Chapitre 1

Introduction, Etat de l'art et définitions principales

1. Quelques définitions pour fixer les bases
2. Un peu d’histoire sur la programmation
3. Langages de programmation et classifications

1 Quelques définitions pour fixer les bases

2 Un peu d’histoire sur la programmation

3 Langages de programmation et classifications

Objectifs et essence de la programmation

- ❑ Résoudre efficacement des problèmes +/- complexes à l'aide d'un ordinateur

- ❑ Créer des logiciels ou applications capables de :
  - ✓ apporter des solutions dans plusieurs domaines : banque, assurance, industrie, etc
  - ✓ fonctionner sur plusieurs supports électroniques : ordinateur, tablette, phone, etc

- ❑ Essence : **Algorithmique**

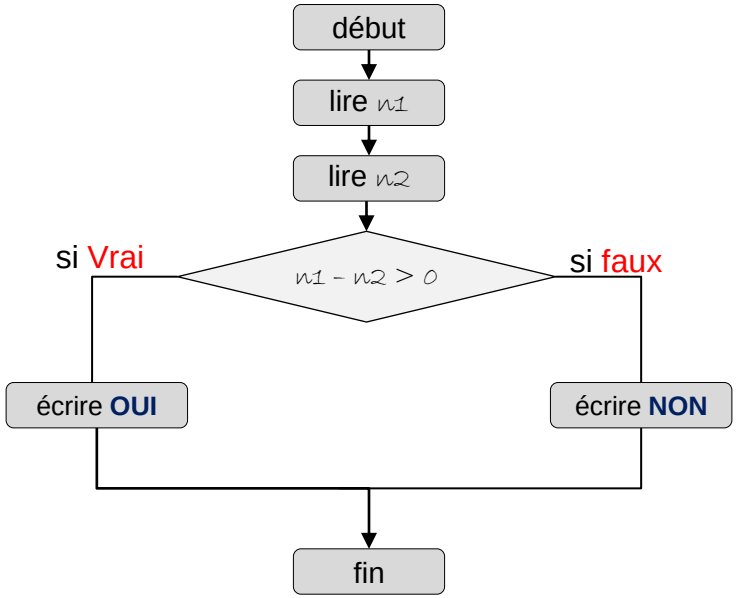
Algorithme ?

Ensemble de règles opératoires (suite finie et non ambigüe d'instructions et d'opérations) ordonnancées pour résoudre un problème donné. C'est la description/spécification (issue des capacités imaginatives d'un humain) d'un processus à exécuter pour apporter solution à un problème.

Exemple de problème

Soit deux nombres entiers $n1$ et $n2$ pris au hasard. Répondre OUI si la différence entre les deux est positif et NON sinon.

Spécification schématique

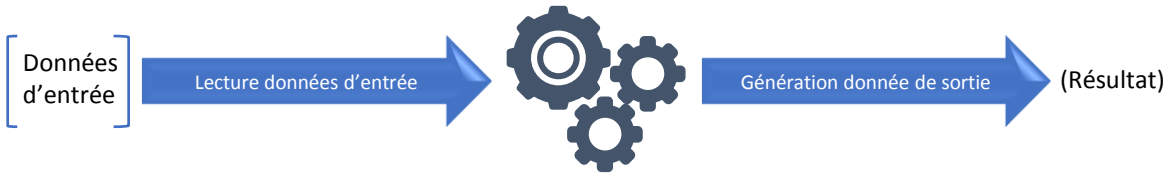


Spécification textuelle

```
début
    lire n1
    lire n2
    si n1 – n2 > 0 alors
        écrire OUI
    sinon
        écrire NON
    fin
fin
```

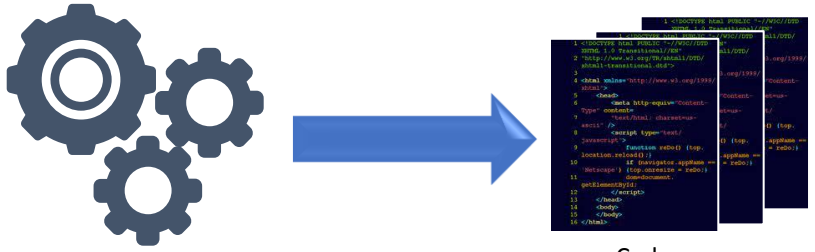
Programme informatique ?

