

DUT MMI – IUT de Marne-la-Vallée  
26/09/2018  
M1202 - Algorithmique

***Cours 1***  
***Introduction aux algorithmes***

# Organisation pratique

---

- **Intervenants**

- Philippe Gambette et Samy Dindane

- **Contact**

- Courriel : philippe.gambette@u-pem.fr (M1202 dans le sujet du courriel)
  - Avant ou après le cours

- **Matériel**

- Ordinateur portable : pas pendant les cours (a priori), à discuter pour les TD.
  - Pas de téléphone portable pendant cours/TD/TP
  - Salles informatiques : ne pas manger, ne pas boire, ne pas débrancher les câbles

# Organisation pratique

- **Déroulement des enseignements**

- Pages web du cours :

- page publique : <http://tinyurl.com/M1202-2018S1>

- page privée : sur eLearning (à venir)

- Séparation cours/TP/TD :

- **nouvelles méthodes de travail**

- **distinguer ce qui est important, à retenir**

- **savoir où retrouver l'information**

- En général, distribution de notes de cours à compléter

- En général, distribution de corrigés des TD (les demander si besoin) :

- **refaire les exercices !**

# Organisation pratique

- **Notes et devoirs**

- Interrogations QCM en début de cours ou TD (signalement des absences pour rattrapage)
- Travaux à la maison sur e-Learning

- **Note finale**

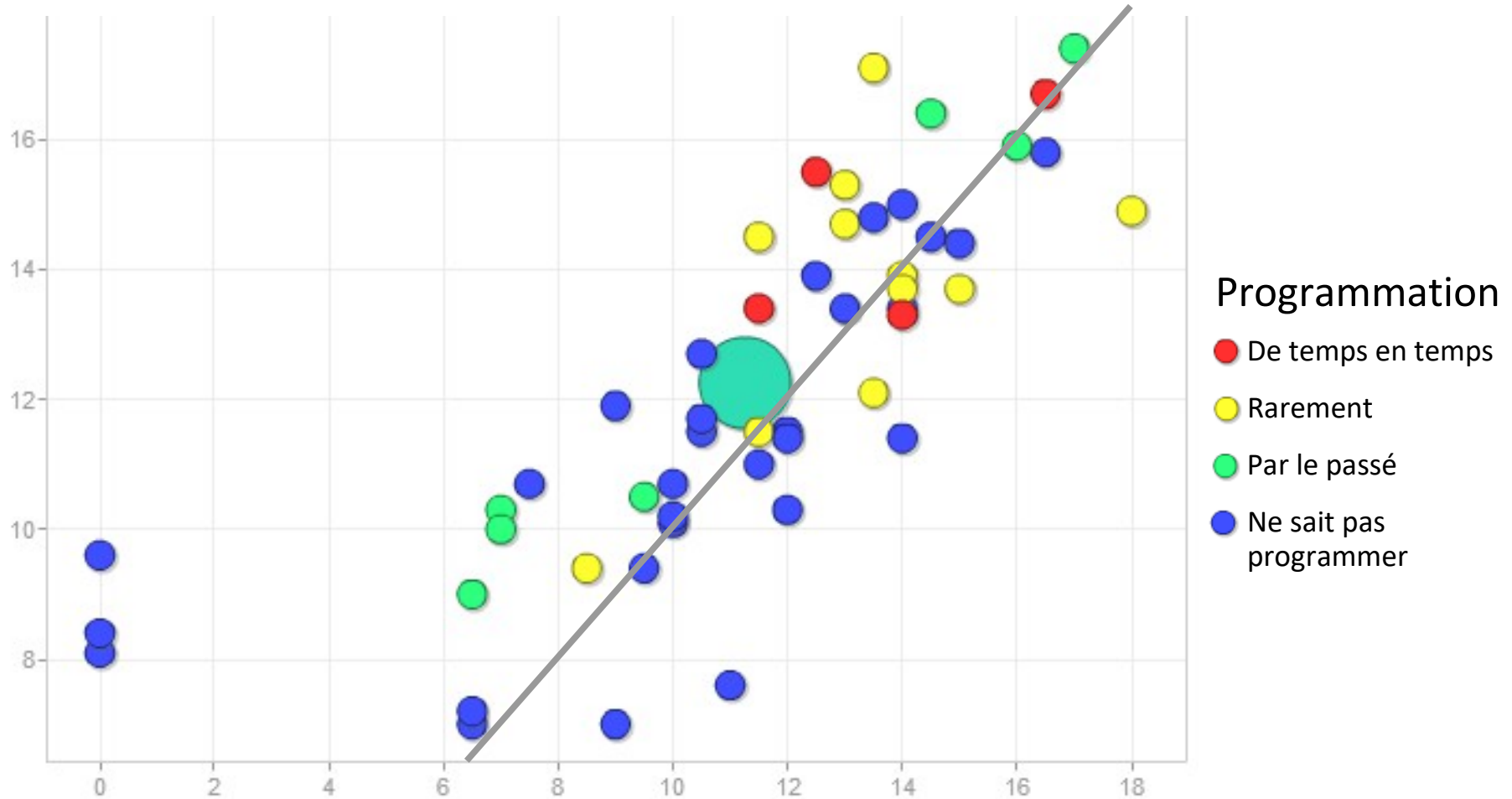
- Préviation : environ 2/3 “compétences”, environ 1/3 “motivation”
- Compétences : 2/3 devoir final (26 novembre 2018), 1/3 QCM
- Motivation : tests d’auto-apprentissage, remplissage du cours à trous, note de TP ?

- **Exercices supplémentaires d'entraînement**

- Sur demande, par courriel
- Sur demande, possibilité d'organiser une séance d'exercices ou de préparation au devoir final.

# Promo 2014-2015

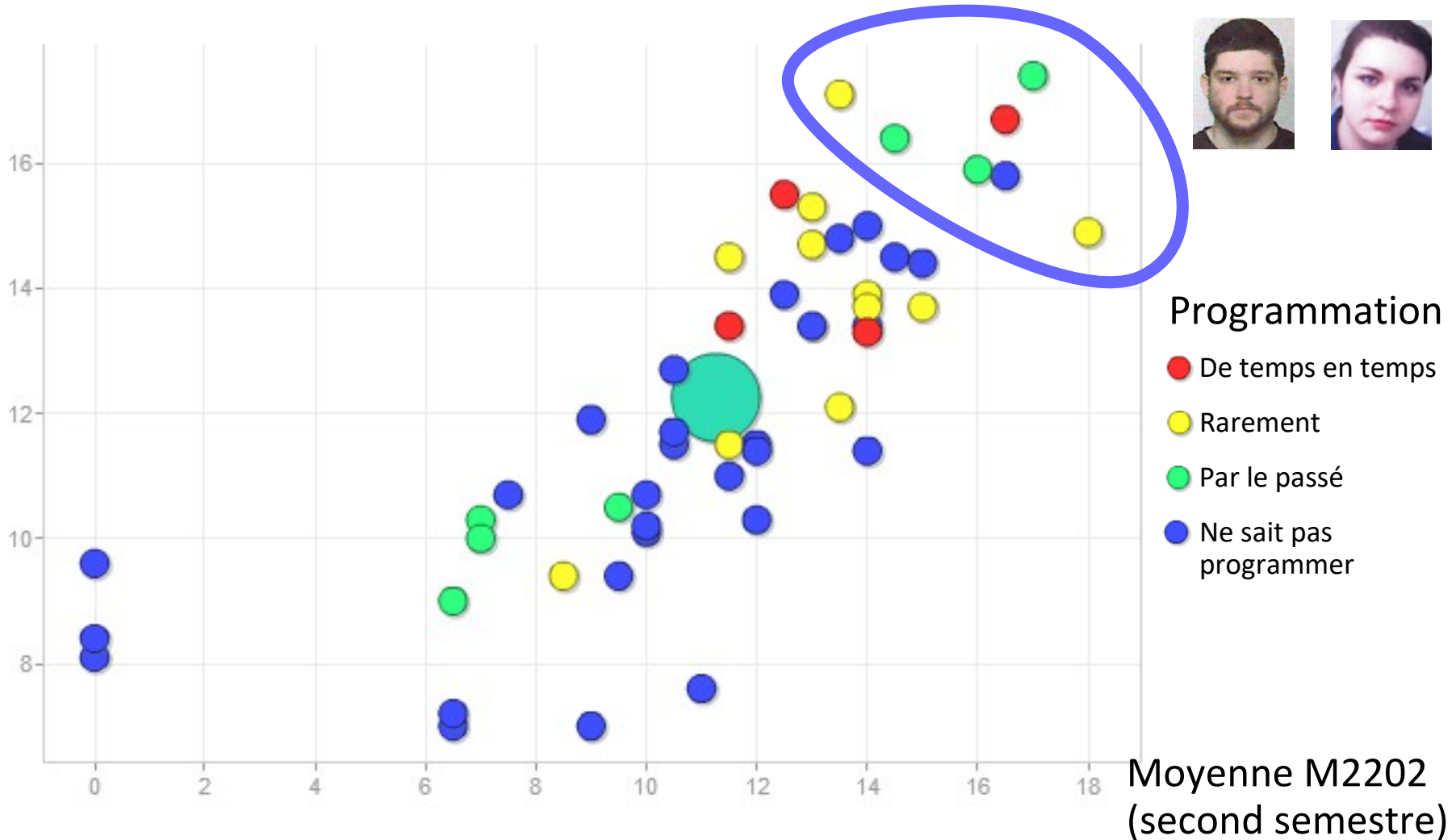
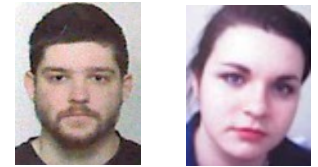
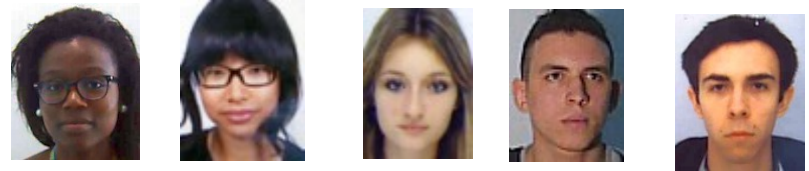
Moyenne M1202  
(premier semestre)



Moyenne M2202  
(second semestre)

# Organisation pratique

Moyenne M1202  
(premier semestre)



# Sources

---

- *Le livre de Java premier langage*, d'A. Tasso
- *JavaScript et les données du Web*, de R. Jeansoulin
- <http://www.pise.info/algo/introduction.htm>
- Cours INF120 de J.-G. Luque
- <http://serecom.univ-tln.fr/cours/index.php/Algorithmie>
- Cours de J. Henriet : <http://julienhenriet.olymp-network.com/Algo.html>

