

De la spécification à l'algorithme (partie 1)

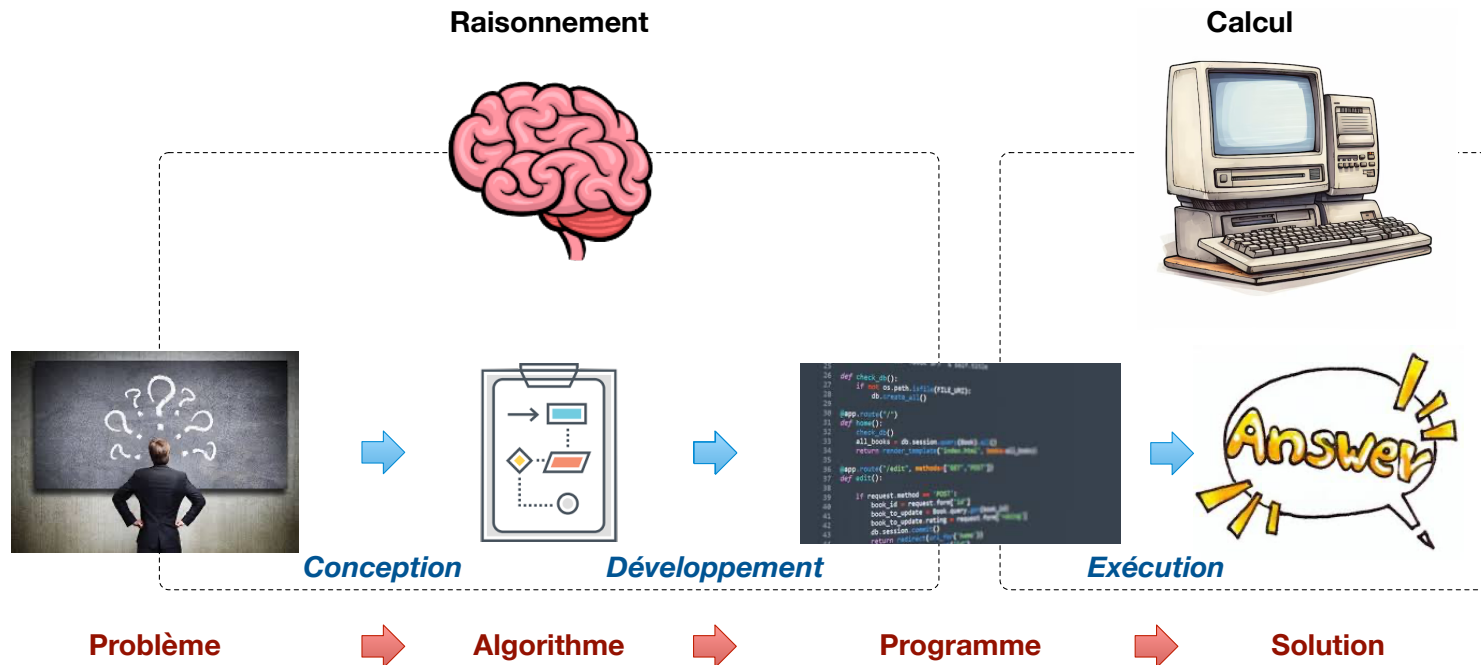
Sébastien Jean

IUT de Valence
Département Informatique

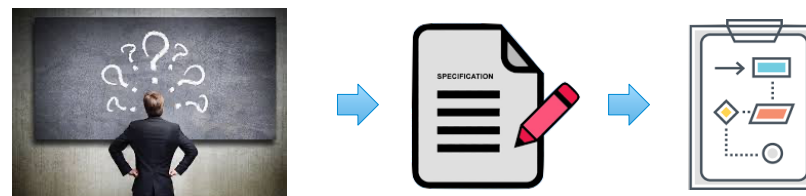
v1.0, 22 septembre 2025

Rappels

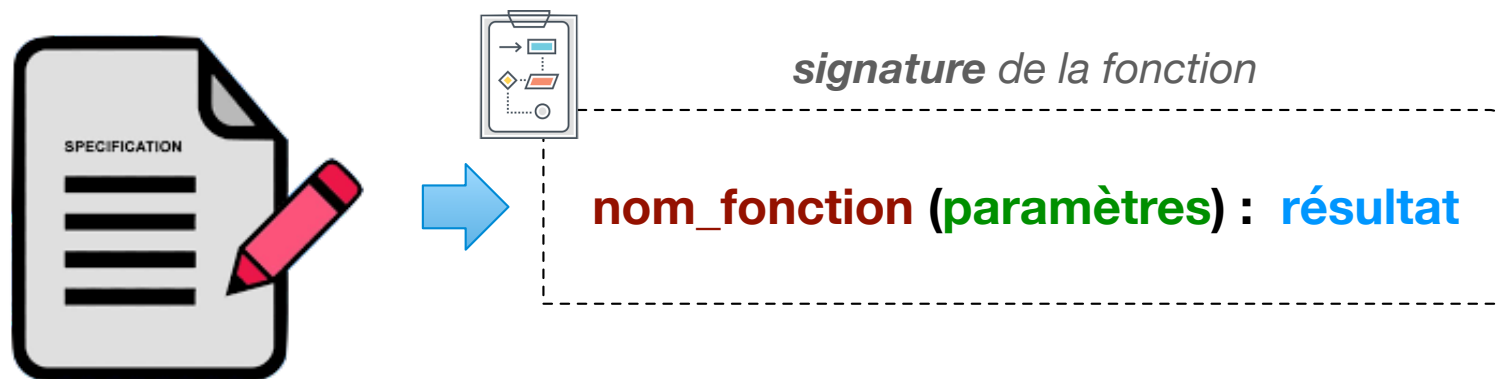
- Du problème à la solution



- Du problème à l'algorithme (focus)



De la spécification à la fonction



- L'**algorithme** est réalisé par une **fonction** (nommée)
- Les **paramètres** (nommés, typés) sont des **données en entrée**
- Le **résultat** (typé) est une (ou un ensemble de) **donnée(s) en sortie**

D'où viennent les données ?

- Si on se place dans la perspective d'un **programme** :
 - Une **donnée d'entrée** peut provenir explicitement d'une **saisie**
 - On ne l'exprimera donc pas avec un paramètre dans la signature de la fonction
 - Le **résultat** peut explicitement n'être destiné qu'à être **affiché**
 - On ne l'exprimera donc avec comme type retourné (aucun)
 - N.B. : Les fonctions ne retournant pas de résultat sont parfois appelées **procédures**
- N.B. : *on peut (et il est préférable de) rendre plus réutilisable un algorithme en considérant qu'il prend en paramètre une donnée lue par un autre algorithme et qu'il retourne un résultat qui sera affiché par un autre algorithme*

Signature de la fonction de calcul de la factorielle

Définition de la factorielle d'un nombre

En mathématiques, la **factorielle** d'un entier naturel n est le **produit des nombres entiers strictement positifs inférieurs ou égaux à n** .

Spécification du calcul de la factorielle

- Donnée d'entrée : n , entier
- Donnée de sortie : r , entier
- Pré-condition : $n \geq 0$
- Post-condition : $r = n! = 1 \times 2 \times \dots \times (n - 1) \times n$

Signature de la fonction factorielle

- **factorielle** (n : entier) : entier

D'autres d'exemples de signatures

Enoncé du problème

Un magasin dispose d'un **nombre donné de pièces** en stock. **Combien reste t'il de pièces** en stock après qu'on en ait **prélevé un nombre donné** ?

Spécification du problème

- Donnée d'entrée : **n** , entier
- Donnée d'entrée : **p** , entier
- Donnée de sortie : **r** , entier
- Pré-condition : **$0 \leq p \leq n$**
- Post-condition : **$r + p = n$**

Signature de la fonction



D'autres d'exemples de signatures

Enoncé du problème

Un magasin dispose d'un **nombre donné de pièces** en stock. **Combien reste t'il de pièces** en stock après qu'on en ait **prélevé un nombre donné** ?