- Succession d'instructions exécutables par ordinateur
- C'est une traduction des règles opératoires conçues via un algorithme en un **langage** compréhensible par un ordinateur
- Il a pour but de récupérer des **données en entrée**, de les **traiter** et de produire une **donnée résultat en sortie**



Le code source ?

- Ensemble d'instructions écrits par un humain dans un **langage** compréhensible par ordinateur et stocké dans un ou plusieurs fichiers texte
- Représente physiquement le corps d'un programme informatique



Programme

Code source

Qu’est-ce donc la Programmation ?

Ensemble de toutes les actions opérationnelles concourant à la création et la maintenance d’un programme informatique. C’est-à-dire, la rédaction de son code source, la vérification de son bon fonctionnement, le débogage et la résolution d’erreurs le cas échéant.

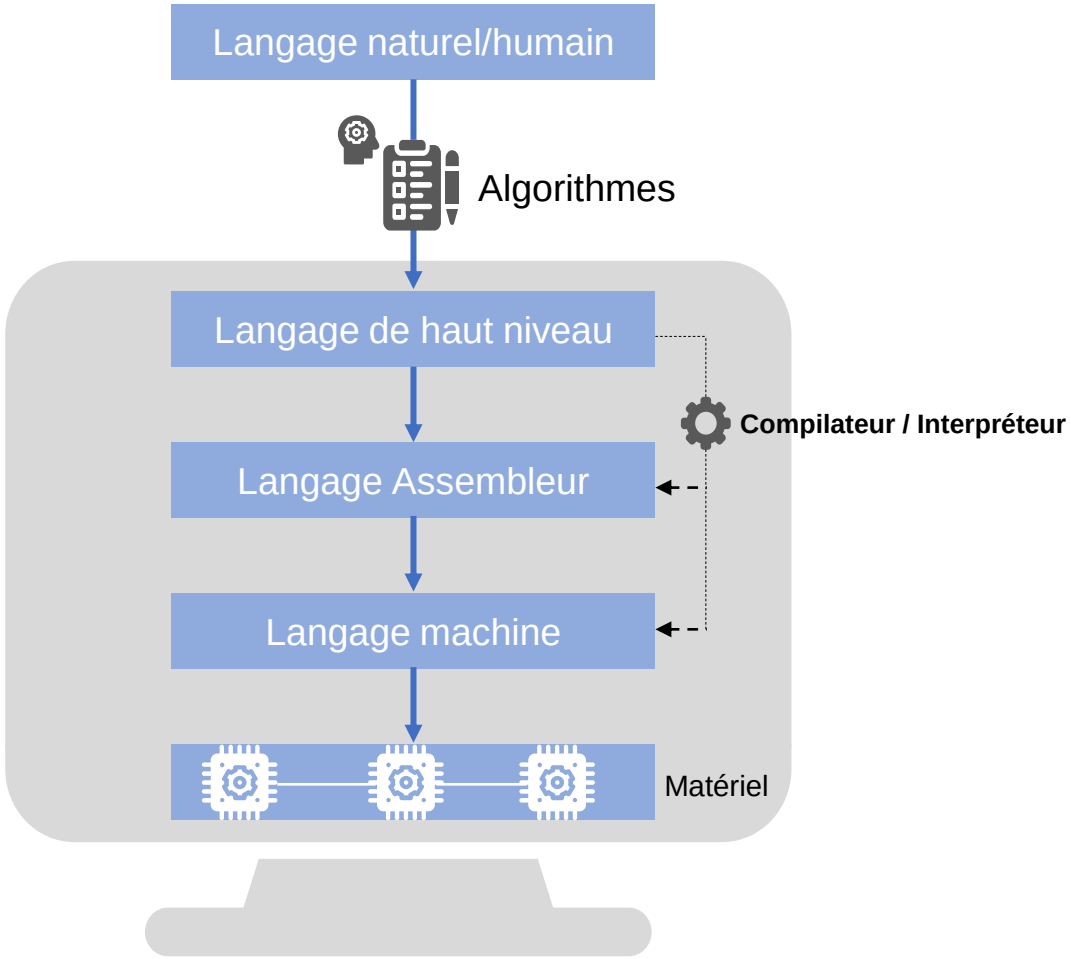
Langage de programmation ? (1/3)

❑ Comme un langage naturel (français, anglais, afrikaans, allemand, etc), c'est un ensemble composé d'un **alphabet**, d'un **vocabulaire**, de **symboles**, de **règles syntaxiques et sémantiques** qui permettent à un humain de données des instructions à un ordinateur.