# Quelles compétences à la fin du semestre ?

## **C1) Être capable de comprendre le fonctionnement d'un algorithme :**

- C1a) en identifiant les différents éléments de base de l'algorithme
- C1b) en simulant son comportement à l'aide d'une trace
- C1c) en interprétant une courte description en français de ses spécifications

## **C2) Être capable de concevoir un algorithme pour résoudre un problème :**

- C2a) en analysant le problème :
  - pour comprendre les besoins
  - pour le découper éventuellement en sous-problèmes
- C2b) en écrivant un algorithme correctement structuré :
  - avec des entrées et des sorties correctement définies
  - en utilisant des structures conditionnelles
  - en utilisant des répétitions d'opérations (à l'aide de boucles ou de la récursivité)
  - en faisant appel à d'autres algorithmes dont les spécifications sont connues, notamment pour interagir avec l'utilisateur
  - en découpant le code de manière modulaire en divers composants indépendants et réutilisables

## **C3) Être capable de comprendre le fonctionnement d'un code Javascript :**

- C3a) en identifiant les divers éléments de structure du code Javascript
- C3b) en identifiant les éventuelles erreurs qu'il contient par un processus de débogage

## **C4) Être capable d'écrire un programme Javascript :**

- C4a) en respectant la syntaxe Javascript
- C4b) en choisissant ou en respectant des conventions de nommage appropriées
- C4c) en testant le code obtenu (par compilation puis exécution)



# Pourquoi coder / programmer ?

C'est **facile**

→ enseigné en primaire depuis peu

C'est **utile**

→ la code au service de l'esthétique et de l'expérience utilisateur sur le web

C'est **efficace**

→ « ère des données »

Ça **paye bien**

→ besoin de développeurs

Ça **permet de comprendre le monde qui nous entoure**

→ classement APB ou Parcoursup, classement des résultats Google, sélection des actualités du fil Facebook ou Twitter, voitures sans conducteur, etc.

*Pour approfondir, la leçon inaugurale de Claire Mathieu, conceptrice avec Hugo Gimbert de l'algorithme de Parcoursup, au Collège de France en 2017 :*

*<https://www.college-de-france.fr/site/claire-mathieu/inaugural-lecture-2017-11-16-18h00.htm>*

# Plan du cours 1 – Introduction aux algorithmes

---

- Introduction aux algorithmes
  - À quoi sert un algorithme ?
  - Algorithme et programme
  - Enjeux de l'algorithmique
  - Composants d'un algorithme
- Variables et affectation

# La recette des crêpes

---

Les différences possibles entre recettes pour un même plat :

# La recette des crêpes

---

Les différences possibles entre recettes pour un même plat :

- Ingrédients (quantités, unités de mesure)
- Matériel utilisé
- Ordre des opérations, nombre d'opérations
- Cuisson, mode d'opération
- Nom de la recette
- Temps de préparation
- Source de la recette
- Style d'écriture
- Langue
- Photo du résultat

# La recette des crêpes

## Le site le plus simple pour faire la pâte à crêpes !

Oyé oyé, braves gens ! Bienvenue sur le site le plus simple pour faire la pâte à crêpe ! Cette recette facile de pâte à crêpe se transmet de bouches à oreilles et maintenant de Facebook en Facebook pour votre plus grand plaisir ! Vous allez adorer faire des crêpes.



Pâte à crêpe pour 15 crêpes:  
50g de beurre, 4 oeufs, 2 cuillères à café de sucre, 1 pincée de sel, 250g de farine et 1/2 litre de lait



Mettre l'ensemble des ingrédients dans un récipient sauf le beurre



Mélanger avec un fouet jusqu'à obtenir de la pâte liquide et sans grumeaux



Ajouter les 50g de beurre fondu: fondre au micro-onde, ça va plus vite! Pour plus de goût, ajouter de la fleur d'oranger ou du rhum...



Si possible, laisser reposer, puis étaler une dose de pâte dans une poêle chaude préalablement graissée



Laisser cuire à feu doux...



...puis retourner pour laisser cuire l'autre côté



Bon appétit !

# La recette des crêpes

## Le site le plus simple pour faire la pâte à crêpes !

Oyé oyé, braves gens ! Bienvenue sur le site le plus simple pour faire la pâte à crêpe ! Cette recette facile de pâte à crêpe se transmet de bouches à oreilles et maintenant de Facebook en Facebook pour votre plus grand plaisir ! Vous allez adorer faire des crêpes.



Pâte à crêpe pour 15 crêpes:  
50g de beurre, 4 oeufs, 2 cuillères à café de sucre, 1 pincée de sel, 250g de farine et 1/2 litre de lait



Mettre l'ensemble des ingrédients dans un récipient sauf le beurre



Mélanger avec un fouet jusqu'à obtenir de la pâte liquide et sans grumeaux



Ajouter les 50g de beurre fondu: fondre au micro-onde, ça va plus vite! Pour plus de goût, ajouter de la fleur d'oranger ou du rhum...



**Si possible**, laisser reposer, puis étaler **une dose** de pâte dans une poêle chaude préalablement graissée



Laisser cuire à feu doux...



**...puis** retourner pour laisser cuire l'autre côté



Bon appétit !

# La recette des crêpes

## Le site le plus simple pour faire la pâte à crêpes !

Oyé oyé, braves gens ! Bienvenue sur le site le plus simple pour faire la pâte à crêpe ! Cette recette facile de pâte à crêpe se transmet de bouches à oreilles et maintenant de Facebook en Facebook pour votre plus grand plaisir ! Vous allez adorer faire des crêpes.



Pâte à crêpe pour 15 crêpes:  
50g de beurre, 4 oeufs, 2  
cuillères à café de sucre, 1  
pincée de sel, 250g de  
farine et 1/2 litre de lait



Mettre l'ensemble des  
ingrédients dans un  
récipient sauf le beurre



Mélanger avec un fouet  
jusqu'à obtenir de la pâte  
liquide et sans grumeaux



Ajouter les 50g de beurre  
fondu: fondre au micro-  
onde, ça va plus vite! Pour  
plus de goût, ajouter de la  
fleur d'oranger ou du  
rhum...



**Si possible**, laisser reposer,  
puis étaler **une dose** de pâte  
dans une poêle chaude  
préalablement graissée



Laisser cuire à feu doux...



**...puis** retourner pour  
laisser cuire l'autre côté



Bon appétit !

# L'“algorithme des crêpes”

**Ingrédients** : beurre, oeufs, sachets de sucre vanillé, farine, lait, sel

**Récipients** : saladier, verre mesureur, poêle, assiette

**Opérations de base** : *mettre dans un récipient*, *mélanger*, *attendre pendant ... minutes*, *retourner*, *laisser cuire pendant ... minutes*

**Algorithme des crêpes** :

**Mettre** 4 oeufs **dans** le saladier

**Mettre** 1 sachet de sucre vanillé **dans** le saladier

**Mettre** 250 g de farine **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 0,5 litre de lait **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 50 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 1 minute

**Mettre** le contenu de la poêle **dans** le saladier

**Mélanger** le contenu du saladier

**Attendre pendant** 60 minutes

**Mettre** 5 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 0.5 minute

**Tant que** le saladier n'est pas vide :

**Mettre** 5 cL du contenu du saladier **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** la poêle

**Laisser cuire** la poêle **pendant** 2 minutes

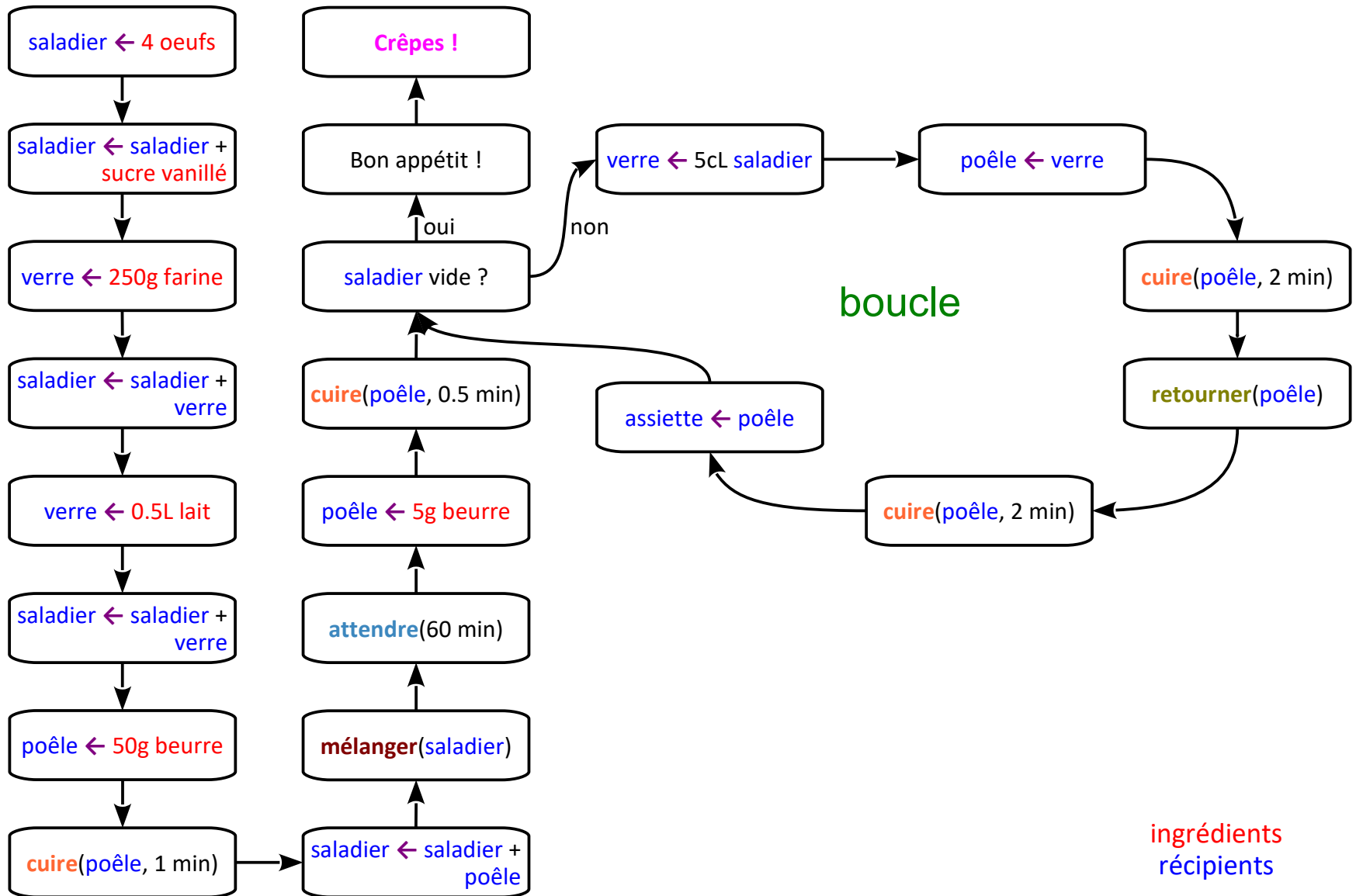
**Retourner** le contenu de la poêle

**Laisser cuire** la poêle **pendant** 2 minutes

**Mettre** le contenu de la poêle **dans** l'assiette



# Organigramme de la recette des crêpes



# À quoi sert un algorithme ?

---

- **À décrire les étapes de résolution d'un problème :**
  - de façon structurée et compacte
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

