

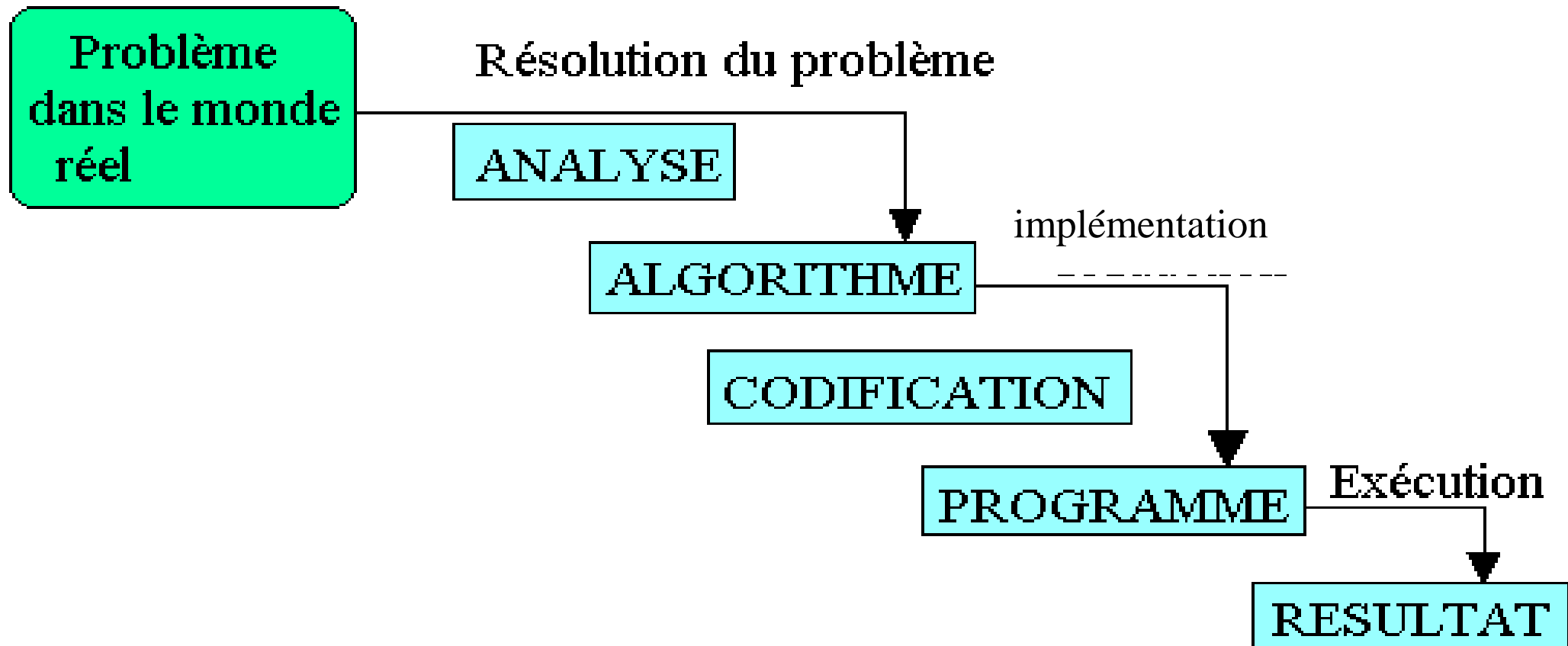
PENSÉE COMPUTATIONNELLE ET PROGRAMMATION

2^{ème} INFO2

Enseignante: Mme Houda Ben Saïd Mangour

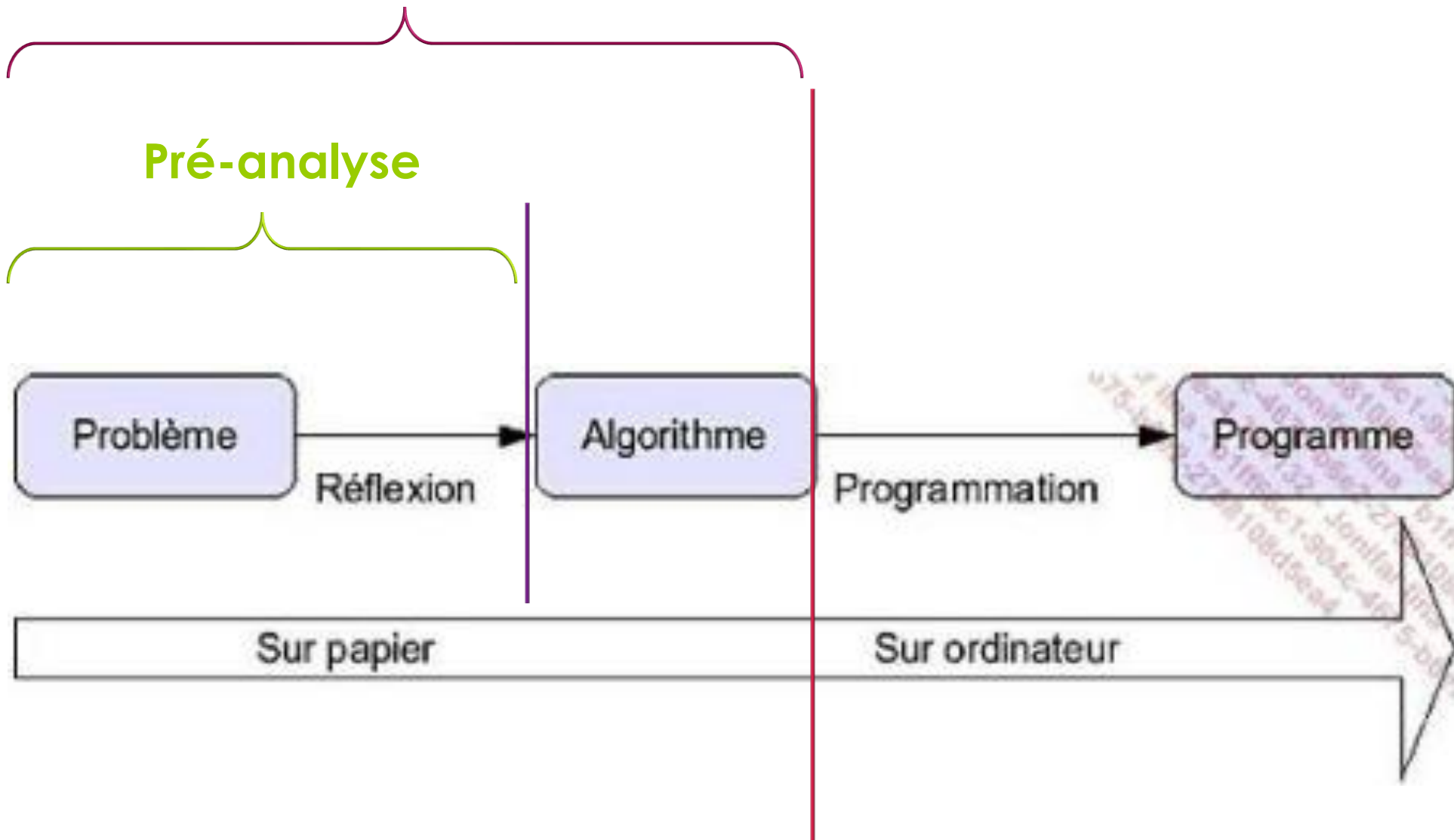
Séance : 29/09/2022

Rappel

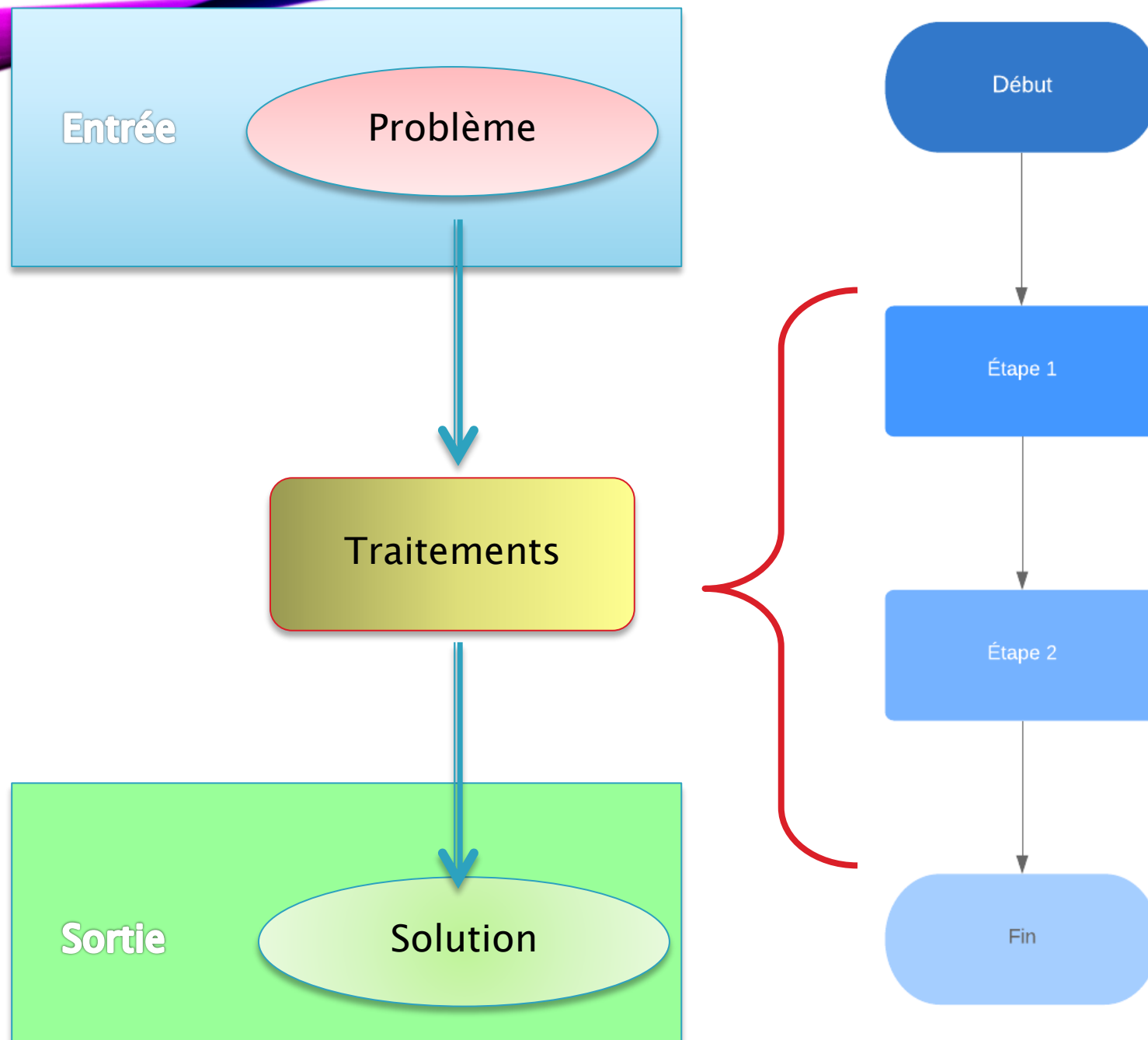


Analyse du problème

Pré-analyse



Un algorithme:



OBJECTIFS DE LA SÉANCE

1. Résoudre le sous-problème

P1

- Représenter la solution sous forme d'un algorithme
- Ecrire l'algorithme sous forme de pseudo-code

2. Tester les solutions du sous-problème

P1