❑ Il s'agit d'une abstraction au dessus des mécanismes internes de calcul d'un ordinateur. Il est généralement accompagné d'un environnement (appelé **compilateur** ou **interpréteur**) qui permet de le traduire en un **langage de bas niveau** plus proche des circuits électroniques de l'ordinateur

Langage de programmation ? (2/3)

Niveaux des langages et compilation

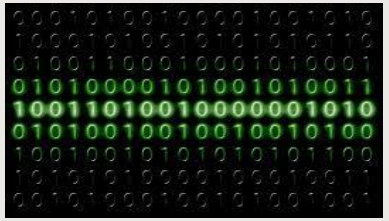


❑ **Langage de haut niveau :** tout langage de programmation qui fait abstraction du langage machine. Ex : Java, C, C++

❑ **Compilateur / Interpréteur** : système (qui lui-même est un programme) capable de traduire un langage de haut niveau en langage de bas niveau

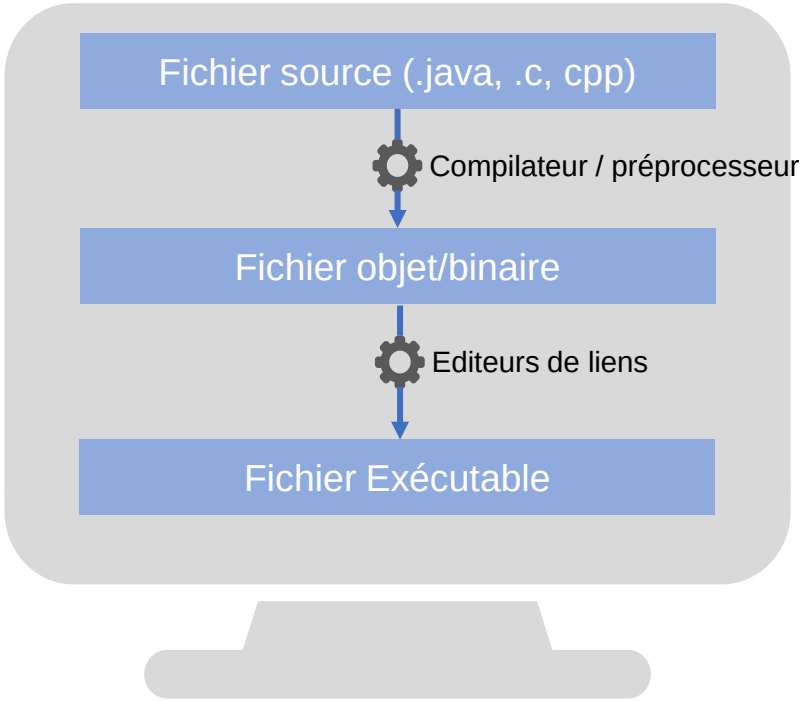
❑ **Assembleur :** langage de bas niveau représentant une étape intermédiaire avant le langage machine

❑ **Langage machine :** langage binaire composé d'une série de 0 et de 1 interprétable par un processeur pour réaliser les calculs par actionnement des transistors et commutateurs électroniques de l'ordinateur



Langage de programmation ? (3/3)

Phases de compilation d'un code source



La compilation passe par différentes phases, produisant ou non des fichiers intermédiaires

1. **Préprocessing :** le **code source** original est transformé en code source brut. Entre autres les commentaires et les espaces de trop sont enlevés

2. **Compilation en fichier objet :** les fichiers de code source brut sont transformés en un fichier dit objet**,** c'est-à-dire un fichier contenant du **langage machine** ainsi que toutes les informations nécessaires pour l'étape suivante (édition des liens)

3. **Edition de liens :** dans cette phase, l'éditeur de liens (linker) s'occupe d'assembler les fichiers objet en une entité exécutable

Programmeur ?

Personne/informaticien qui utilise un langage de programmation pour traduire en programme des spécifications déjà définies. Il est essentiellement dans les tâches opérationnelles

Développeur ?