# À quoi sert un algorithme ?

- À décrire les **étapes** de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

“**étapes**” aussi appelées “**pas de l'algorithme**”

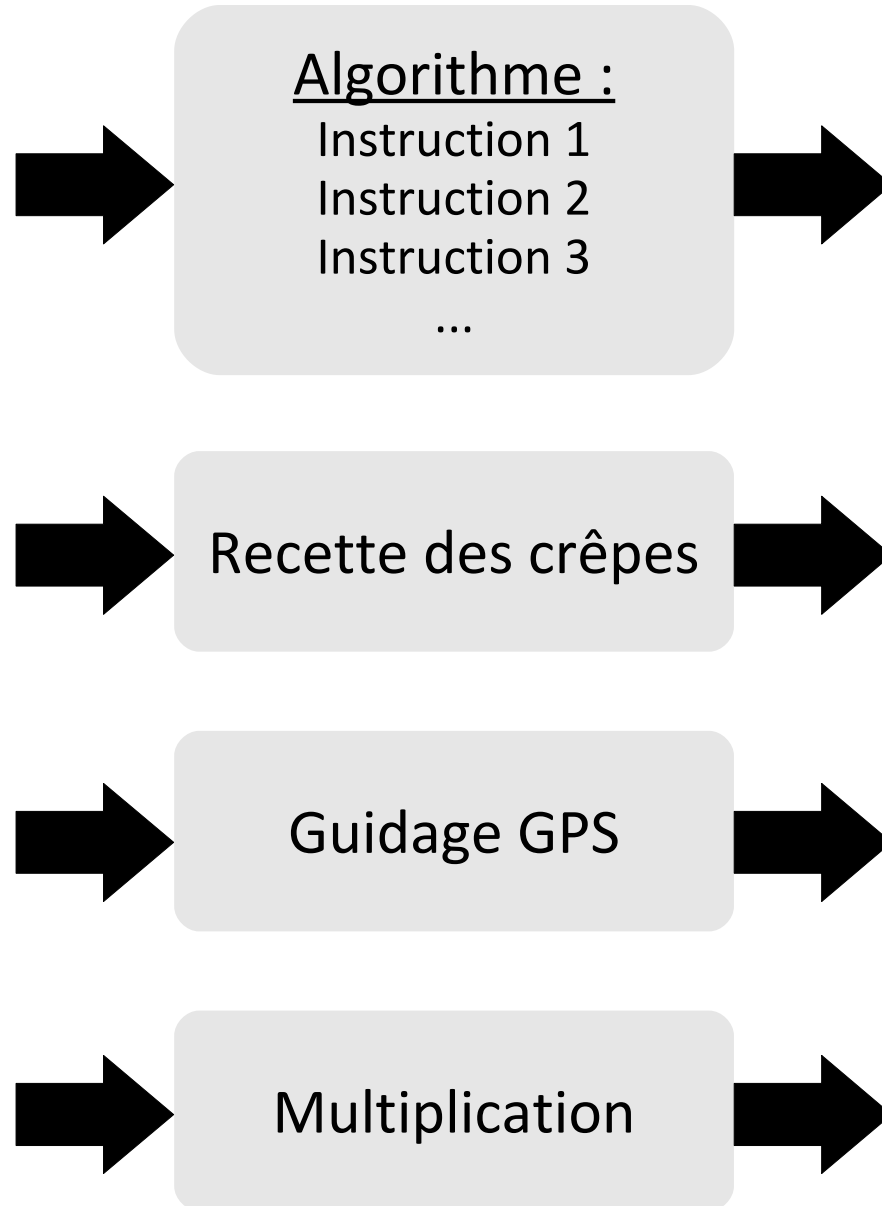
# À quoi sert un algorithme ?

- À décrire les étapes de **résolution d'un problème** :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

Les **données** du problème en **entrée**

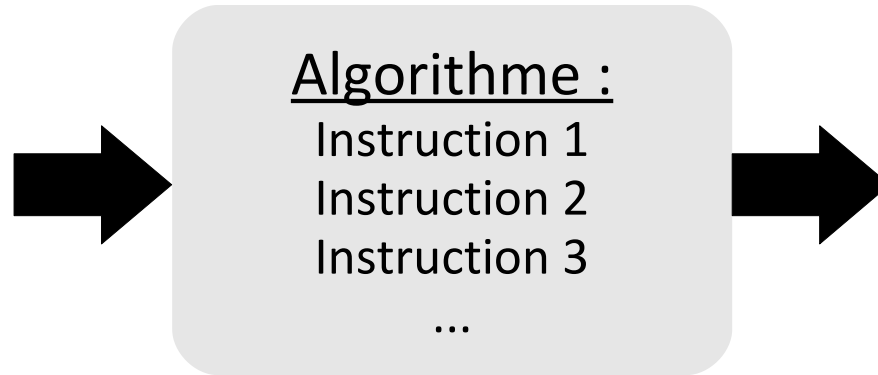
Le **résultat** de sa résolution en **sortie**

# Composants d'un algorithme

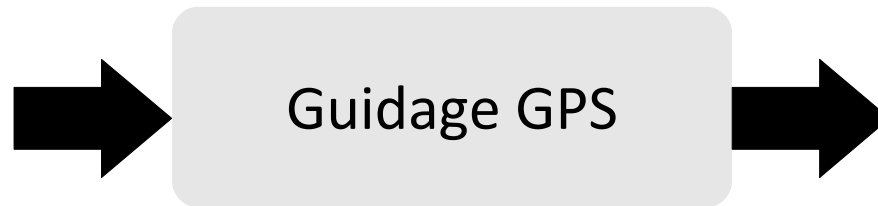
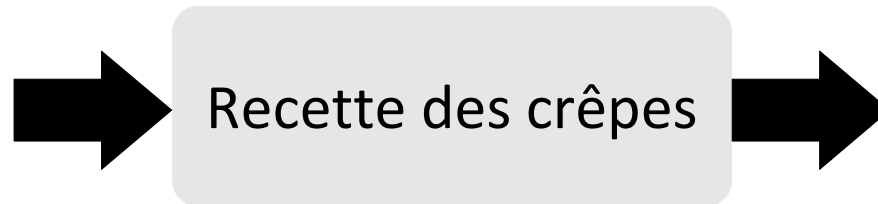


# Composants d'un algorithme

Données du problème  
entrées de l'algorithme

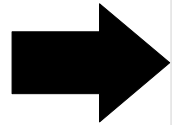


Résultat  
sorties de l'algorithme

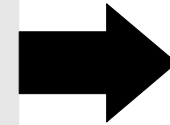


# Composants d'un algorithme

Données du problème  
entrées de l'algorithme

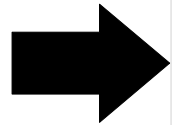


Algorithme :  
Instruction 1  
Instruction 2  
Instruction 3  
...

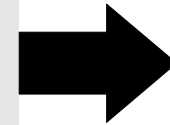


Résultat  
sorties de l'algorithme

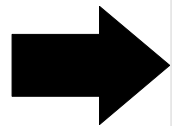
ingrédients  
beurre, oeufs, sachets  
de sucre vanillé, farine,  
lait, sel et quantités



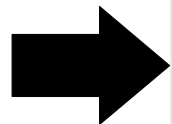
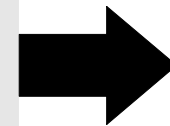
Recette des crêpes



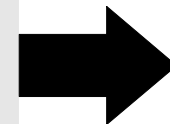
crêpes



Guidage GPS

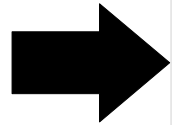


Multiplication

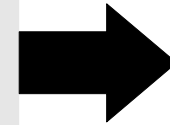


# Composants d'un algorithme

Données du problème  
entrées de l'algorithme



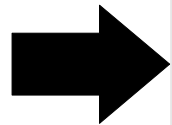
Algorithme :  
Instruction 1  
Instruction 2  
Instruction 3  
...



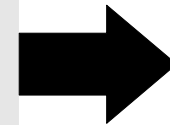
Résultat  
sorties de l'algorithme

ingrédients

beurre, oeufs, sachets de sucre vanillé, farine, lait, sel et quantités



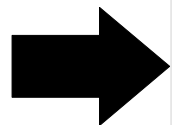
Recette des crêpes



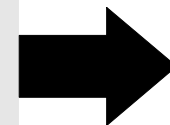
crêpes

informations sur l'environnement

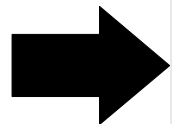
plan des rues, position de la voiture, position de la destination, contraintes (péages ou non)