Spécification du problème

- Donnée d'entrée : n , entier
- Donnée d'entrée : p , entier
- Donnée de sortie : r , entier
- Pré-condition : $0 \leq p \leq n$
- Post-condition : $r + p = n$

Signature de la fonction

- **stock_restant** (n : entier, p : entier) : entier

D'autres d'exemples de signatures

Enoncé du problème

Trouver le **maximum** entre **deux nombres**

Spécification du problème

- Donnée d'entrée : **a** , réel
- Donnée d'entrée : **b** , réel
- Donnée de sortie : **m** , réel
- Pré-condition : (aucune)
- Post-condition : **$m = a$ si $a \geq b$, $m = b$ si $a < b$**

Signature de la fonction



D'autres d'exemples de signatures

Enoncé du problème

Trouver le **maximum** entre **deux nombres**

Spécification du problème

- Donnée d'entrée : **a** , réel
- Donnée d'entrée : **b** , réel
- Donnée de sortie : **m** , réel
- Pré-condition : (aucune)
- Post-condition : **$m = a$ si $a \geq b$, $m = b$ si $a < b$**

Signature de la fonction

- **max** (**a** : réel, **b** : réel) : réel

D'autres d'exemples de signatures

Enoncé du problème

Déterminer si un **nombre** est **pair**

Spécification du problème

- Donnée d'entrée : ***n***, **entier**
- Donnée de sortie : ***pair***, **booléen**
- Pré-condition : (aucune)
- Post-condition : ***pair = vrai si $n \text{ modulo } 2 = 0$, $pair = faux$ sinon***

Signature de la fonction



D'autres d'exemples de signatures

Enoncé du problème

Déterminer si un **nombre** est **pair**

Spécification du problème

- Donnée d'entrée : ***n***, **entier**
- Donnée de sortie : ***pair***, **booléen**
- Pré-condition : (aucune)
- Post-condition : ***pair = vrai si $n \text{ modulo } 2 = 0$, $pair = faux$ sinon***

Signature de la fonction

- ***est_pair*** (***n : entier***) : **booléen**

Corps de la fonction

Définition (rappel)

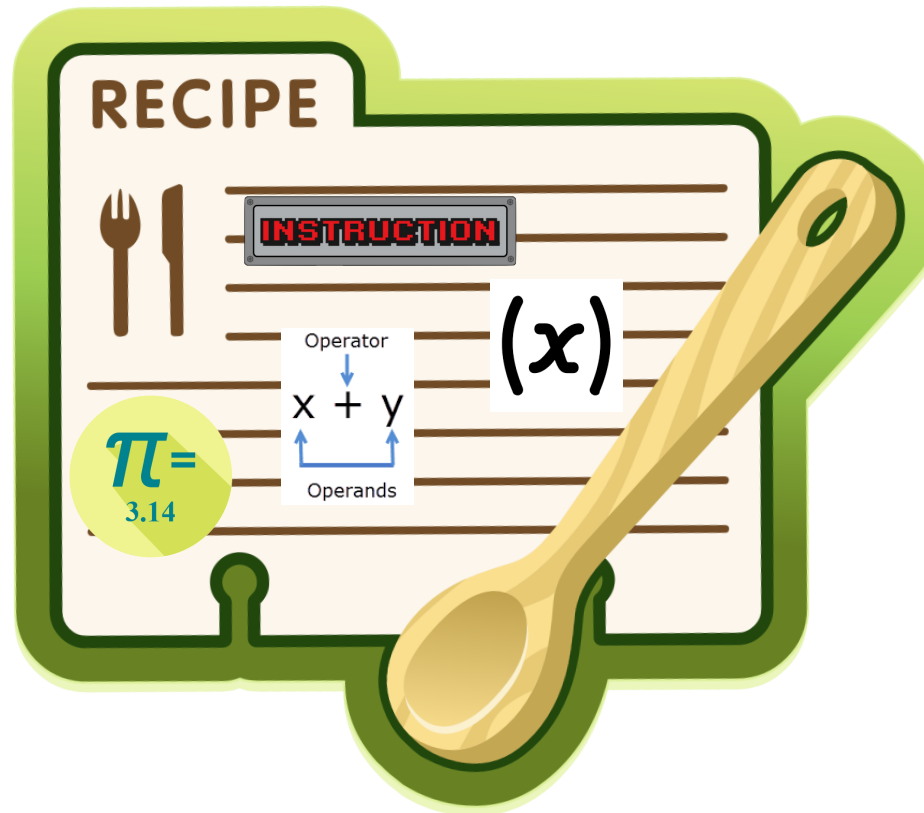
Un **algorithme** est une **suite finie et non ambiguë d'instructions et d'opérations** permettant de **résoudre** une classe de **problèmes**