- Implémenter l'algorithme en python

Mini-projet N°1: Calculatrice

Le projet consiste à programmer une calculatrice qui permet de:

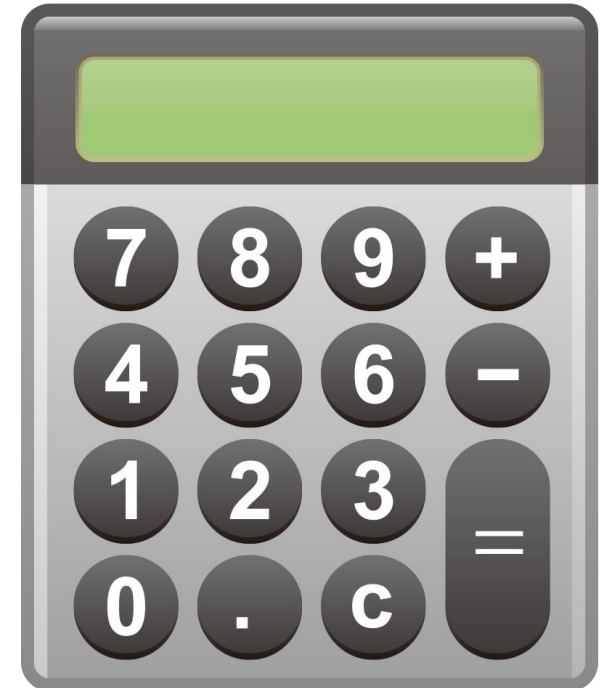
- ▶ Calculer la soustraction et la somme de deux entiers **A** et **B**.
- ▶ Calculer le carré des 10 premiers entiers.

P1

Cette calculatrice doit aussi :

- ✓ Reconnaître si un nombre donné est négatif/positif ou nul.
- ✓ Déterminer si un nombre donné est paire ou impaire.

P2



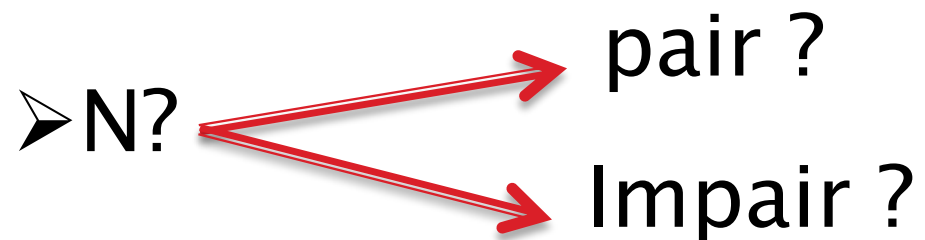
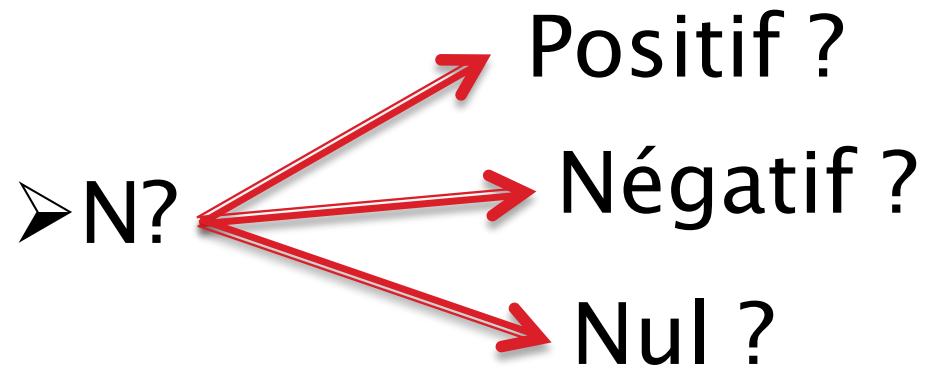