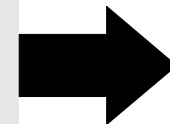
Guidage GPS



itinéraire ou prochaine direction à prendre

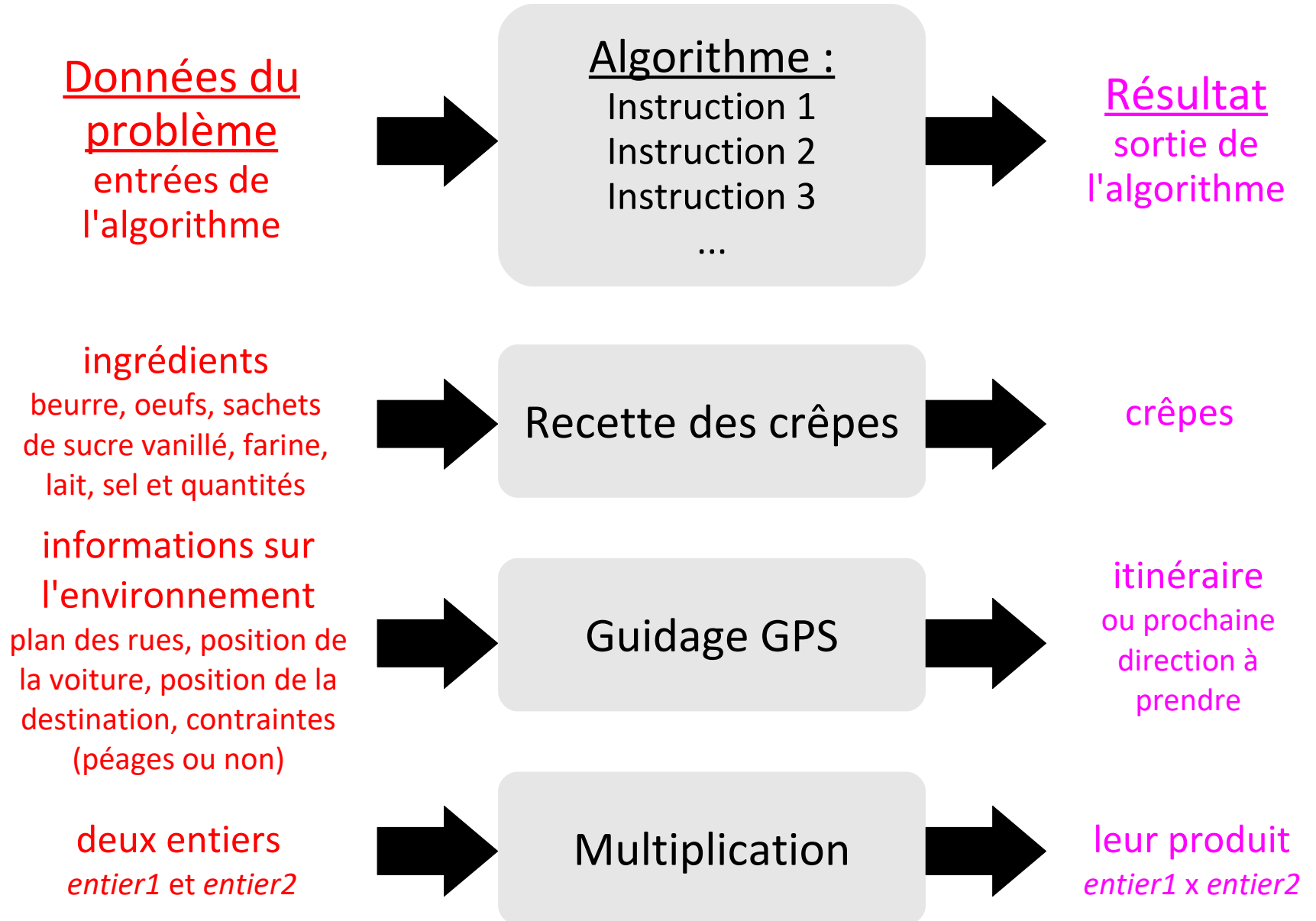


Multiplication





# Composants d'un algorithme



# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - **de façon structurée et compacte**
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

Méthode de résolution d'un problème :

**facile à comprendre**

**facile à transmettre**

# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - **à partir d'opérations de base**
  - indépendamment d'un langage de programmation

Méthode de résolution d'un problème :

**adaptée aux moyens à disposition**

**adaptée aux connaissances des personnes qui l'utilisent**

# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

Méthode de résolution d'un problème :

**adaptée pour des problèmes qui se traitent sans ordinateur**  
**compréhensible sans apprendre un langage de programmation**

# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

## *La “minute culturelle”*

### Algorithmes **sans ordinateur** :

- Euclide (vers -300) : calcul du PGCD de 2 nombres



# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

## *La “minute culturelle”*

### Algorithmes **sans ordinateur** :

- Euclide (vers -300) : calcul du PGCD de 2 nombres
- Al-Khwarizmi (825) : résolution d'équations



# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

## *La “minute culturelle”*

### Algorithmes **sans ordinateur** :

- Euclide (vers -300) : calcul du PGCD de 2 nombres
- Al-Khuwārizmī (825) : résolution d'équations
- Ada Lovelace (1842) : calcul des nombres de Bernoulli sur la *machine analytique* de Charles Babbage



# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

Trois **langages** abordés dans ce cours :

**organigramme**

**pseudo-code**

**Javascript**



# À quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

Trois **langages** abordés dans ce cours :

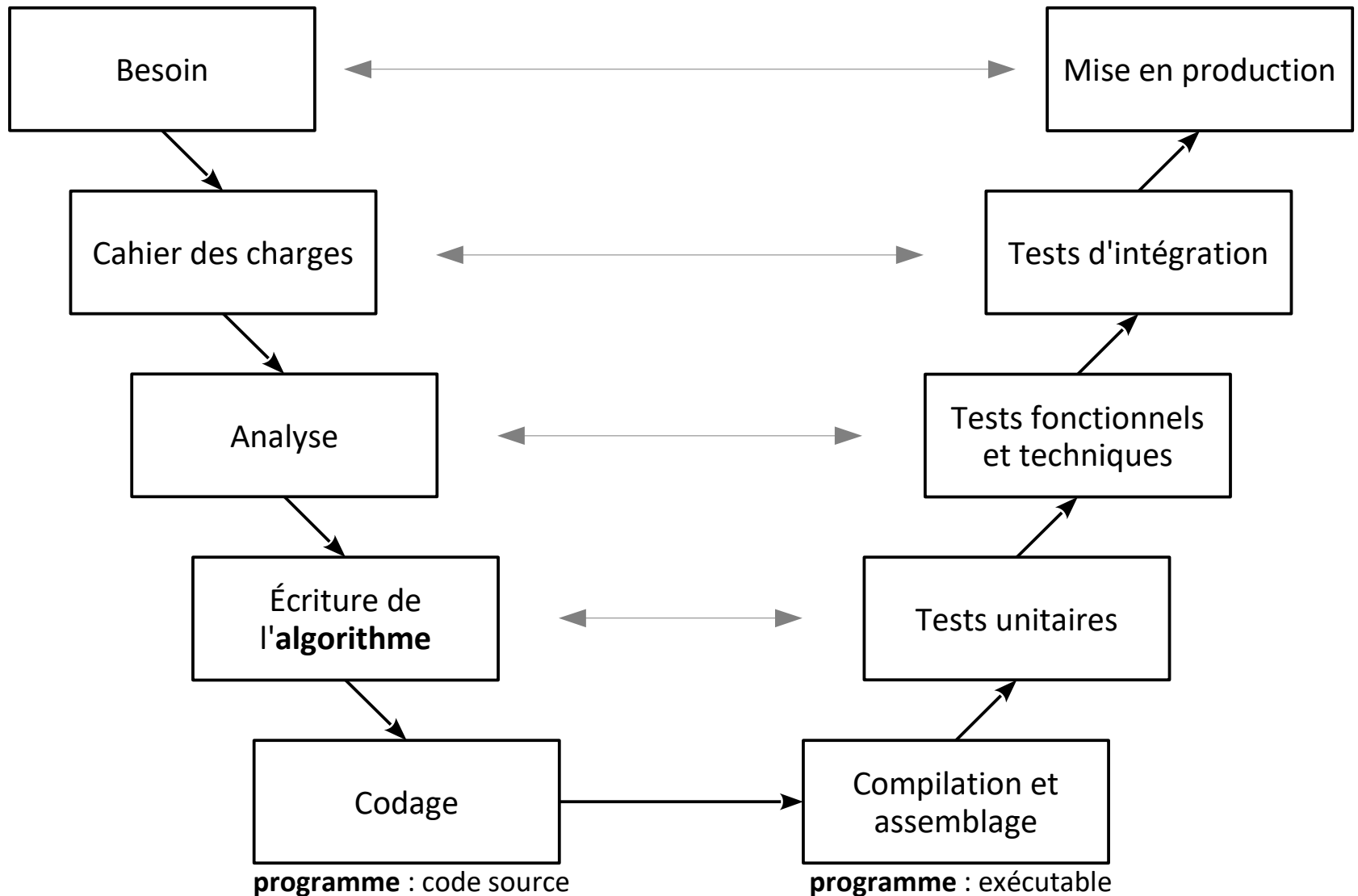
**organigramme**

**pseudo-code**

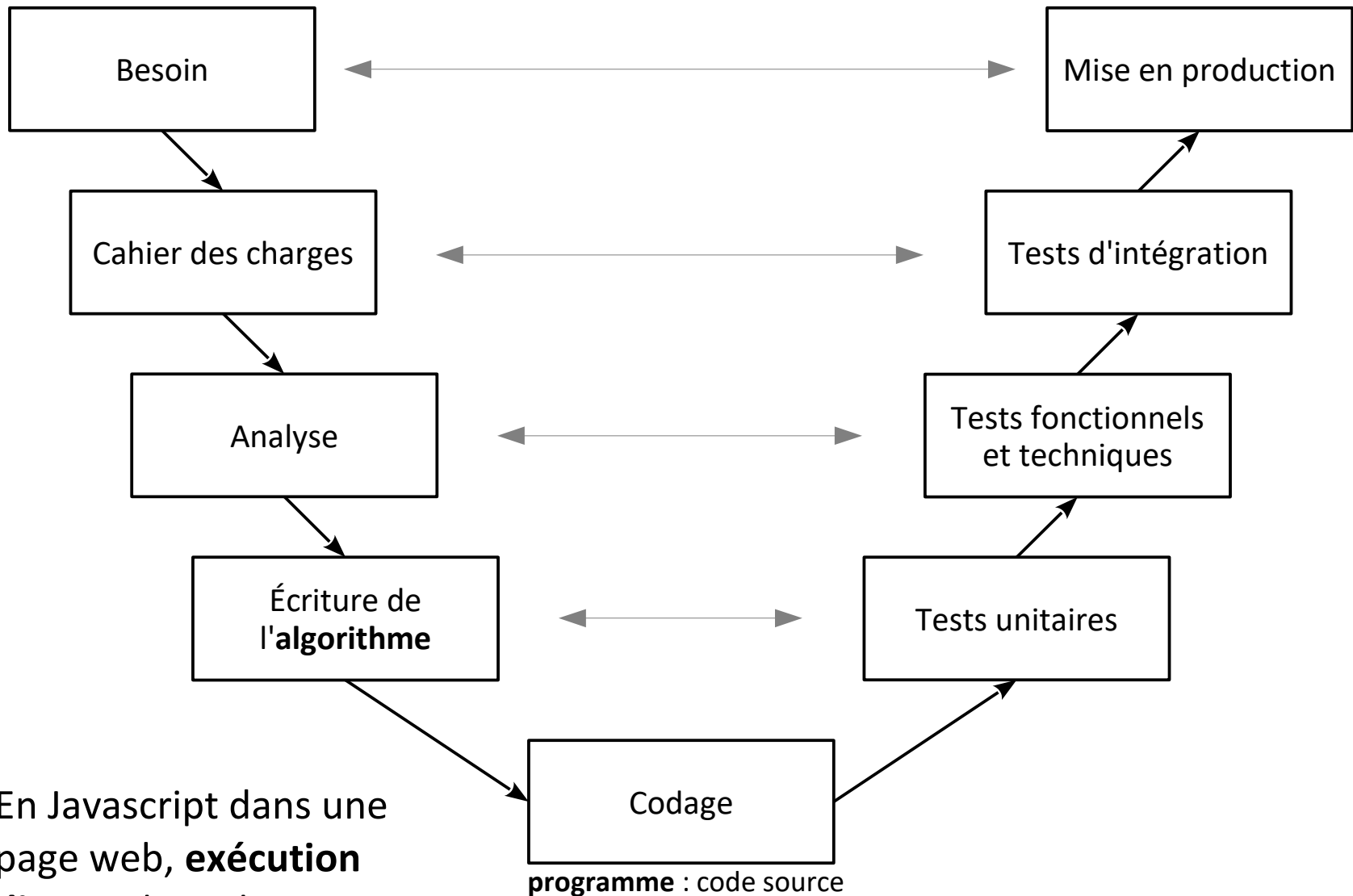
**Javascript**

avantage aux  
littéraires !