- La **signature de la fonction**, les **pré et post-conditions** expriment le **contrat** que doit remplir la fonction
- Le **corps de la fonction** exprime la **réalisation de ce contrat**



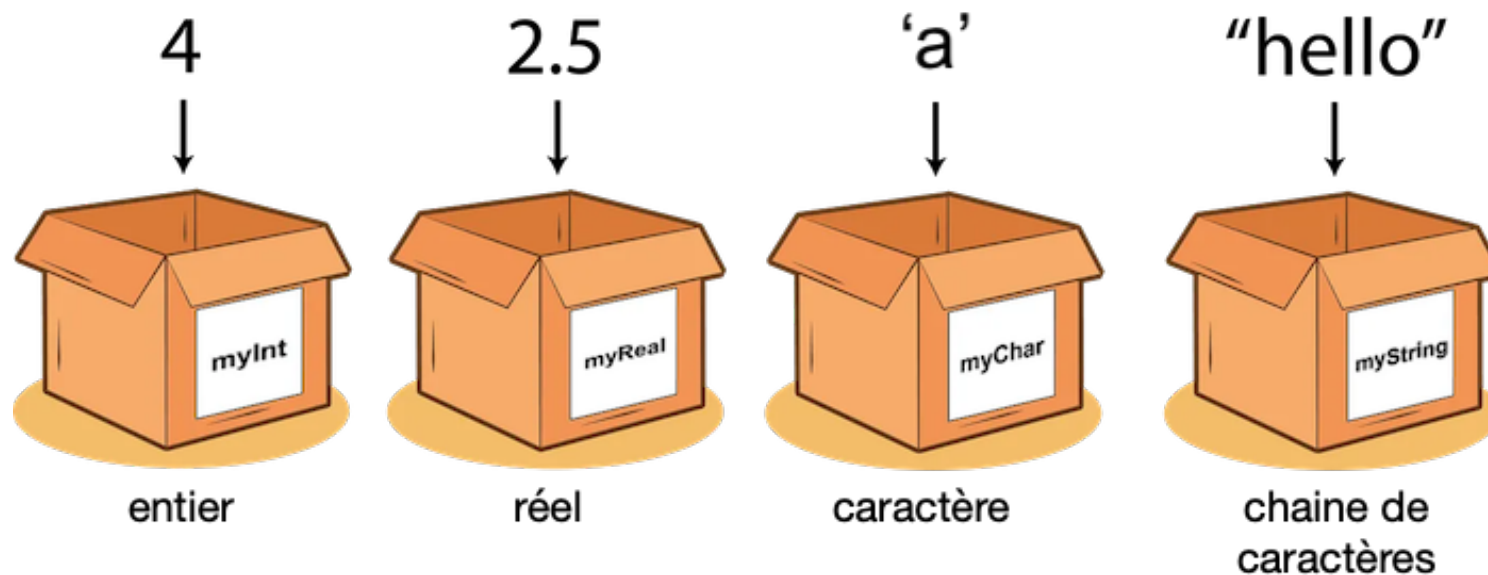
Que trouve t'on dans le corps d'une fonction ?