• Déterminer le problème

➤ $A+B=?$

➤ $A-B=?$

➤ $n^2=?$





2

• Analyse du problème

a

Diviser le problème en sous-problèmes

b

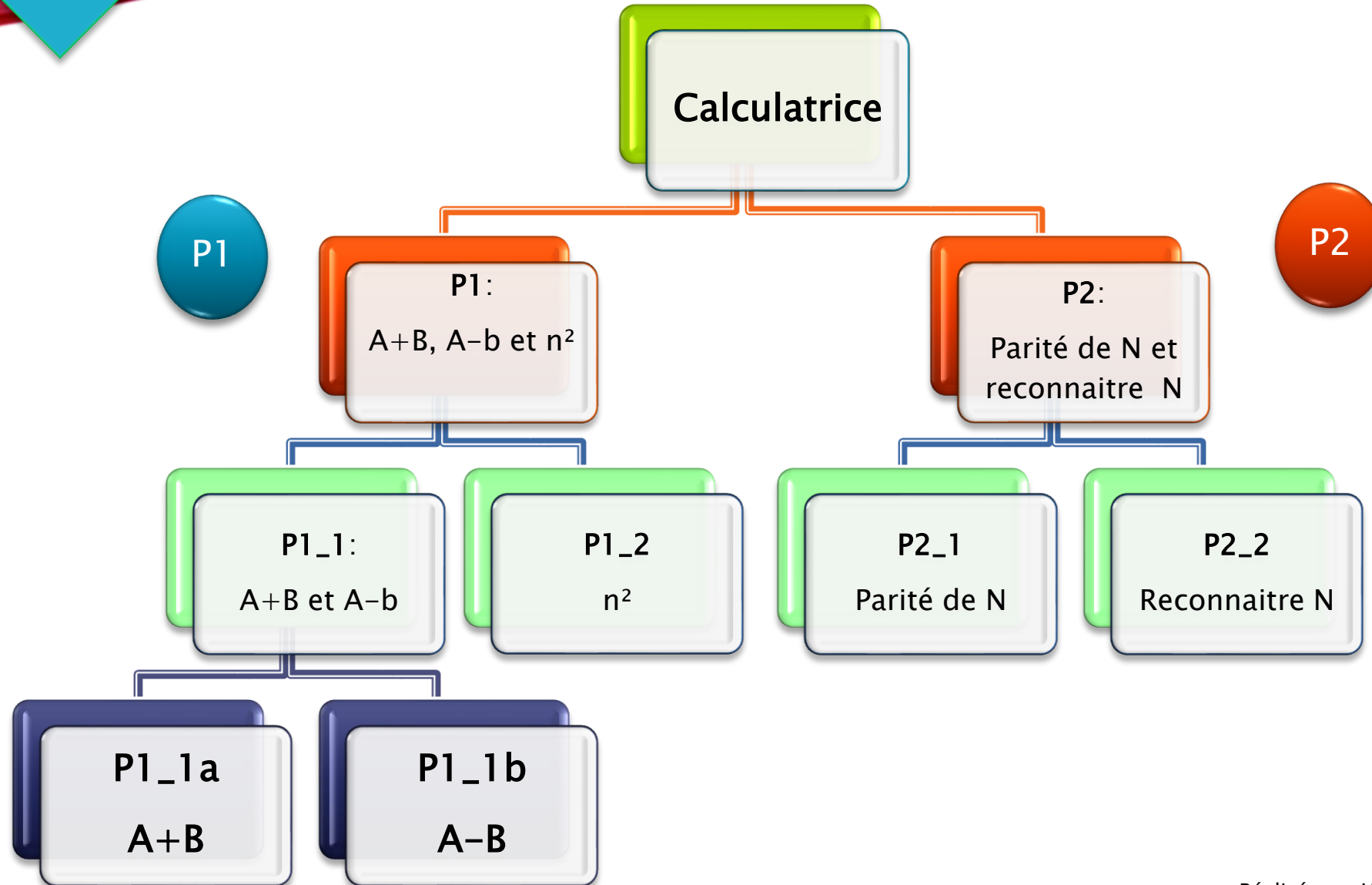
Pré-analyse de chaque sous-problème

c

Algorithme

a

- Diviser le problème en plusieurs sous-problèmes



P1

Calculatrice

P1:
 $A+B$, $A-b$ et n^2

P1_1:
 $A+B$ et $A-b$

P1_2
 n^2

P1_1a
 $A+B$

P1_1b
 $A-B$

Calculatrice

P1:
 $A+B$, $A-b$ et n^2

P1-1

P1_1:
 $A+B$ et $A-b$

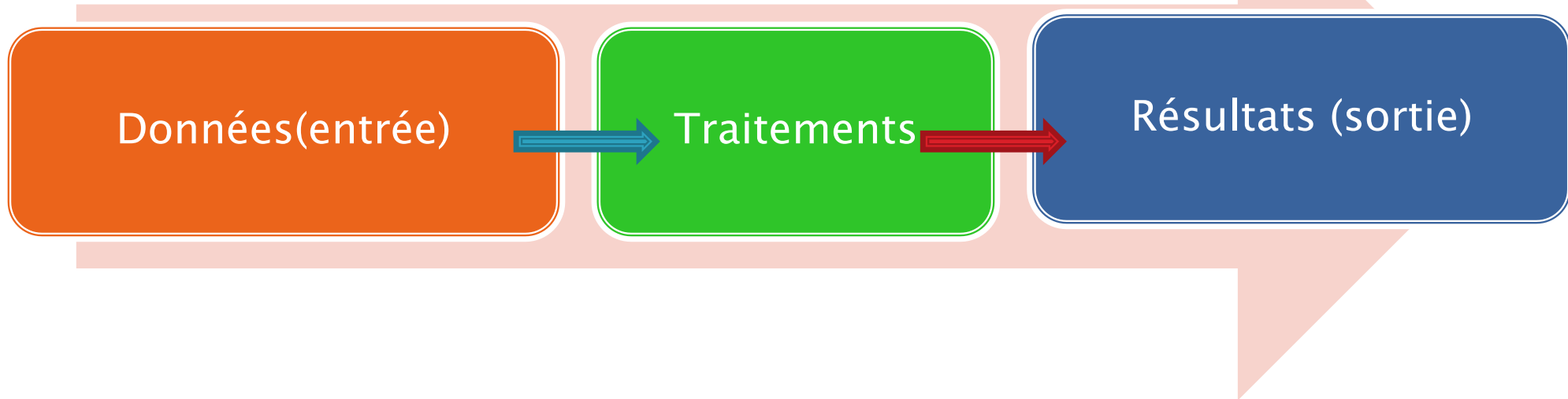
P1-1a

P1_1a
 $A+B$

P1_1b
 $A-B$

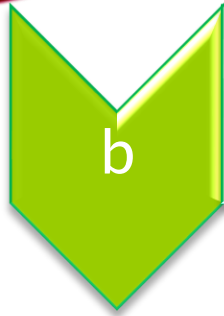
P1-1b

Notez bien



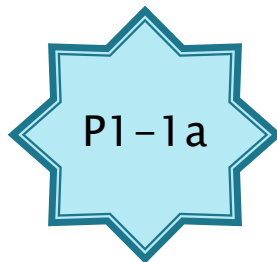
On doit:

1. Déterminer les données en entrée et celles de sortie
2. Attribuer un identifiant à chaque donnée (objet)



• Pré-Analyse du Sous-problème P1-1

Pré-analyse: ➤ $A + B = ?$ ➤ $A - B = ?$



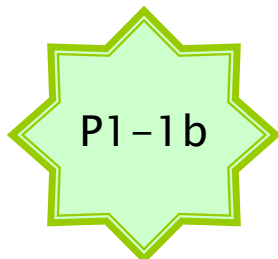
Entrée

A
B

Traitement :
Calculer la
somme de A et B

Addition

Sortie



Entrée

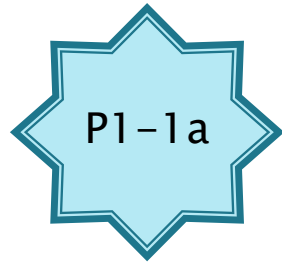
A
B

Traitement :
Calculer la
soustraction de A
et B

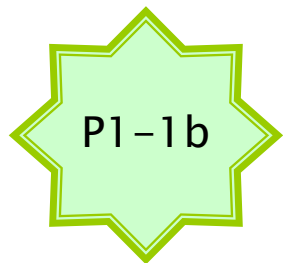
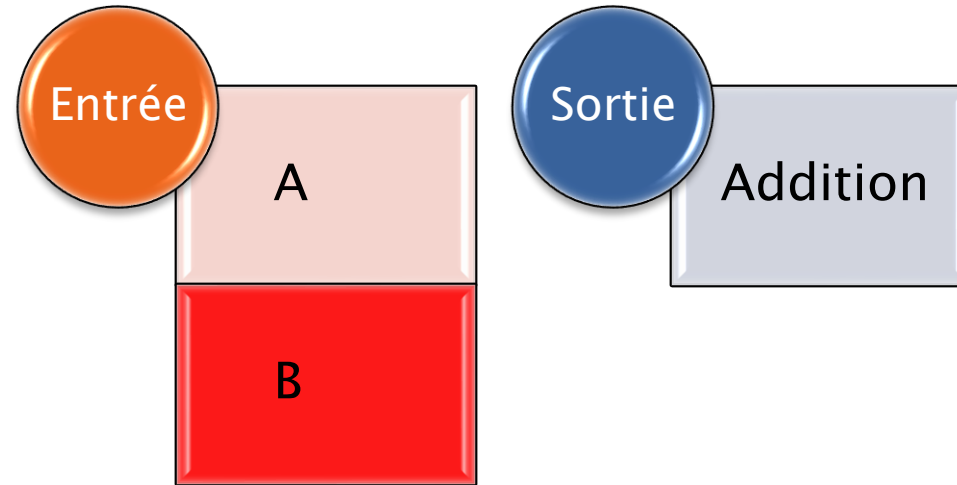
Soustraction

Sortie

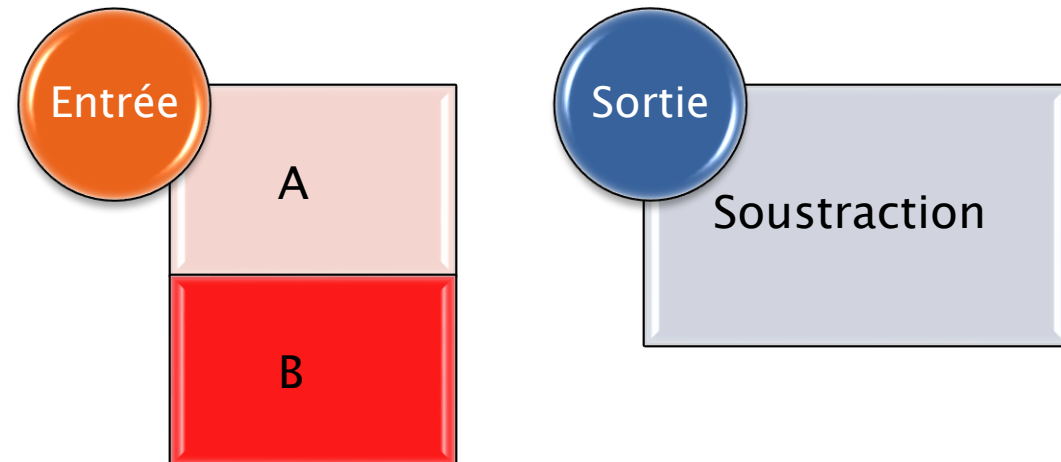
Déterminer les données du Sous-problème



➤ $A + B = ?$



➤ $A - B = ?$

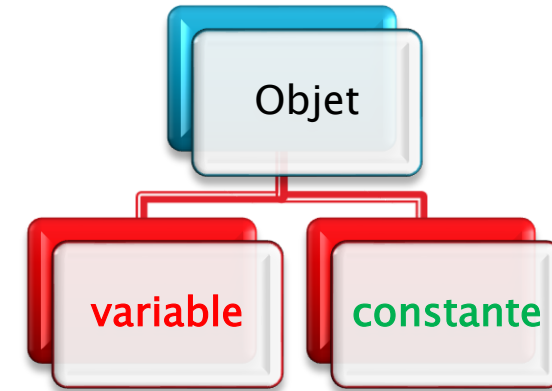


Donner un identificateur aux objets du Sous-problème

	Objet	Identificateur/Nom de Objet	
Objets d'entrée de l'addition	A	A1	
	B	B1	
Objets d'entrée de la soustraction	A	A2	
	B	B2	
Objet de sortie de l'addition	Addition	Add	
Objet de sortie de la soustraction	soustraction	S	



Un objet est une entité utilisée par un programme pour stocker les données.



Contrairement à la **constante**, la valeur de la variable peut changer le long de l'exécution du programme d'où le nom **variable**.

Les **variables** possèdent chacune un nom/identificateur qui permet de les reconnaître et de les utiliser dans un programme/algorithme .



Par analogie les variables peuvent être comparées à des boîtes étiquetées:





Une variable est un **conteneur** pour **une valeur**

Une variable peut contenir différents **types** de valeurs

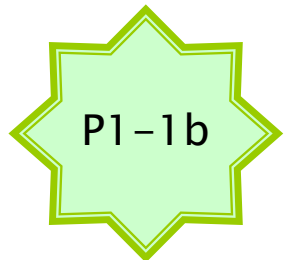
Lors de la **déclaration d'une variable**, le fait d'indiquer le **type** d'une variable va lui faire stocker **un type** de données bien spécifique.