Aussi appeler **analyste-programmeur**, c'est une personne qui en plus de la casquette de **programmeur**, a les capacités de spécifier les besoins fonctionnels du logiciel à développer, de réaliser sa conception générale et détaillée et qui enfin a une bonne vision globale de l'ensemble de ses composants

1 Quelques définitions pour fixer les bases

2 Un peu d’histoire sur la programmation

3 Langages de programmation et classifications

Historique de la programmation

Peut être repartie en quatre grandes périodes

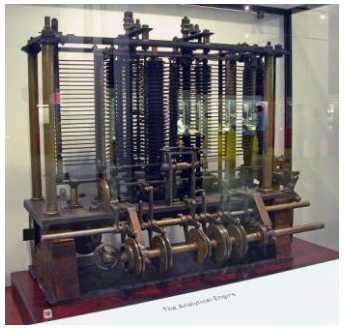
- ❑ **L'ère des événements précurseurs de la programmation**
- ❑ **L'ère des premiers ordinateurs électroniques**
- ❑ **L'ère de « l'adolescence » de la programmation**
- ❑ **L'ère actuelle**

Historique de la programmation (1/4)

L'ère des évènements précurseurs de la programmation

- Avant les années **1940**
- **1834 :** création par l'anglais **Charles Babbage** de la machine mécanique à calculer appelée **machine analytique** utilisant les cartes perforées pour encoder l'information
- **1840 :** l'anglaise **Ada Lovelace** crée l'algorithmique en définissant les principes d'itérations successives dans l'exécution d'une opération
- **1842 :** Ada Lovelace utilise la machine analytique de Babbage pour calculer les nombres de Bernouilli à l'aide des cartes perforées. Ce travail est considéré comme le tout premier programme au monde

Machine analytique

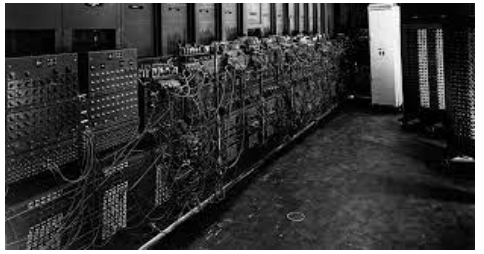


Cartes perforées

Historique de la programmation (2/4)

L'ère de la naissance des premiers ordinateurs électroniques

- Entre les années **1940 - 1960**
- **1945 :** création de l'**ENIAC** (Electronic Numerical Integrator and Computer) par les ingénieurs de l'université de Pennsylvanie **John P. Eckert** et **John W. Mauchly** : 30 tonnes, 30m^3, 18000 tubes électroniques, mais 1000 fois plus rapides que les appareils mécaniques existantes. Résolution des problèmes de la physique nucléaire, météorologie, etc



- **1946 :** **John Von Neumann** analyse les défauts de l'ENIAC et définit les spécifications à la base de l'architecture des **ordinateurs** d'aujourd'hui
- **1951 :** création de l'**Univac I** par Eckert & Mauchly, premier ordinateur commercialisé
- **Décennie 1950 – 1960 :** **Grace Murray Hopper** crée le tout premier compilateur **A0** qui permet de générer du code binaire à partir d'un code source. Création des langages *Fortran*, *Lisp*, *COBOL*, *ALGOL*

Historique de la programmation (3/4)

L'ère de « l'adolescence » de la programmation

- Entre les années **1960 - 1990**
- **Décennie 1960-1970 :**
  - perfectionnement des technologies existantes avec la mise au point de la pensée informatique
  - création du premier langage de programmation orienté objet et de simulation par **Ken Thomson** et **Dennis Ritchie** : ***Simula***
- **1969-1973 :** naissance du langage *C* utilisé jusqu'à nos jours
- **1970 :** naissance du langage *Pascal* (héritier de l'*ALGOL*)
- **1974 :** naissance du langage *SQL*
- **1983 :** naissance du langage *C++*

Historique de la programmation (4/4)

L'ère actuelle de la programmation

- Depuis **1990**
- Naissance des langages interprétés et des premiers langages du Web :
  Python (1991), Ruby (1993), PHP (1995), Java (1995), JavaScript (1996), C# (2000)
- Foisonnement de nouveaux langages du web (Angular, Vue.js, ReactJS, etc) et autres : Kotlin, Groovy, etc
- Naissance et expansion du **Cloud computing** et des langages de programmation orientés Infrastructures
  - Infrastructure As Code
  - Configuration As Code
- Création des langages spécialisés dans l'intégration et le déploiement d'applications
- etc

1 Quelques définitions pour fixer les bases

2 Un peu d’histoire sur la programmation

3 Langages de programmation et classifications

Catégorisation des langages

Plusieurs manières de classifier les langages de programmation…

- ❑ **Généralistes** vs **Spécialisés**
- ❑ **De Haut niveau** vs **De Bas niveau**
- ❑ **Compilés** vs **Interprétés**
- ❑ par **paradigmes**
- ❑ etc