- Variables, constantes
- Expressions
- Instructions



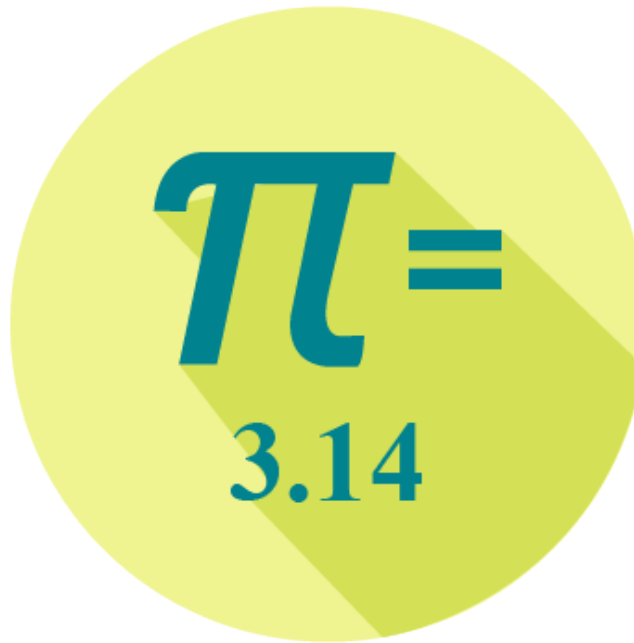
Variables et constantes

- Une **variable** est un **conteneur** qui permet de **mémoriser** une **valeur**
 - Une variable est **nommée** et **typée**
 - On peut **affecter une valeur** à une variable
 - la valeur doit être du type de la variable
 - la valeur précédente est perdue



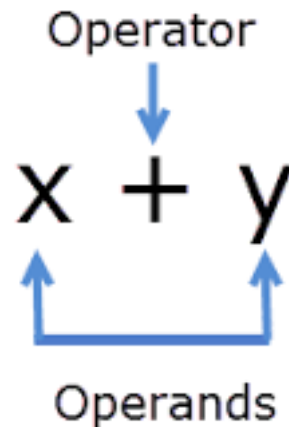
Variables et constantes

- Une **constante** est similaire à une **variable** mais sa **valeur** est **fixe**
 - Une constante est **nommée** et **typée**
 - La valeur de la constante est **fixe**, on ne peut pas la réaffecter



Expressions

- Une *expression* peut être :
 - Une **valeur immédiate** → par ex. 42
 - La **valeur d'une variable** (désignée par son nom) → par ex. x
 - La **valeur d'une constante** (désignée par son nom) → par ex. PI
 - Le **résultat de l'appel à une fonction**
 - Un **opérateur** appliqué à une (*unaire*) ou deux (*binaire*) **expressions**



Instructions

- Une *Instruction* peut être :
 - L'**affection** d'une expression à une variable
 - Une **structure de contrôle**
 - L'**appel à une fonction** (ne retournant pas de résultat)
 - Une **alternative**
 - Une **itération**
 - Une **séquence d'instructions**,
 - Le **renvoi du résultat final**

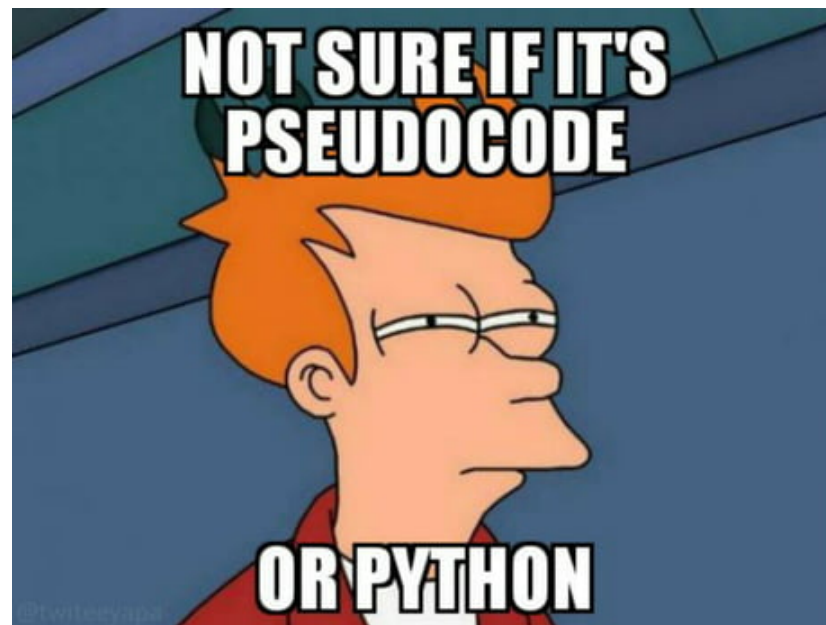


Alternatives et itérations

- Une **alternative** permet l'**exécution unique et conditionnelle** d'une ou plusieurs instructions
- Une **iteration** permet l'**exécution répétée et conditionnelle** d'une ou plusieurs instructions