# De l'algorithme au programme, le "cycle en V"

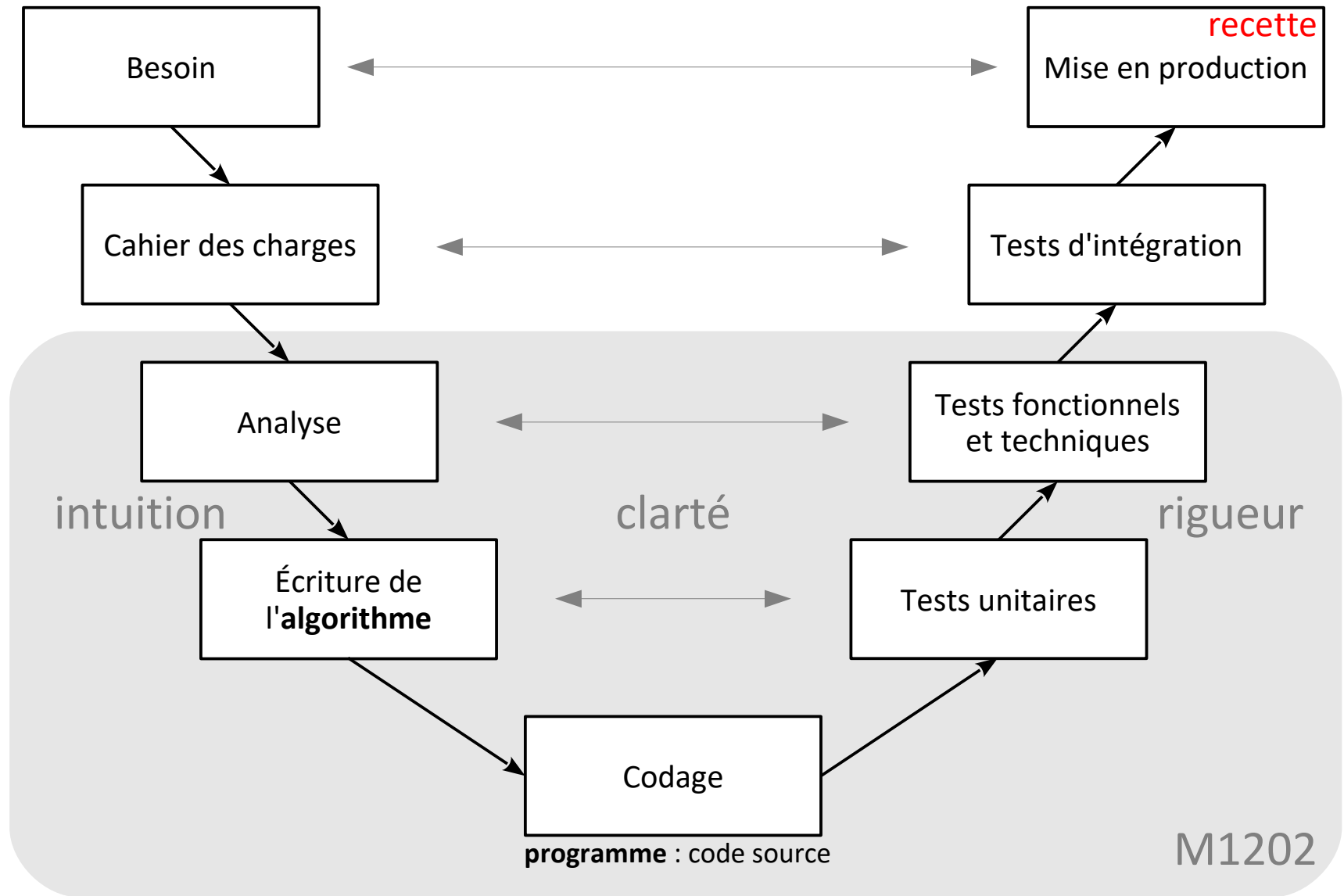


# De l'algorithme au programme, le "cycle en V"

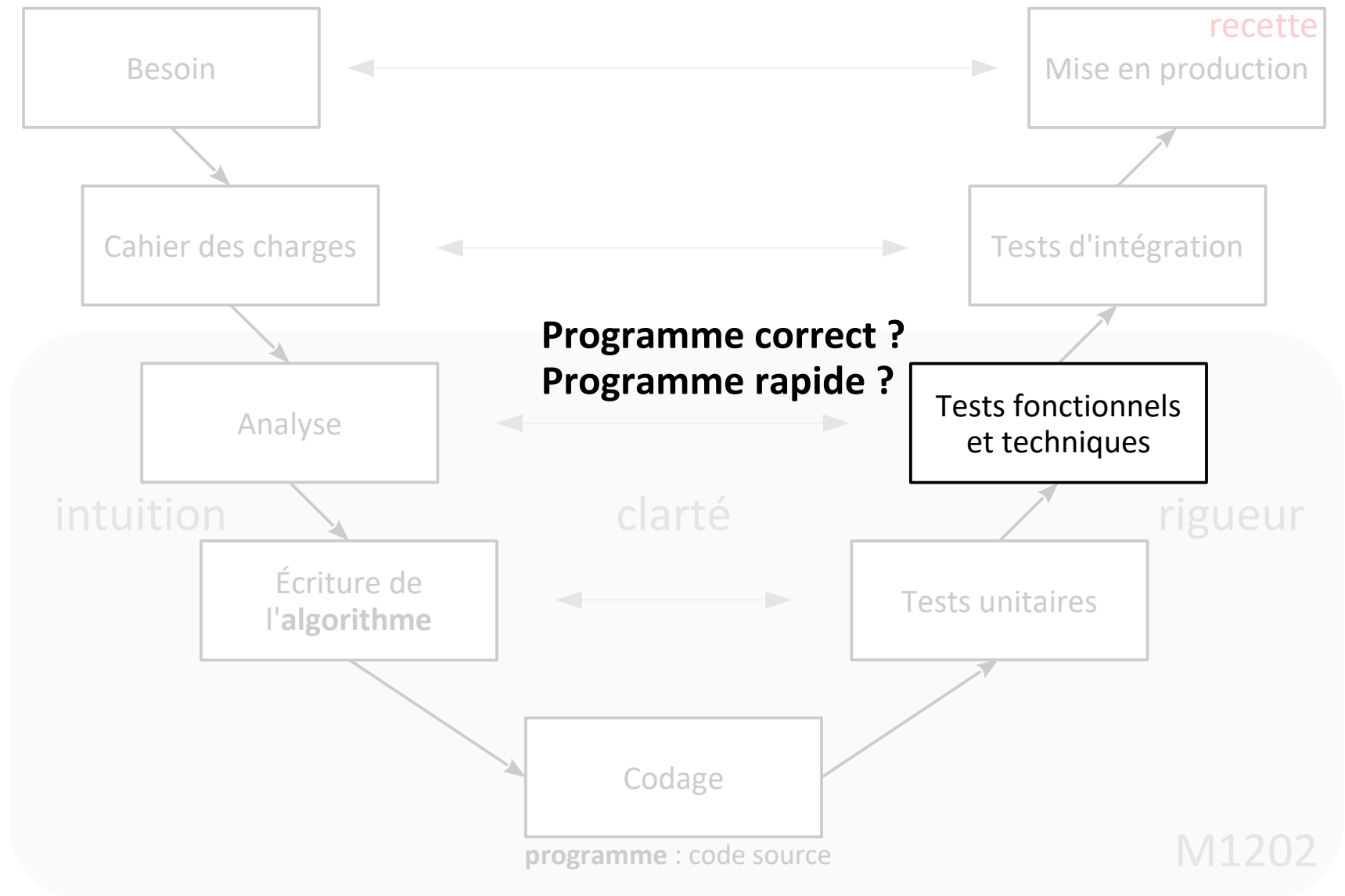


En Javascript dans une page web, **exécution directe** du code source par le navigateur

# De l'algorithme au programme, le "cycle en V"



# De l'algorithme au programme, le "cycle en V"



# Enjeux de l'algorithmique

---

**Algorithme correct ?**

**Algorithme rapide ?**

# Enjeux de l'algorithmique

## Algorithme correct ?

- donne le résultat attendu ?
- quel que soit le type d'entrées ?

## Algorithme rapide ?

- se termine ?
- en combien de temps ?



crêpes, GPS,  
multiplication

# Enjeux de l'algorithmique

## Algorithme correct ?

- donne le résultat attendu ? → **preuve de correction**
- quel que soit le type d'entrées ? → **débuggage, tests unitaires**

## Algorithme rapide ?

- se termine ? → **preuve de terminaison**
- en combien de temps ? → **complexité**



crêpes, GPS,  
multiplication



# Enjeux de l'algorithmique - correction

**Correction** : L'algorithme donne-t-il le résultat attendu ?

**Preuve de correction** :

- « invariant » : propriété vraie tout au long de l'algorithme
    - vraie à la première étape
    - si vraie à une étape, vraie à l'étape suivante
- ⇒ vrai à la fin

**En pratique**, pour débiter :

- vérifier sur les “cas de base”
- vérifier sur des exemples aléatoires

# Enjeux de l'algorithmique - terminaison

L'algorithme se termine-t-il en un temps fini ?

## *Algorithme des crêpes :*

**Mettre** 4 oeufs **dans** le saladier

**Mettre** 1 sachet de sucre vanillé **dans** le saladier

**Mettre** 250 g de farine **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 0,5 litre de lait **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 50 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 1 **minute**

**Mettre** le contenu de la poêle **dans** le saladier

**Mélanger** le contenu du saladier

**Attendre pendant** 60 **minutes**

**Mettre** 5 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 0.5 **minute**

**Tant que** le saladier n'est pas vide :

**Mettre** 5 cL du contenu du saladier **dans**  
le verre mesureur

**Mettre** le contenu du verre mesureur  
**dans** la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Retourner** le contenu de la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Mettre** le contenu de la poêle **dans**  
l'assiette

# Enjeux de l'algorithmique - terminaison

L'algorithme se termine-t-il en un temps fini ?

## *Algorithme des crêpes :*

**Mettre** 4 oeufs **dans** le saladier

**Mettre** 1 sachet de sucre vanillé **dans** le saladier

**Mettre** 250 g de farine **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 0,5 litre de lait **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 50 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 1 **minute**

**Mettre** le contenu de la poêle **dans** le saladier

**Mélanger** le contenu du saladier

**Attendre** **pendant** 60 **minutes**

**Mettre** 5 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 0.5 **minute**

**Tant que** le saladier n'est pas vide :

**Mettre** 5 cL du contenu du saladier **dans**  
le verre mesureur

**Mettre** le contenu du verre mesureur  
**dans** la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Retourner** le contenu de la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Mettre** le contenu de la poêle **dans**  
l'assiette

→ Le saladier sera forcément vide à un moment donné !