Types:

- ❖ Les entiers
- ❖ Les flottants/réels
- ❖ Les chaînes de caractères
- ❖ Les booléens (VRAI ou FAUX)
- ❖ Les caractères

De retour au Sous-Problème P1-1

Les variables des sous-problèmes



Opérations d'entrée

Les opérations d'entrée sont les instructions qui permettent à l'utilisateur de rentrer ou de saisir des valeurs au clavier pour qu'elles soient utilisées par le programme

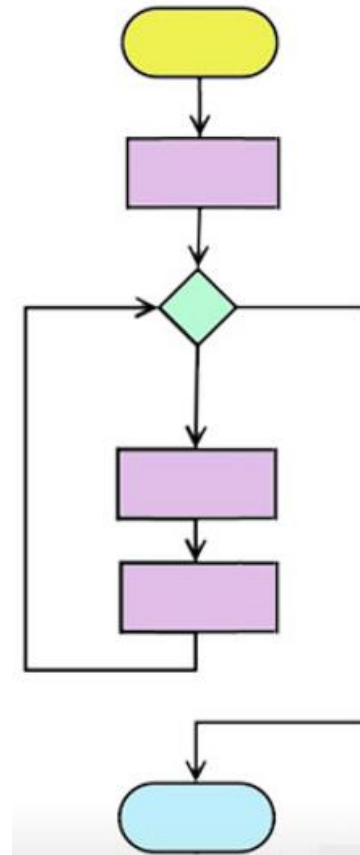
Opérations de sortie

Les opérations de sortie sont les instructions qui permettent au programme de communiquer des valeurs à l'utilisateur en les affichant à l'écran.

Opérations d'affectation

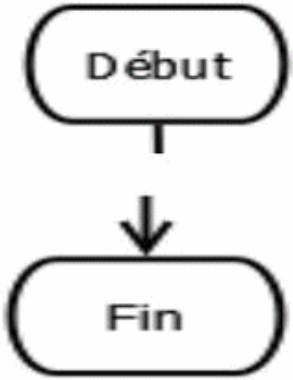
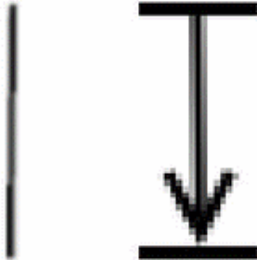

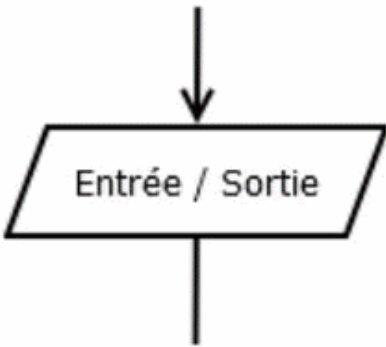
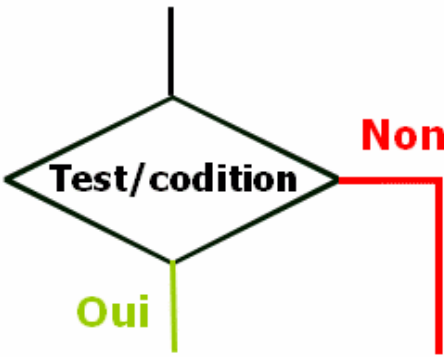
Les opérations d'affectation sont les actions qui permettent d'affecter une valeur à une variable. Elle est représentée par une flèche orientée vers la gauche « ← ».

COMMENT REPRÉSENTER GRAPHIQUEMENT UN ALGORITHME?



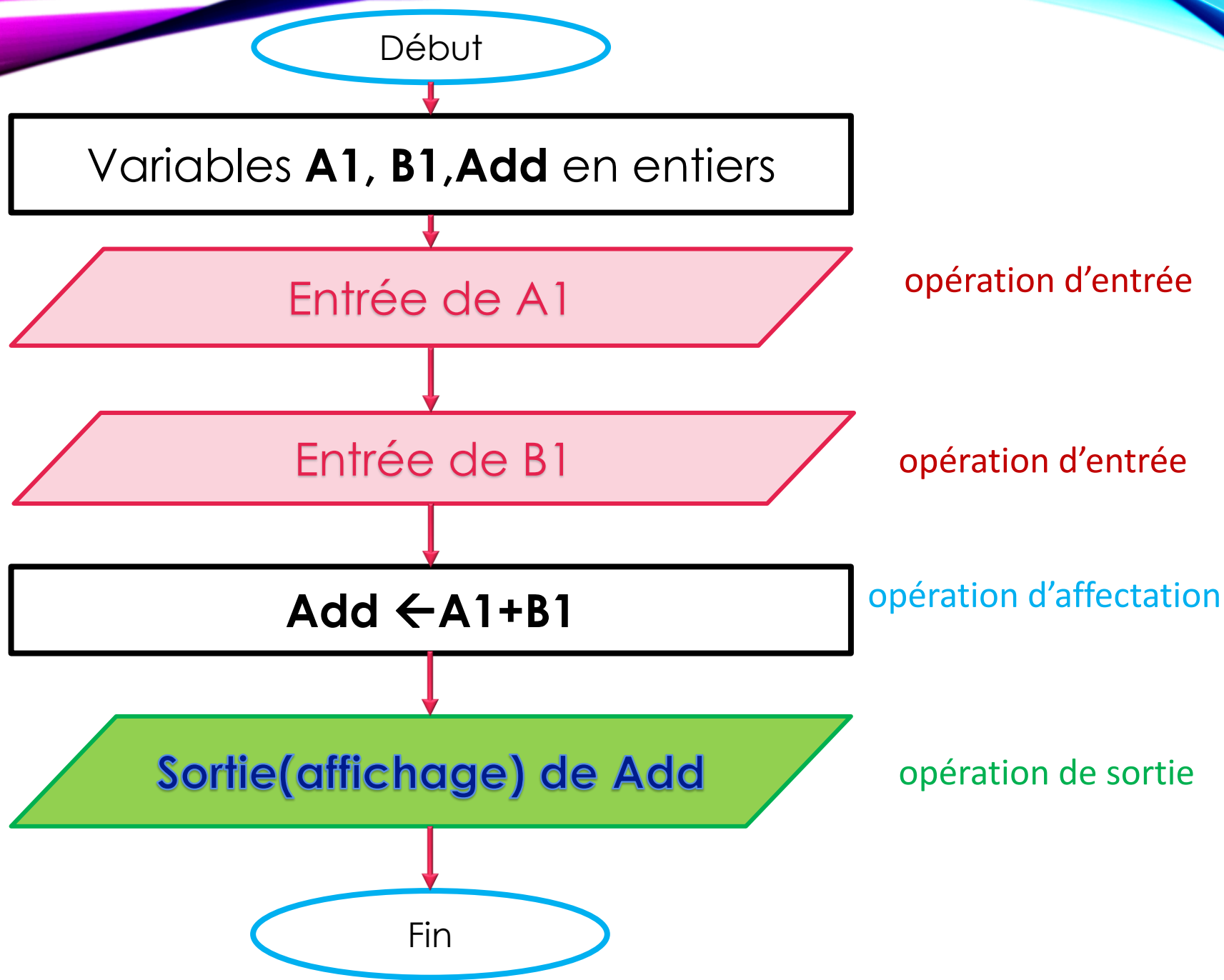
Règles de dessin et symboles

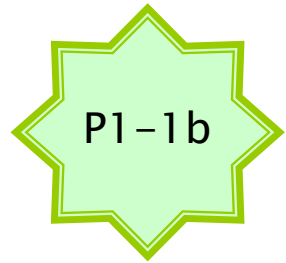
Pour construire un algorithme, on utilise des symboles normalisés

