent les relations entre les données
  - ◆ principalement à l'aide d'**opérateurs**
- **Actions**
  - ◆ commandes, instructions, procédures, fonctions, méthodes...
  - ◆ **affectation** – action spécifique, souvent notée avec un opérateur et même considérée opérateur
- **Structures de contrôle**
  - ◆ modifient le cours d'exécution du programme : conditions, répétitions...



*Nous allons nous baser sur Matlab/Octave mais nous essaierons d'aborder (dans la mesure du possible) tous les langages en général*

# Syntaxe

- Tout comme un langage humain, chaque langage de programmation possède sa propre syntaxe
  - ◆ plus systématique, formalisée et stricte que celles des langages humains
  - ◆ pour que les codes soient plus faciles à analyser par l'ordinateur
  - ◆ des synonymes, homonymes, alternatives, exceptions etc. sont très rares
- La **syntaxe précise** :
  - ◆ comment on saisit des valeurs et désigne des données, des actions...
  - ◆ comment on compose des expressions à l'aide d'opérateurs, fonctions...
  - ◆ comment on compose des instructions
  - ◆ comment on compose un programme et ses parties plus petites
- Le « comment » veut dire :
  - ◆ quels caractères on utilise pour un élément propre du langage
  - ◆ quels caractères servent de séparateurs, discriminants, marques de début, fin ou continuation...
  - ◆ dans quelle ordre ces éléments peuvent ou doivent apparaître
  - ◆ pour que l'ordinateur comprenne quels actions il doit exécuter

# Spécificités de Matlab concernant les instructions

- **Instruction** est l'étape élémentaire d'un programme ou le fragment élémentaire d'un code
- La fin de ligne est équivalente à la fin de l'instruction
  - ♦ Pour continuer une instruction dans une ligne suivante, utiliser `...` ou `\`
  - ♦ `a = ...` ou `a = \`  
`b + 2` `b + 2`
- Affichage du résultat
  - ♦ Si une instruction finit avec le point-virgule `;`, son résultat n'est pas affiché
  - ♦ Dans tous les autres cas, le résultat est affiché
- Autre application du point-virgule
  - ♦ Il sert à placer plusieurs instructions dans une ligne  
`a = 0; b = pi; c = sin(a) + cos(b);`
  - ♦ Si on veut afficher les résultats d'instructions particulières, alors on les sépare avec la virgule `,`  
`a = 0, b = pi, c = sin(a) + cos(b)`
- Si aucune variable n'est affectée explicitement, le résultat est enregistré dans la variable spéciale *ans*
  - ♦ Après chaque instruction, son contenu est remplacé

# Données

- Une **donnée** peut être considérée comme une boîte contenant une information (**valeur**) d'une certaine forme (**type**), caractérisée par une étiquette (**nom**)
- Une donnée peut être :
  - ♦ **constante** – si sa valeur ne peut changer pendant l'exécution du programme
  - ♦ **variable** – si sa valeur peut être changée pendant l'exécution du programme
- Pour les **noms (désignations)**, dans la plupart des langages il existe de très similaires :
  - ♦ règles :
    - ▶ peut être composée seulement des caractères ASCII **a...z**, **A...Z**, **0...9** et **\_**
    - ▶ ne peut pas commencer avec un chiffre
  - ♦ conventions :
    - ▶ est écrite en minuscules (majuscules sont réservées pour des constantes)
    - ▶ si composée de plusieurs mots, on les délimite avec le tiret bas ou avec des majuscules et on omet les prépositions, p. ex. *adresseDomicile*, *AdresseDomicile* ou *adresse\_domicile* (en non *adressedudomicile*)
    - ▶ la convention de Matlab, c'est *adresseDomicile* ou *adresse\_domicile*



# Spécificités de Matlab concernant les données

- Pas de nécessité de **déclarer** (annoncer) les variables et leurs types
  - ♦ toute variable est en fait un tableau de nombres
  - ♦ matrice, vecteur, nombre, caractère, chaîne de caractères...
- Matlab discerne les **majuscules** et les **minuscules** dans les noms des variables (et fonctions)
- Toute variable numérique est **réelle**
  - ♦ pas de nombreux types différents comme entier petit, entier grand, réel petite précision, réel grande précision...
  - ♦ plus facile à utiliser, parfois cause des problèmes
  - ♦ moins économique en termes de mémoire et vitesse d'opérations
- Toute variable peut être **complexe** ( $a + b*i$ )
- Très simple d'utiliser des **matrices** (et vecteurs)
- Une **chaîne de caractères** est un vecteur contenant les codes ASCII de ces caractères un par un

# Types de données

- **Types fondamentaux** (simples) de données
  - 1) numérique
  - 2) alphanumérique
  - 3) logique
- En fait toute donnée est numérique parce que ce qu'un processeur ne travaille qu'avec des nombres (binaires)
- Mais ça se manifeste de manière différente dans de différents langages haut-niveau
  - ◆ logique : p. ex. **faux**  $\Leftrightarrow$  0, **vrai**  $\Leftrightarrow$  1
  - ◆ alphanumérique : p. ex. **A**  $\Leftrightarrow$  65 (selon le jeu de caractères utilisé)
    - ▶ ASCII = ISO 646 – code américain (1963), 128 caractères (0h...7Fh, 7 bits)
    - ▶ extensions de ASCII à 256 caractères (0h...FFh, 8 bits), p. ex. ISO 8859 – longtemps populaires mais problématiques : ne couvraient que de petits groupes d'alphabets et plusieurs jeux existaient pour un même groupe
    - ▶ ISO 10646 (Unicode) – 1 114 112 codes (0h...10FFFFh, 21 bits) mais seulement une partie (env. 100 000) est utilisée ce qui permet d'utiliser 8 bits pour latin basique et 16 bits pour la plupart des alphabets/syllabaires

# « Constantes » numériques de Matlab

- Nombres:
  - ♦ **pi** = 3,1416... le nombre  $\pi$
  - ♦ **i** =  $\sqrt{-1}$  l'unité imaginaire
  - ♦ **e** = 2,7183... la base du logarithme naturel
- Valeurs logiques
  - ♦ **true** = 1 vrai
  - ♦ **false** = 0 faux
- Attention :
  - ♦ En Matlab, il n'y a pas de vraies constantes ; ce sont des variables avec valeurs prédéfinies
  - ♦ Il est alors possible de modifier leurs valeurs
  - ♦ Jamais n'appellez vos variables comme les variables prédéfinies de Matlab
  - ♦ De même il est possible de changer les définitions des fonctions prédéfinies de Matlab
  - ♦ Jamais n'appellez vos fonctions comme les fonctions prédéfinies de Matlab

# Types composés – tableaux

- **Types composés** (évolués) de données :
  - ♦ tableaux
  - ♦ chaînes de caractères (chaînes) – genre de tableau
  - ♦ d'autres **structures de données** – enregistrements, listes, arbres, piles...
- Un **tableau** est une suite de données du même type
  - ♦ Il correspond à la notion de matrice ou vecteur en mathématiques
  - ♦ Tout élément possède une ou plusieurs **indices** (en fonction du nombre des dimensions du tableau qui peut être 1, 2 ou plus)
  - ♦ Numérotation des éléments la plus évidente : 1, 2, 3, 4... (Matlab)
  - ♦ mais beaucoup de langages commencent par 0 – c'est plus efficace
- Un **enregistrement** est un ensemble de données qui peuvent être de types différents
  - ♦ données d'un employé :      numéro d'identification – nombre  
prénom, nom – chaîne de caractères  
date de naissance – nombre(s) ou chaîne(s)