→ preuve mathématique...

# Enjeux de l'algorithmique - terminaison

*La "minute votes SMS"*

**Problème** : aller en voiture de Châtelet à la Tour Montparnasse

# Enjeux de l'algorithmique - terminaison

*La “minute votes SMS”*

**Problème** : aller en voiture de Châtelet à la Tour Montparnasse

**Algorithme “du repère visuel”** :

A tout instant on sait où se trouve la Tour Montparnasse

→ prendre la rue qui s'en rapproche le plus

# Enjeux de l'algorithmique - terminaison

*La “minute votes SMS”*

**Problème** : aller en voiture de Châtelet à la Tour Montparnasse

**Algorithme “du repère visuel”** :

A tout instant on sait où se trouve la Tour Montparnasse

→ prendre la rue qui s'en rapproche le plus

algorithme de la famille des  
**algorithmes gloutons**



toujours choisir le **profit maximum** !

# Enjeux de l'algorithmique - terminaison

*La “minute votes SMS”*

**Problème** : aller en voiture de Châtelet à la Tour Montparnasse

**Algorithme “du repère visuel”** :

A tout instant on sait où se trouve la Tour Montparnasse

→ prendre la rue qui s'en rapproche le plus

**Question** : l'algorithme “du repère visuel” termine ?

# Enjeux de l'algorithmique - terminaison

*La "minute votes SMS"*

**Problème** : aller en voiture de Châtelet à la Tour Montparnasse

**Algorithme "du repère visuel"** :

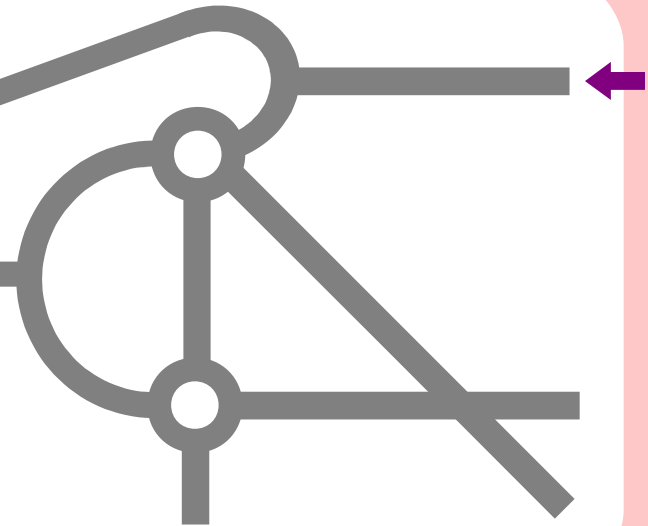
A tout instant on sait où se trouve la Tour Montparnasse

→ prendre la rue qui s'en rapproche le plus

**Question** : l'algorithme "du repère visuel" termine ?



distance (en millimètres, au mm près)  
entre la position actuelle et la Tour  
Montparnasse, entière, positive,  
strictement décroissante ?







# Enjeux de l'algorithmique - complexité

**Complexité** : Combien de temps l'algorithme prend-il pour se terminer ?

**Théorie de la complexité** :

- nombre d'opérations en fonction de la taille du problème, dans le pire cas
- prouver qu'on ne peut pas utiliser moins d'opérations pour résoudre le problème, dans le pire cas

**En pratique**, pour débiter :

- vérifier sur des exemples aléatoires
- connaître les cas difficiles

# Enjeux de l'algorithmique - complexité

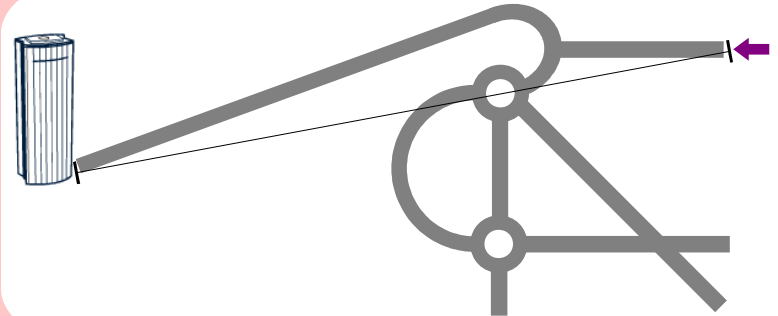
**Complexité** : Combien de temps l'algorithme prend-il pour se terminer ?

## **Théorie de la complexité :**

- nombre d'opérations en fonction de la taille du problème, dans le pire cas
- prouver qu'on ne peut pas utiliser moins d'opérations pour résoudre le problème, dans le pire cas

## **En pratique, pour débiter :**

- vérifier sur des exemples aléatoires
- connaître les cas difficiles



*impossible de faire mieux que la ligne droite !*

# Quels types d'instructions ?

Divers types d'instructions :

- déclaration d'un algorithme

- appel d'un algorithme

- déclaration d'une **variable**

- **affectation** d'une **variable**

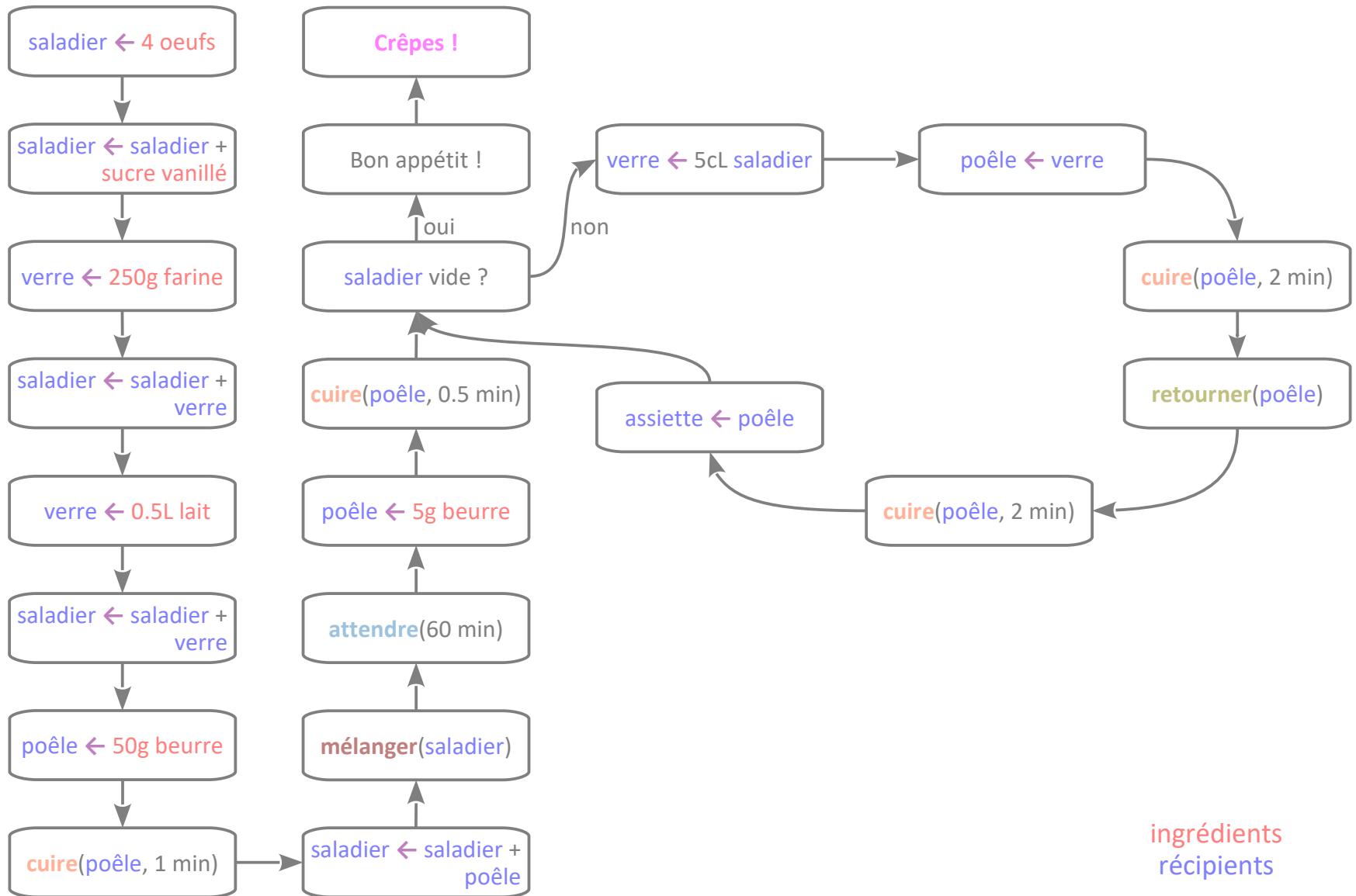
pour stocker des valeurs, des résultats intermédiaires