				
<p>Début / fin Il n'y a qu'un seul « Début » mais il peut y avoir plusieurs « Fin »</p>	<p>Liaison, avec ou sans flèche, permettent de guider la lecture de l'algorithme</p>	<p>Instructions : elles permettent de réaliser les traitement : calculs, comparaisons, ...</p>	<p>Lecture ou écriture d'une donnée</p>	<p>Condition / Test logique</p>

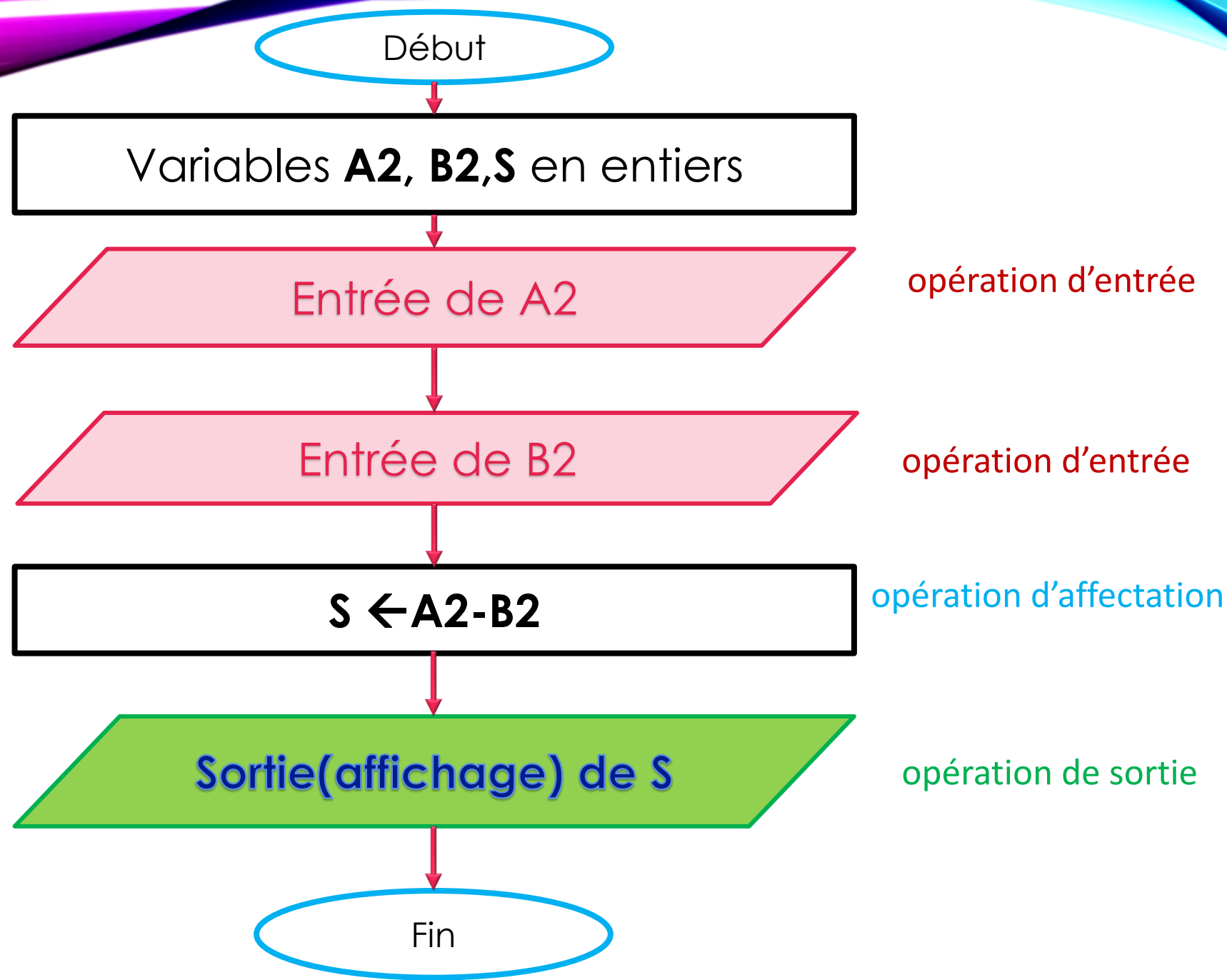


Algorithme solution
du sous-problème
P1-1a:





Algorithme solution
du sous-problème
P1-1b:



COMMENT ÉCRIRE UN ALGORITHME EN PSEUDO-CODE?