Classification par Spécificité

Les Langages :

- ❑ **Généralistes**
- ❑ **Spécialisés**

Langages généralistes/spécialisés (1/2)

Langage spécialisé : langage de programmation dont les spécifications sont conçues pour répondre aux contraintes d'un domaine d'application précis. On l'appelle aussi **langage dédié** ou **DSL** (Domain Specific language)

- Alphabet, vocabulaire, symboles, syntaxe et règles complètement attachés au domaine d'application métiers et pas réutilisables ailleurs
- Mise en œuvre complètement libre à son créateur. Pas de contrainte sur le formalisme
- Exemples domaines d'application et langages :
  - **calcul scientifique :** *Fortran*, *Matlab*
  - **Base de données :** *SQL*, *4GL*, *SAS*
  - **Echantillonnage sonore :** *Csound*

Langages généralistes/spécialisés (2/2)

Langage généraliste : langage de programmation dont les spécifications sont conçues pour répondre aux contraintes de plusieurs domaines métiers. On le qualifie aussi de **GPL** (General Purpose Language)

- Alphabet, vocabulaire, symboles, syntaxe et règles détachés de tout domaine métiers
- Offre un ensemble de bibliothèques de fonctions génériques à usage transverse
- En générale permet l'ajout de **frameworks** qui étendent ses fonctionnalités
- Quelques exemples :
  - *Java*
  - *C*
  - *C#*
  - *C++*
  - *Python*
  - *PHP*

Classification par hiérarchisation

Les Langages :

- **De haut niveau**
- **De bas niveau**

Langages de haut niveau/de bas niveau (1/2)

Langage de haut niveau : langage de programmation qui fait abstraction du fonctionnement interne de la machine. Il donne la possibilité à un humain d'écrire plus facilement et rapidement un programme en utilisant un vocabulaire proche du langage naturel. Il est souvent qualifié de **langage de 3ème ou 4ème génération**

- ❑ peut être de type généraliste ou spécialisé
- ❑ dispose d'une plateforme/système permettant de le convertir en un langage de bas niveau : **compilateur**
- ❑ Quelques exemples :

- Java	*- C*	*- Fortran*	*- SAS*
- C#	*- C++*	*- Matlab*	*- Csound*
- Python	*- PHP*	*- SQL*	*- 4GL*

Langages de haut niveau/de bas niveau (2/2)

Langage de bas niveau : langage de programmation dont les concepts sont beaucoup plus proche du fonctionnement interne de la machine. Son utilisation est donc pénible, voir impossible lorsqu'on se rapproche de plus en plus du matériel électronique implémenté via les **circuits logiques** dans les transistors

- Permet de manipuler explicitement des registres, des adresses mémoires et les jeux instructions machine (instructions machines qu'un processeur est capable d'exécuter)
- Deux types de langage les plus connus :
  - **Assembleur** : *langage**s** dont la syntaxe est limitée au jeu d'instructions compréhensible par le processeur de la machine*
  - **Langage machine** ou **langage binaire :** *écrit à l'aide de séquences binaire de 0 et 1 (ex :* 0001101001111*). Chaque séquence binaire est une instruction Assembleur*

Classification par mode d'exécution

Les Langages :

- ❑ **Compilés**
- ❑ **Interprétés**

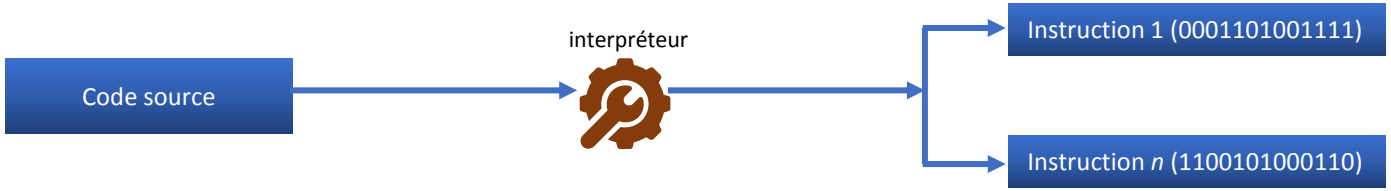
Langages compilés/interprétés (1/2)

Langage compilé : langage de programmation dont le code source d'un programme est traduit en une fois (par un compilateur) en un **fichier exécutable** et en se basant sur l'architecture électronique de la machine sous-jacente.