- **entrées** / **sorties**

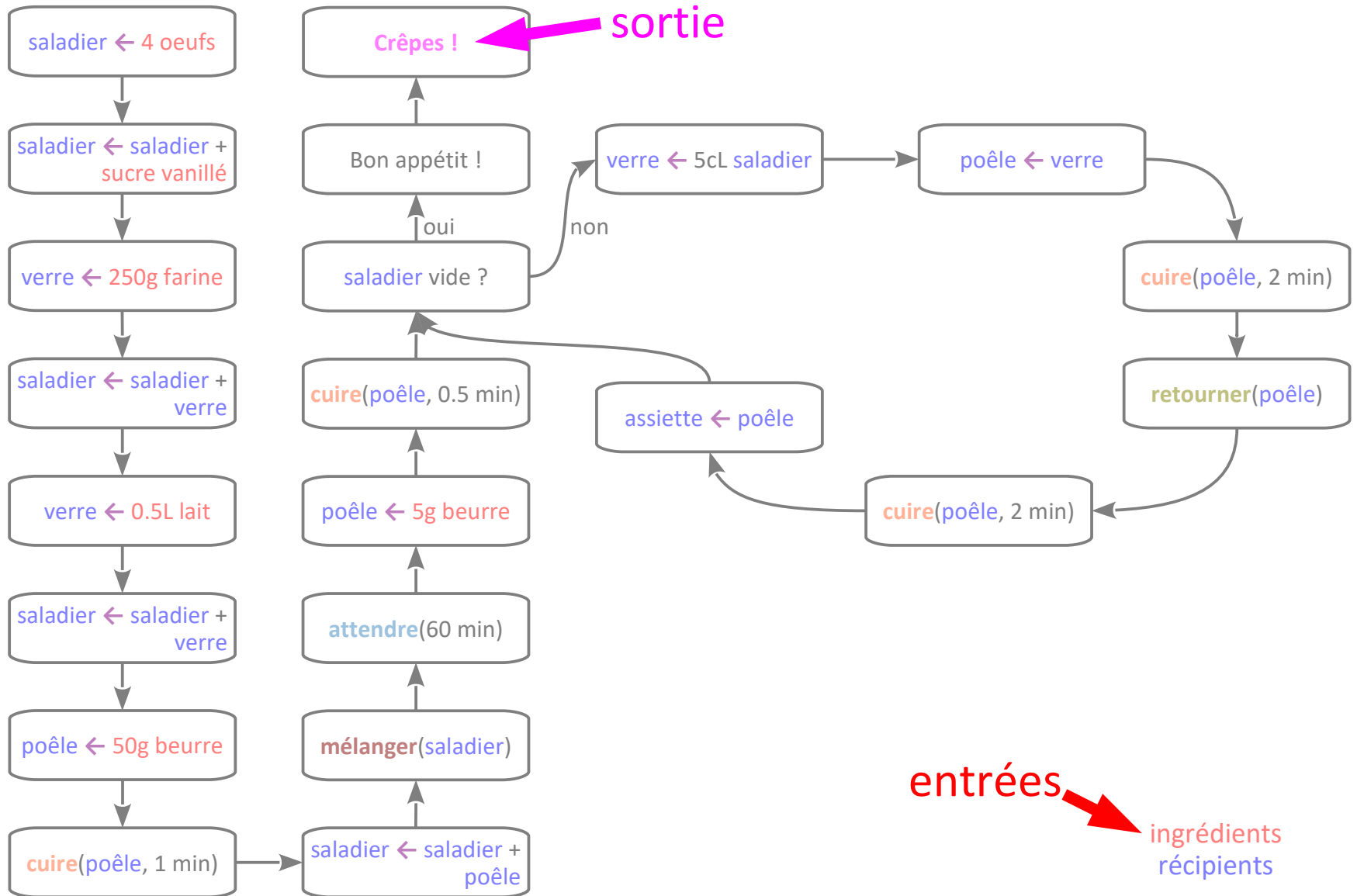
- **boucle**

- test

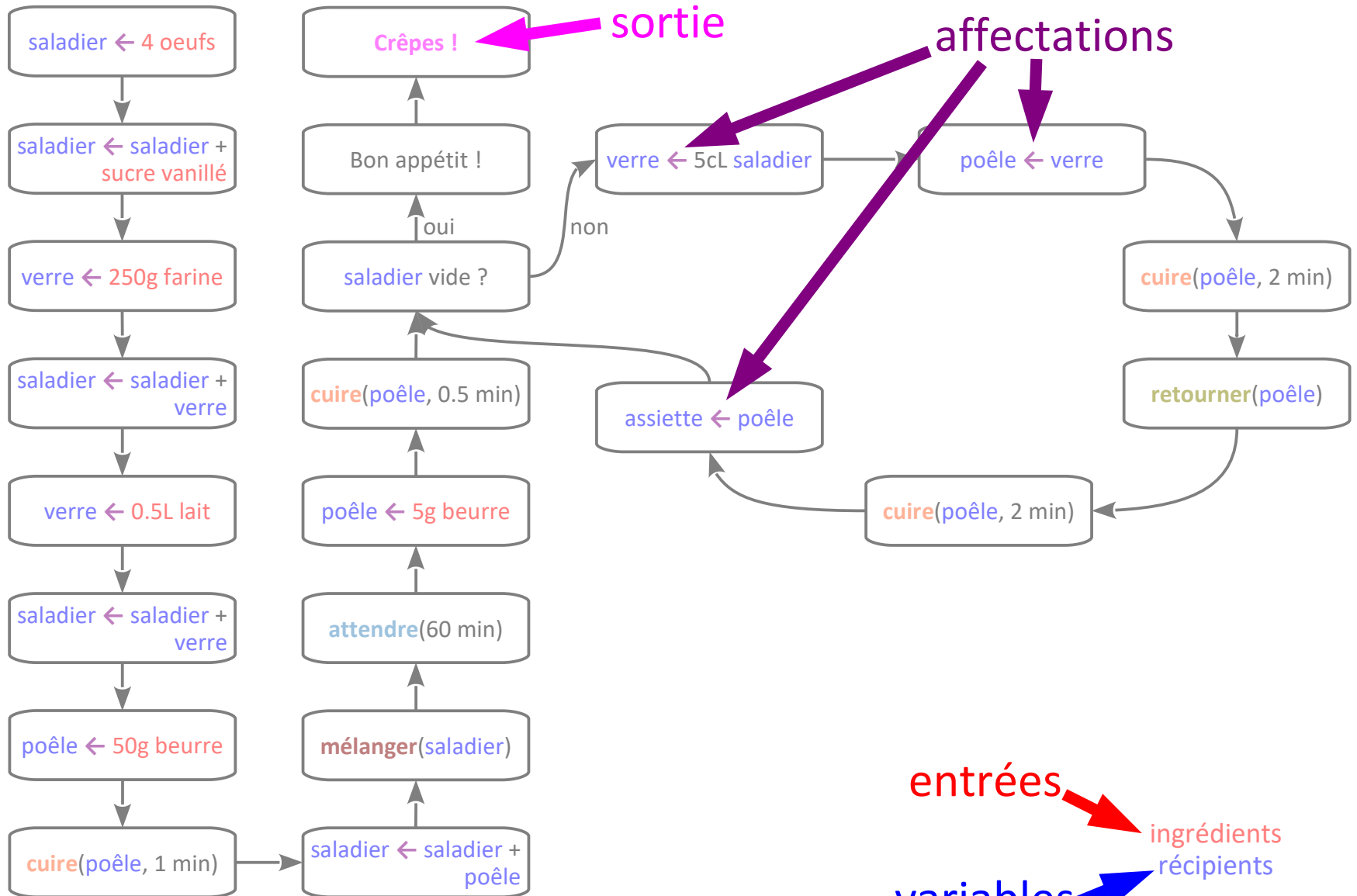
# Organigramme de la recette des crêpes



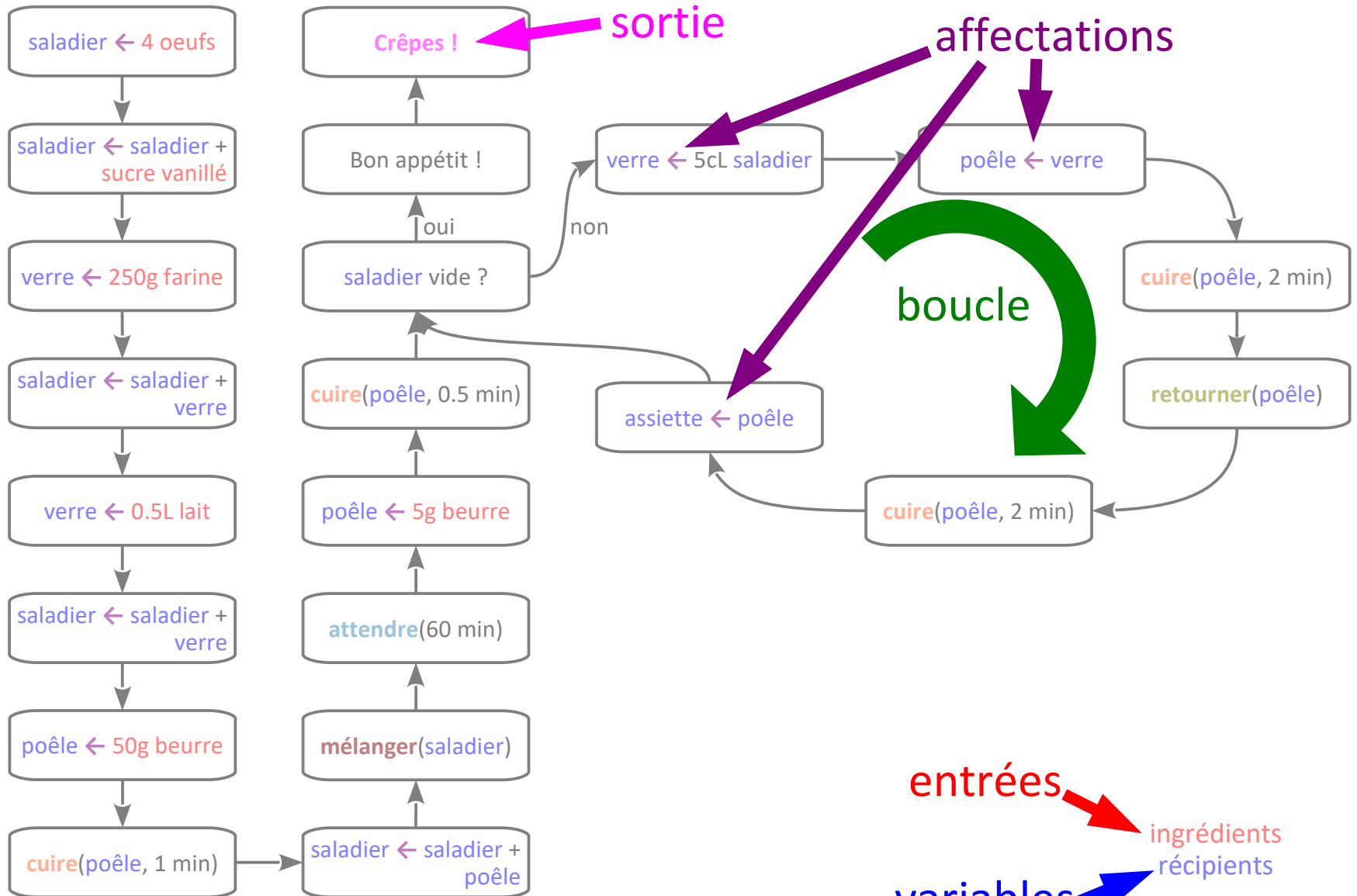
# Organigramme de la recette des crêpes



# Organigramme de la recette des crêpes

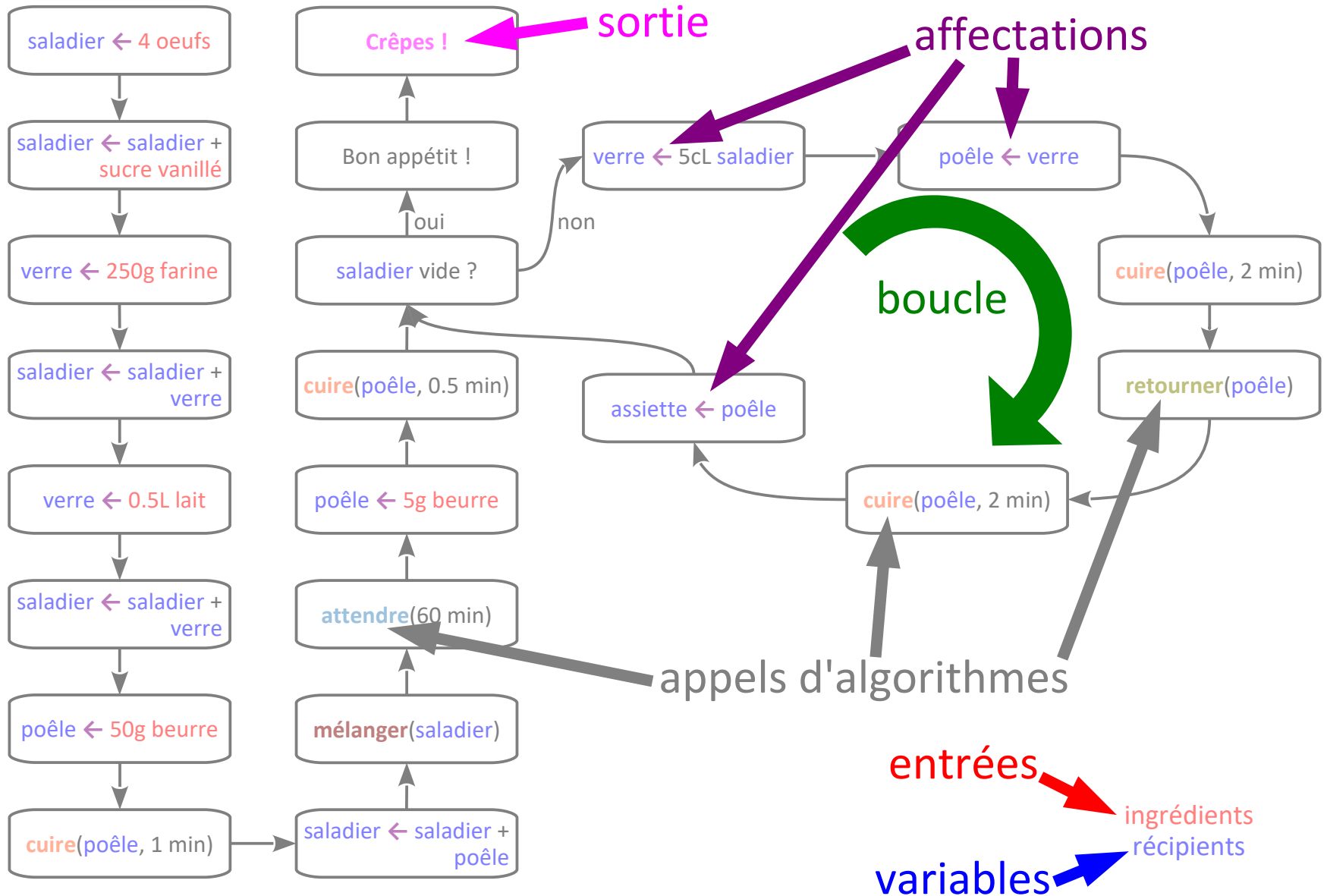


# Organigramme de la recette des crêpes





# Organigramme de la recette des crêpes



# Premier programme Javascript

entrées ?

sorties ?

```
async function fonctionPrincipale(){
  //Identification de l'internaute
  var nom = await reponseALAQuestion("Comment vous
    appelez-vous ?");
  afficheConsole("Nom entré : "+nom)

  affiche(" ");
  affiche("Bonjour "+nom+" !");

  //Choix du nombre aléatoire
  var nombreATrouver = nombreAleatoire(1,10);
  affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
  affiche("Essayez de le deviner.");

  //Premier essai de l'utilisateur
  var reponse = await reponseALAQuestion("Premier essai :");
  var nombreUtilisateur = convertitEnEntier(reponse);

  if (nombreUtilisateur == nombreATrouver){
    affiche("Bravo "+nom+", vous avez trouvé
      du premier coup !");
  }
}
```

# Premier programme Javascript

sortie : "gambette" si l'utilisatrice ou utilisateur a entré ce nom

```
async function fonctionPrincipale(){
  //Identification de l'internaute
  var nom = await reponseALAQuestion("Comment vous
    appelez-vous ?");
  afficheConsole("Nom entré : "+nom)

  affiche(" ");
  affiche("Bonjour "+nom+" !");

  //Choix du nombre aléatoire
  var nombreATrouver = nombreAleatoire(1,10);
  affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
  affiche("Essayez de le deviner.");

  //Premier essai de l'utilisateur
  var reponse = await reponseALAQuestion("Premier essai :");
  var nombreutilisateur = convertitEnEntier(reponse);

  if (nombreutilisateur == nombreATrouver){
    affiche("Bravo "+nom+", vous avez trouvé
      du premier coup !");
  }
}
```

entrées

sortie : 3 si l'ordinateur a choisi ce numéro au hasard

# Premier programme Javascript

```
async function fonctionPrincipale(){
  //Identification de l'internaute
  var nom = await reponseALAQuestion("Comment vous
    appelez-vous ?");
  afficheConsole("Nom entré : "+nom)

  affiche(" ");
  affiche("Bonjour "+nom+" !");

  //Choix du nombre aléatoire
  var nombreATrouver = nombreAleatoire(1,10);
  affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
  affiche("Essayez de le deviner.");

  //Premier essai de l'utilisateur
  var reponse = await reponseALAQuestion("Premier essai :");
  var nombreUtilisateur = convertitEnEntier(reponse);

  if (nombreUtilisateur == nombreATrouver){
    affiche("Bravo "+nom+", vous avez trouvé
      du premier coup !");
  }
}
```

variables

# Premier programme Javascript

```
async function fonctionPrincipale(){
  //Identification de l'internaute
  var nom = await reponseALAQuestion("Comment vous
    appelez-vous ?");
  afficheConsole("Nom entré : "+nom)

  affiche(" ");
  affiche("Bonjour "+nom+" !");

  //Choix du nombre aléatoire
  var nombreATrouver = nombreAleatoire(1,10);
  affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
  affiche("Essayez de le deviner.");

  //Premier essai de l'utilisateur
  var reponse = await reponseALAQuestion("Premier essai :");
  var nombreUtilisateur = convertitEnEntier(reponse);

  if (nombreUtilisateur == nombreATrouver){
    affiche("Bravo "+nom+", vous avez trouvé
      du premier coup !");
  }
}
```

variables affectations

# Premier programme Javascript

```
async function fonctionPrincipale(){
  //Identification de l'internaute
  var nom = await reponseALaQuestion("Comment vous
    appelez-vous ?");
  afficheConsole("Nom entré : "+nom)

  affiche(" ");
  affiche("Bonjour "+nom+" !");

  //Choix du nombre aléatoire
  var nombreATrouver = nombreAleatoire(1,10);
  affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