Pseudo-code

- Pour **exprimer le corps de la fonction**, il faut un **langage** qui soit :
 - Suffisamment **proche du langage naturel**
 - **Indépendant d'un langage de programmation** particulier
- On appelle ce langage ***pseudo code***
 - **Il n'existe pas de standard**, mais toutes les formes de pseudo code permettent d'exprimer les mêmes choses



Pseudo-code : forme générale d'un algorithme

- Syntaxe :

```
FONCTION nom (paramètres) : type du résultat  
    (déclarations des variables et constantes)  
    (instructions)  
  
FIN FONCTION
```

- N.B. : l'**indentation** facilite la lecture

Pseudo-code : déclarations des variables et constantes

- Syntaxe :

VARIABLE somme : entier

CONSTANTE MAX : entier (42)

Pseudo-code : affectation d'une variable, appel de fonction

- Syntaxe :

somme ← 42

taille ← LIRE_ENTIER()

AFFICHER("le message")

Pseudo-code : opérateurs arithmétiques

- Syntaxe :

somme	←	-	somme	//	Opposé
somme	←	somme	+	1	// Addition
somme	←	somme	-	1	// Soustraction
somme	←	somme	*	3	// Multiplication
x	←	somme	/	2	// Division (résultat réel)
somme	←	somme	div	2	// Division entière
somme	←	somme	mod	2	// Reste de la division entière (modulo)

Pseudo-code : opérateurs logiques

- Syntaxe :

`est_pair ← VRAI`

`est_pair ← FAUX`

`est_pair ← NON est_pair`

`est_pair ← est_pair OU VRAI`

`est_pair ← est_pair ET VRAI`

`est_pair ← est_pair OUEX VRAI // OU Exclusif`

- N.B. : l'expression utilisant l'opérateur logique est de type booléen

Pseudo-code : opérateurs relationnels

- Syntaxe :

condition	←	somme	≤	2
condition	←	somme	<	2
condition	←	somme	=	2
condition	←	somme	≠	2
condition	←	somme	≥	2
condition	←	somme	>	2

- N.B. : l'expression utilisant l'opérateur relationnel est de type booléen

Pseudo-code : retour de résultat

- Syntaxe :

RETOURNER 1

RETOURNER "Abitbol"

RETOURNER x

RETOURNER a mod 2

- N.B. : plus **aucune instruction ne peut être exécutée** après un RETOURNER (**sortie de la fonction en exportant le résultat**)

Corps de la fonction de calcul du stock restant

Spécification du problème

- Donnée d'entrée : n , entier
- Donnée d'entrée : p , entier
- Donnée de sortie : r , entier
- Pré-condition : $0 \leq p \leq n$
- Post-condition : $r + p = n$

Signature de la fonction

- **stock_restant** (n : entier, p : entier) : entier

Corps de la fonction



Corps de la fonction de calcul du stock restant

```
FONCTION stock_restant (n : entier , p : entier) : entier
```

```
    VARIABLE resultat : entier
```

```
    resultat  $\leftarrow$  n - p
```

```
    RETOURNER resultat
```

```
FIN FONCTION
```

Corps de la fonction de calcul de la surface d'un cercle

Définition de la surface d'un cercle

La **surface** d'un cercle de **rayon** r est égale à $\pi \times r^2$

Spécification du problème

- Donnée d'entrée : r , **réel** (le rayon)
- Donnée de sortie : s , **réel** (la surface)
- Pré-condition : $r \geq 0$
- Post-condition : $s = \pi \times r^2$

Signature de la fonction

- **surface** (r : **réel**) : **réel**

Corps de la fonction



Corps de la fonction de calcul de la surface d'un cercle

```
FONCTION surface (r : réel) : réel
```

```
VARIABLE resultat : réel
```

```
CONSTANTE PI : réel (3.14)
```

```
    resultat  $\leftarrow$  PI * r * r
```

```
    RETOURNER resultat
```

```
FIN FONCTION
```

Fin !

