Dev&script - Introduction à la programation

> ***DISCLAIMER : Cette page est en cours de rédaction et son contenu peut être faux ou inexact. Merci de lire cette page avec toutes les précautions nécessaires.***

> Titre de la page : <Techno> - <Titre de la procédure / du cours> (à supprimer une fois le titre placé)

Difficulté*	Difficulté : Notions :

*placer l'icône correspondant à la difficulté. supprimer cet encarts et le tableau.

				
Débutant	Novice	Intermédiaire	Confirmé	Expert

I. Introduction

Indications de mise en page (peut être supprimé)

> *Ceci est une citation.*
>
> *- par *****

Ceci est une information.

Ceci est une astuce ou optimisation.

Ceci est est un avertissement.

Ceci est un prérequis.

```
Ceci est un bloc de code
une série de commandes
```

Ceci est une section repliable pour une étape optionnelle ou un fichier de configuration

```
Ceci
est
un
fichier
de
configuration
```

colonne 1	Colonne 2	Colonne 3
ceci	est	un tableau

Prérequis Matériels	
core	GUI

• 1 vcpu • 2 Gb RAM • 20Gb dd	• 2 à 4 vcpu • 2 à 4 Gb RAM • 60Gb dd

II. Chapitre1

2.1 Section 1

2.1.1 Sous-section

1 Comprendre ce qu’est un programme et un langage

Objectif de la section

Donner aux étudiants une vision claire de ce qu’est la programmation, pourquoi elle existe, et comment les langages ont évolué.

Résumé

On explique la notion de **machine**, de **binaire**, puis l’apparition de l’assembleur et des langages haut niveau. Les étudiants comprennent que programmer, c’est **donner des instructions à une machine**, et que les langages sont des outils pour simplifier cette communication.

Points clés

- Qu’est-ce qu’un programme ?

- Binaire → assembleur → langages haut niveau
- Pourquoi les langages évoluent
- Notion de compilation/interprétation (premier aperçu)

2 Modes d’exécution : compilé vs interprété

Objectif

Comprendre comment un programme “vit” et s’exécute.

Résumé

On distingue les langages **compilés** (transformés en binaire avant exécution) et **interprétés** (lus ligne par ligne). On montre les impacts sur la vitesse, la portabilité, et les usages.

Points clés

- Compilation : C, Rust, Go
- Interprétation : Python, JavaScript
- Cas hybrides : Java, C#, PHP
- Avantages / inconvénients
- Impact sur le développement

3 Domaines de programmation : frontend, backend, scripts, mobile, IA

□□ Objectif

Montrer que “coder” n’est pas une seule activité, mais une multitude de métiers et de contextes.

□□ Résumé

On présente les grands domaines : **frontend**, **backend**, **DevOps**, **mobile**, **embarqué**, **IA**, etc. Les étudiants comprennent où se situent les langages et pourquoi certains sont spécialisés.

□□ Points clés

- Frontend : HTML, CSS, JS
- Backend : Python, PHP, Java, Node.js
- DevOps : Bash, YAML, Terraform
- Mobile : Swift, Kotlin
- Embarqué : C, MicroPython
- IA : Python, R
- Vision globale des écosystèmes

4□ Paradigmes de programmation

□□ Objectif

Comprendre les différentes façons de structurer la pensée algorithmique.

□□ Résumé

On introduit les paradigmes : **procédural**, **objet**, **fonctionnel**, **déclaratif**. On montre que ce sont des manières de penser le code, pas des langages.

□□ Points clés

- Procédural : séquences, fonctions
- Objet : classes, instances, encapsulation

- Fonctionnel : immutabilité, fonctions pures
- Déclaratif : “dire ce qu’on veut”, pas “comment le faire”
- Transition naturelle vers SQL

5 Formats de données et langages descriptifs

Objectif

Comprendre comment les programmes échangent, stockent et décrivent des données.

Résumé

On présente JSON, YAML, XML, CSV. Les étudiants voient la différence entre **langages de programmation** et **langages descriptifs**.

Points clés

- JSON : structure de données
- YAML : configuration
- XML : historique, encore utilisé
- CSV : données tabulaires
- Pourquoi ces formats sont essentiels (API, DevOps, config)

6 SQL : le langage déclaratif de la donnée

Objectif

Comprendre comment on interagit avec une base de données.

□□ Résumé

SQL est un langage **déclaratif** : on décrit le résultat souhaité. On introduit les requêtes simples et la logique relationnelle.

□□ Points clés

- SQL ≠ langage de programmation
- SELECT, INSERT, UPDATE, DELETE
- Tables, relations, clés
- Lien avec backend et API
- Pourquoi SQL est incontournable

7□ Vibe coding, no-code et low-code

□□ Objectif

Montrer les outils modernes qui permettent de créer sans écrire du code traditionnel.

□□ Résumé

On présente les plateformes visuelles (Node-RED, n8n, Make, Zapier) et les outils IA (Copilot, ChatGPT). Les étudiants comprennent que coder aujourd'hui, c'est aussi **composer**, **assembler**, **automatiser**.

□□ Points clés

- Node-RED : programmation par flux
- n8n / Make : automatisation visuelle
- Zapier : intégration d'applications
- IA + code : génération, assistance, correction

- Comment ces outils démocratisent la programmation

□□ Synthèse générale du cours

Ce cours donne une vision complète de la programmation moderne :

- d’où elle vient
- comment elle fonctionne
- où elle s’applique
- comment les données circulent
- comment les paradigmes structurent la pensée
- comment coder aujourd’hui peut être visuel, assisté, ou automatisé

C’est une progression **logique**

Revision #1
Created 2026-08-12 14:35:53 UTC by Olivier
Updated 2026-08-12 14:37:18 UTC by Olivier