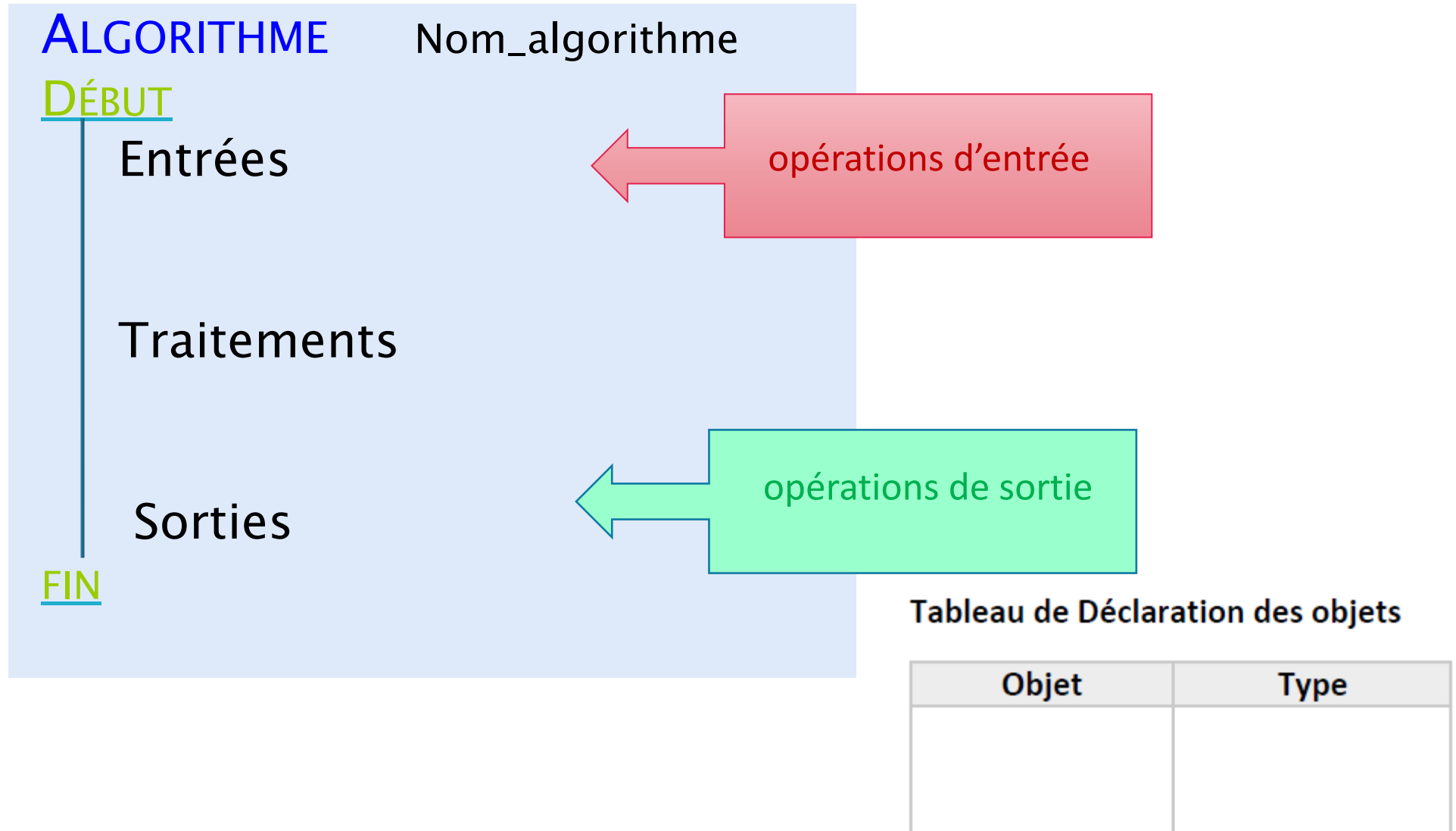
PSEUDO-CODE

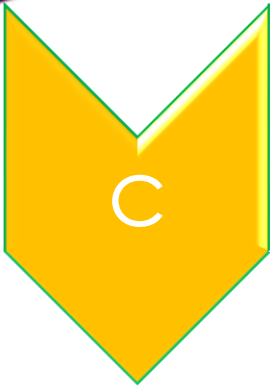
Le pseudo-code est un langage pour **exprimer clairement et formellement un algorithme**.

Il exprime des idées formelles dans une **langue près du langage naturel** de ses usagers (pour nous, le français) en lui **imposant une forme rigoureuse(syntaxe)**.

Il n'y a pas de standard (et donc pas de normalisation) pour l'écriture d'un algorithme en pseudo-code mais seulement quelques **conventions** partagées par un grand nombre !

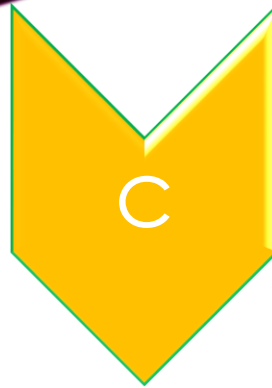
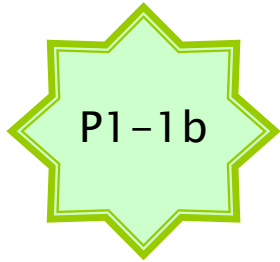
Structure d'un algorithme





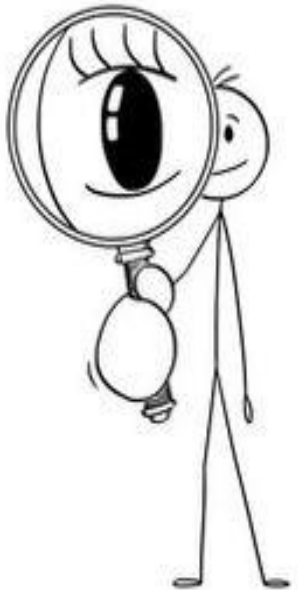
- Algorithmme du S-P(P1-1a)

Variables	A1,B1, Add : entiers
Instructions d'entrée	Saisir le contenu de A1 Saisir le contenu de B1
Traitements	Add reçoit le résultat de l'opération mathématique (addition) $A1+B1$
Instructions de sortie	Afficher le contenu de Add

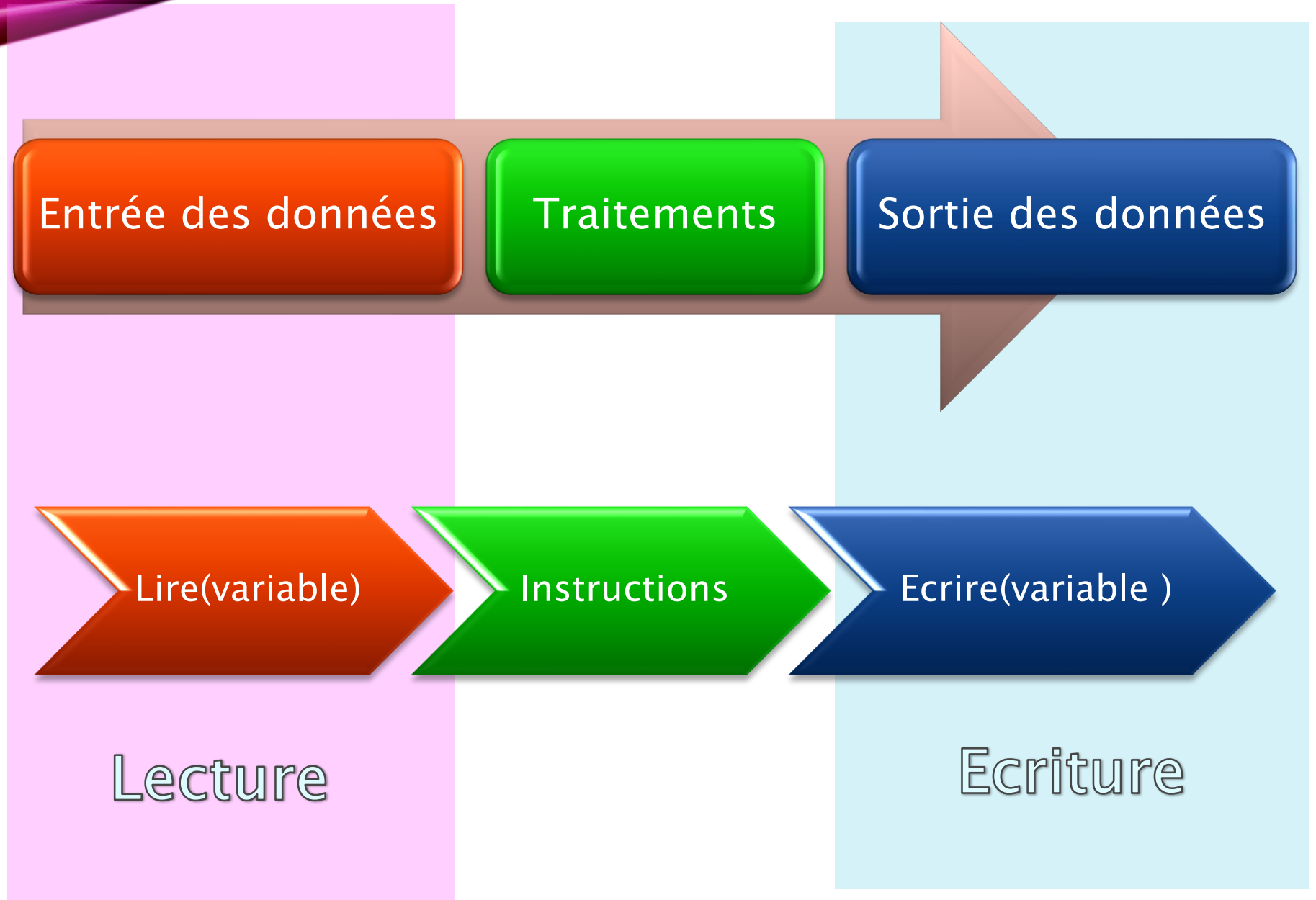


- Algorithmme du S-P(P1-1b)

Variables	A2, B2, S : entiers
Instructions d'entrée	Saisir le contenu de A2 Saisir le contenu de B2
Traitements	S reçoit le résultat de l'opération mathématique (soustraction) $A2 - B2$
Instructions de sortie	Afficher le contenu de S



Syntaxe Algo:



ALGORITHME P1-1a_addition

DÉBUT

Lire(A1)
Lire(B1)

Lecture: Entrée de données

Add \leftarrow A1 + B1

Instruction

Ecrire(Add)

Ecriture: Sortie(affichage)
de données

FIN

T.D.O:

Objets	Types
A1	Entier
B1	Entier
Add	Entier

à noter!