- Fichier exécutable incluant uniquement le jeu d'instructions du processeur de la machine sur laquelle le code source est compilé
- Le programme est donc exécutable que sur les machines disposant du même jeu d'instructions. Il n'est donc pas exécutable sur tous les ordinateurs
- Quelques exemples : C, C++, Pascal, Ocaml, etc

Langages compilés/interprétés (2/2)

Langage interprété : langage de programmation dont le code source d'un programme est traduit au moment de son exécution. Il n'y a plus une phase arrêtée de génération d'un fichier exécutable**.** Les instructions du code source sont traduites les unes après les autres et exécutées en temps réel



- L'interpréteur est donc un système qui permet tout cela
- Le même code source pourra marcher directement sur tout ordinateur, car c'est lors de la phase d'exécution que l'architecture de la machine exécutante est prise en compte
- Quelques exemples : Java, Python, C#, etc

Classification paradigmatique

La classification des langages par paradigmes correspond à la classification la plus pertinente, car elle catégorise les langages suivant leurs principes de fonctionnement. On parle de **paradigme de programmation**

Un paradigme est un ensemble partagé de croyances et de valeurs, une manière commune de voir les choses.
Dans notre contexte, c'est donc un style fondamental de programmation

Quelques grands groupes de paradigmes :

- **Programmation impérative**
- **Programmation déclarative**
- **Programmation événementielle**
- etc

Classification des langages par paradigmes (1/4)

Programmation impérative : modèle de programmation dit classique ou traditionnel et le plus couramment utilisé. Le principe de base est que toutes les instructions du langage utilisé sont écrites de façon séquentielle et strictement exécutées dans l'ordre d'apparition

Trois sous-catégories :

- **Programmation procédurale :** étend l'approche impérative avec la possibilité de subdiviser la série des instructions en plusieurs parties (appelées **procédures** ou **fonctions**) plus facilement maîtrisables
- **Programmation modulaire :** étend l'approche procédurale sur des programmes à code source est très volumineux, en subdivisant ce dernier en blocs logiques indépendants (**regroupement en fichiers thématiques**) les uns des autres pour plus de clarté
- **Programmation orientée objet : correspond à celle étudiée dans ce cours**

Classification des langages par paradigmes (2/4)

Programmation Déclarative : modèle de programmation dont le principe de base réside dans la **description du résultat final souhaité**. Il s'agit donc en premier lieu de l'« objectif » à atteindre, et non du **« déroulement » des étapes de résolution** comme c'est le cas en programmation impérative. Le code source est donc plus difficile à comprendre en raison de son **haut niveau d'abstraction**, mais est également plus court et plus précis

Trois sous-catégories :

- **Programmation descriptive :** définit un programme qui permet de décrire des structures de données
- **Programmation logique :** définit un programme à l'aide d'un ensemble de faits élémentaires (données d'entrées) les concernant et de règles de **logique mathématique** leur associant des conséquences plus ou moins directes
- **Programmation fonctionnelle :** définit un programme comme un emboîtement de fonctions que l'on peut voir comme des « boîtes noires » imbriquées les unes dans les autres. Chaque boîte possédant des paramètres d'entrée mais une seule en sortie

Classification des langages par paradigmes (3/4)

Il existe plusieurs autres paradigmes de programmation

- **Programmation événementielle :** modèle de programmation basé sur les **évènements**. Un événement représente un message envoyé au programme. Ce dernier ***réagit*** donc à des évènements provenant du système (modification d'un fichier, déclenchement d'une minuterie, etc), ou de l'utilisateur (clic souris, appui sur touche clavier, etc). L'ordre d'exécution des instructions n'est pas important comme dans le cas de la programmation impérative
- **Programmation réactive…**
- **Programmation orientée aspect…**
- **Programmation réflexive…**
- **etc…**

Classification des langages par paradigmes (4/4)

Un langage de programmation peut appartenir à plusieurs paradigme. Ce qui signifie qu'il concentre en lui les principes de base des ces paradigmes de programmation

	Java	Python	C	C++	HTML	XML	Lisp	Haskell	Prolog	Node.js	Android	Angular	C#
Impérative	X	X	X	X									X
Déclarative	X	X			X	X	X	X	X				
Evénementiel	X									X	X	X	X


SUCCESS