  affiche("Essayez de le deviner.");

  //Premier essai de l'utilisateur
  var reponse = await reponseALaQuestion("Premier essai :");
  var nombreUtilisateur = convertitEnEntier(reponse);

  if (nombreUtilisateur == nombreATrouver){
    affiche("Bravo "+nom+", vous avez trouvé
      du premier coup !");
  }
}
```

variables affectations appel d'algorithme

# Premier programme Javascript

```
async function fonctionPrincipale(){
  //Identification de l'internaute
  var nom = await reponseALaQuestion("Comment vous
    appelez-vous ?");
  afficheConsole("Nom entré : "+nom)

  affiche(" ");
  affiche("Bonjour "+nom+" !");

  //Choix du nombre aléatoire
  var nombreATrouver = nombreAleatoire(1,10);
  affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
  affiche("Essayez de le deviner.");

  //Premier essai de l'utilisateur
  var reponse = await reponseALaQuestion("Premier essai :");
  var nombreUtilisateur = convertitEnEntier(reponse);

  if (nombreUtilisateur == nombreATrouver){
    affiche("Bravo "+nom+", vous avez trouvé
      du premier coup !");
  }
}
```

variables affectations appel d'algorithme test

# Variables et affectation

Dans un algorithme, une **variable** possède :

- un **nom**,
- une **valeur**,
- un **type** (ensemble des valeurs que peut prendre la variable).

La **valeur** d'une variable :

- est **fixe à un moment donné**,
- peut **changer au cours du temps**.

En revanche, le nom d'une variable ne change pas.

Il est **possible en Javascript** mais **pas recommandé** de la changer de type.



# Variables et affectation

Dans un algorithme, une **variable** possède :

- un **nom**,
- une **valeur**,
- un **type** (ensemble des valeurs que peut prendre la variable).

La **valeur** d'une variable :

- est **fixe à un moment donné**,
- peut **changer au cours du temps**.

L'**affectation** change la valeur d'une variable :

- $a \leftarrow 5$  (pseudo-code) /  $a=5$  (Javascript) :
  - la variable  $a$  prend la valeur 5
  - la valeur précédente est perdue (“écrasée”)
- $a \leftarrow b$  (pseudo-code) /  $a=b$  (Javascript) :
  - la variable  $a$  prend la valeur de la variable  $b$
  - la valeur précédente de  $a$  est perdue (“écrasée”)
  - la valeur de  $b$  n'est pas modifiée
  - $a$  et  $b$  devraient être de même type (ou de type compatible)

# Variables et affectation

Dans un algorithme, une **variable** possède :

- un **nom**,
- une **valeur**,
- un **type** (ensemble des valeurs que peut prendre la variable).

La **valeur** d'une variable :

- est **fixe à un moment donné**,
- peut **changer au cours du temps**.

L'**affectation** change la valeur d'une variable :

- $a \leftarrow 5$  (pseudo-code) /  $a=5$  (Javascript) :
  - la variable  $a$  prend la valeur 5
  - la valeur précédente est perdue (“écrasée”)
- $a \leftarrow b$  (pseudo-code) /  $a=b$  (Javascript) :
  - la variable  $a$  prend la valeur de la variable  $b$
  - la valeur précédente de  $a$  est perdue (“écrasée”)
  - la valeur de  $b$  n'est pas modifiée
  - $a$  et  $b$  devraient être de même type

(ou de type compatible)

La recette de cuisine avec récipiens n'est qu'une métaphore

# Noms des variables

Dans un **algorithme**, choisir pour les variables :

- un nom composé de **lettres** et éventuellement de **chiffres**
- un nom **expressif**, par exemple :
  - *chaine, requête1...* pour une chaîne de caractères
  - *n, a, b, compteur, nbOperations, longueur...* pour un entier
  - *x, y, température* pour un réel
  - *estEntier, testEntier, trouvé...* pour un booléen
- un nom **assez court** (il faut l'écrire !)
- éviter les **noms réservés** : *pour, tant que, si...*

Dans un **programme** :

- **éviter** les lettres accentuées et la ponctuation
- préférer l'**anglais** si votre code source est diffusé largement
- être **expressif** et **lisible** :
  - *est\_entier* ou *estEntier* plutôt que *estentier*

Votre code sera relu, par vous ou par d'autres...