- L'opération d'entrée **LIRE(Nom_variable)**: permet de stocker les valeurs saisies au clavier par les utilisateurs dans l'espace mémoire déjà réservé.



à noter!

- L'instruction d'affectation permet de stocker les valeurs des variables dans l'espace mémoire déjà réservé

Nom_variable ← Valeur numérique

Nom_variable ← Valeur d'une Expression



à noter!

- L'opération de sortie **Ecrire(Nom_variable)**: permet d':
 - ✓ Afficher à l'écran le contenu d'une variable

Exemple:

Ecrire(N)



à noter!

- L'opération de sortie **Ecrire("message texte ")**: permet d':
 - ✓ Afficher à l'écran le texte écrit entre les guillemets

Exemple:

Ecrire("Bonjour INFO_2")

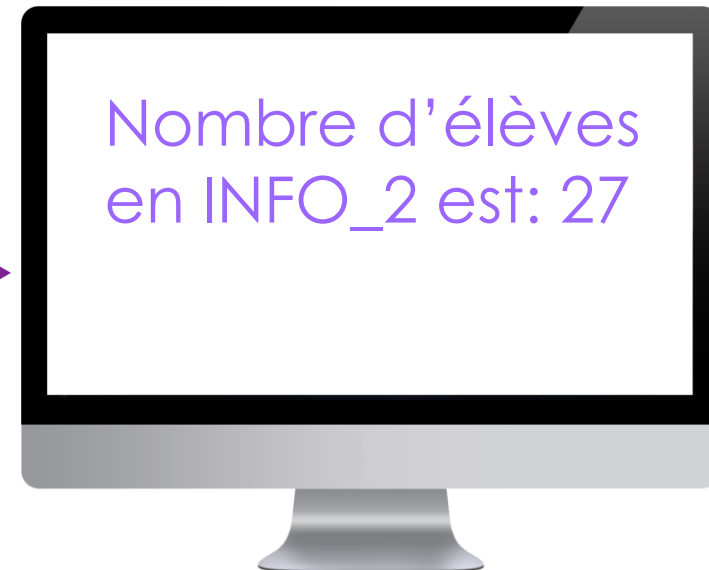


à noter!

- L'opération de sortie **Ecrire("message texte ",Nom_variable):** permet de:
 - ✓ Afficher à l'écran le texte écrit entre les guillemets suivi du contenu (valeur) de la variable
→ C'est l'affichage mixte

Exemple:

Ecrire("Nombre d'élèves en INFO_2 est:", NB_el) ↔



AMÉLIORATIONS DU PSEUDO-CODE

Pour faciliter la compréhension de la solution on va réécrire le pseudo-code

ALGORITHME

P1-1a_addition

DÉBUT

Ecrire("Donner la valeur de A1:")

Lire(A1)

Ecrire("Donner la valeur de B1:")

Lire(B1)

$\text{Add} \leftarrow A1 + B1$

Ecrire("La somme de", A1, "et de", B1, "est:", Add)

FIN

T.D.O:

Objets	Types
A1	Entier
B1	Entier
Add	Entier



On peut Modifier un algorithme existant

ALGORITHME P1-1 **b_soustraction**

DÉBUT

Ecrire("Donner la valeur de **A2**:")

Lire(A2)

Ecrire("Donner la valeur de **B2**:")

Lire(B2)

S ← **A2 - B2**

Ecrire("La **soustraction** de", **A2**, "et de", **B2**, "est:", **S**)

FIN

L'algorithme
précédent
devient:

T.D.O:

Objets	Types
A2	Entier
B2	Entier
S	Entier


A stick figure is sitting at a desk, holding a pen to its chin in a thinking pose. A thought bubble above its head contains the text "Algorithme réalisé veut dire solution trouvée".

Algorithme réalisé
veut dire solution
trouvée

Algorithme
=
Solution

A stick figure is sitting at a desk, looking stressed with sweat drops on its head. It is holding a pen to its chin. A thought bubble above its head contains the text "Comment vérifier si la solution trouvée est correcte?".

Comment vérifier si
la solution trouvée
est correcte?

A stick figure is sitting at a desk, holding a pen to its chin. A thought bubble above its head contains the text "On doit implémenter l'Algorithme en un langage de programmation".

On doit implémenter
l'Algorithme en un
langage de
programmation

CODIFICATION

Activité (CODIFICATION)

Essayer de réécrire l'algorithme avec un langage compréhensible par l'ordinateur (Python)



Vous pouvez vous servir du
Mémento_modifié publié dans
Classroom

Mémento_1

Blocs d'instructions

instruction parente :

bloc d'instructions 1...

⋮

instruction parente :

bloc d'instructions 2...

⋮

instruction suivante après bloc 1

🔧 régler l'éditeur pour insérer 4 espaces à la place d'une tabulation d'indentation.

entier, flottant, booléen, chaîne, octets

Types de base

```
int 783 0 -192 0b010 0o642 0xF3
      zéro      binaire      octal      hexa
float 9.23 0.0 -1.7e-6
bool True False
str "Un\nDeux"
      retour à la ligne échappé
      'L\''âme'
      ' échappé
bytes b"toto\xfe\775"
      hexadecimal      octal
      📌 immutables
```

Chaîne multiligne :

```
"""X\tY\tZ
1\t2\t3"""
```

tabulation échappée

Identificateurs

pour noms de variables,
fonctions, modules, classes...

a...zA...Z_ suivi de **a...zA...Z_0...9**

- ☐ accents possibles mais à éviter
- ☐ mots clés du langage interdits
- ☐ distinction casse min/MAJ

☺ a toto x7 y_max BigOne

☹ 8y and for

= Variables & affectation

🔧 affectation ⇔ **association** d'un nom à une valeur

- 1) évaluation de la valeur de l'expression de droite
- 2) affectation dans l'ordre avec les noms de gauche

x=1.2+8+sin(y)

a=b=c=0 affectation à la même valeur

y,z,r=9.2,-7.6,0 affectations multiples

a,b=b,a échange de valeurs

a,*b=seq } dépaquetage de séquence en
***a,b=seq** } élément et liste

x+=3 incrémentation ⇔ **x=x+3** et

x-=2 décrémentation ⇔ **x=x-2** */=

x=None valeur constante « non défini » %=

del x suppression du nom x ...

```
print("v=", 3, "cm :", x, ", ", y+4)
```

éléments à afficher : valeurs littérales, variables, expressions

Options de **print**:

- ☐ **sep=" "** séparateur d'éléments, défaut espace
- ☐ **end="\n"** fin d'affichage, défaut fin de ligne
- ☐ **file=sys.stdout** print vers fichier, défaut sortie standard

s = input("Directives:")

🔧 **input** retourne toujours une **chaîne**, la convertir vers le type désiré (cf. encadré Conversions au recto).

Affichage

Saisie