Sommaire

1. Langages de programmation

Étape 1 : Apprendre Python (fondations solides)

1. Installation & environnement

- Installer Python (version 3.x)
- Utiliser un IDE simple (VS Code, PyCharm Community, ou Jupyter Notebook pour débuter)

2. Bases de Python

- Variables et types de données (int, float, str, bool)
- Opérations arithmétiques et logiques
- Entrée/sortie (print, input)
- Structures conditionnelles : if, elif, else
- Boucles : for, while
- Listes, tuples, dictionnaires, ensembles (création, accès, modification)

3. Fonctions

- Définir et appeler des fonctions
- Arguments et valeurs de retour
- Portée des variables (locale vs globale)

4. Programmation orientée objet (POO) basique

- Classes et objets
- Attributs et méthodes
- Constructeur (**init**)
- Encapsulation (private/public)

5. Manipulation de fichiers

- Ouvrir, lire, écrire des fichiers texte
- Gestion des exceptions (try/except)

6. Bibliothèques importantes pour débuter

- NumPy (manipulation de tableaux numériques)
- pandas (manipulation de données tabulaires)
- matplotlib (visualisation simple)

7. Projets simples à réaliser

- Calculatrice
- Gestion d'une liste de tâches
- Analyse simple de données (fichier CSV)

Étape 2 : Apprendre C++ (approfondir la programmation et comprendre la mémoire)

1. Installation

- Installer un compilateur (GCC, MinGW, ou IDE comme Visual Studio ou Code::Blocks)

2. Bases

- Syntaxe C++ (différences avec Python)
- Variables, types de données (int, float, char, bool)
- Entrée/sortie avec cin, cout
- Contrôle de flux (if, switch, boucles for, while)

3. Pointeurs et gestion mémoire

- Déclaration et utilisation des pointeurs
- Allocation dynamique (new/delete)
- Tableaux et pointeurs

4. Programmation orientée objet avancée

- Classes, objets, héritage
- Constructeurs/destructeurs
- Polymorphisme, classes abstraites, interfaces
- Templates (généricité)

5. Bonnes pratiques

- Gestion des erreurs
- Commentaires, documentation

6. Projets simples

- Gestion d'un inventaire (ex : magasin)
- Mini jeu (ex : deviner un nombre)

Étape 3 : Apprendre Java (programmation orientée objet en entreprise)

1. Installation

- JDK (Java Development Kit) et IDE (Eclipse, IntelliJ IDEA)

2. Bases

- Syntaxe Java
- Types de données, variables
- Entrée/sortie (Scanner, System.out.println)
- Contrôle de flux (if, switch, boucles)

3. POO en Java

- Classes, objets, héritage
- Interfaces, classes abstraites
- Gestion des exceptions
- Collections (Listes, Maps, Sets)

4. Concepts avancés

- Threads (multithreading)
- Gestion mémoire avec Garbage Collector
- Packages et modularité

5. Projets simples

- Application console gestion bibliothèque
- Mini application multi-thread (ex : simulation)

Étape 4 : Découvrir .NET (C#)

1. Installation

- Installer Visual Studio Community Edition
- Créer un projet console simple

2. Bases C#

- Syntaxe proche de Java et C++
- Variables, types, opérateurs
- Structures conditionnelles et boucles

3. POO

- Classes, propriétés, méthodes
- Héritage, interfaces, événements

4. Concepts avancés

- LINQ (langage de requêtes intégré)
- Programmation asynchrone (async/await)

5. Projets simples

- Application console CRUD (Create, Read, Update, Delete)
- Mini app Windows Forms (interface graphique)

Étape 5 : Frontend Web avec Angular & React

1. Bases JavaScript/TypeScript

- Variables, types, fonctions
- ES6+ : arrow functions, modules, classes
- DOM et événements

2. Angular

- Concepts clés : composants, services, modules
- Data binding, directives
- Routing
- Appel API avec HttpClient

3. React

- JSX, composants fonctionnels et classes
- Props, state, gestion d’événements
- Hooks (useState, useEffect)
- Gestion d’état avancée (Redux ou Context API)

4. Projets simples

- To-do list interactive
- Application météo avec appel API
- Mini blog avec routing

2. Systèmes & cloud

Étape 1 : Bases des systèmes d'exploitation

A. Unix/Linux

- **Installation** :
  - Installer une distribution Linux (Ubuntu est idéale pour débuter) sur machine virtuelle ou en dual boot
- **Commandes essentielles** :
  - Navigation dans les dossiers : `ls`, `cd`, `pwd`
  - Gestion des fichiers : `cp`, `mv`, `rm`, `mkdir`, `touch`
  - Affichage de contenu : `cat`, `less`, `head`, `tail`
  - Permissions : `chmod`, `chown`
  - Recherche : `find`, `grep`
- **Gestion des processus** :
  - `ps`, `top`, `kill`
- **Gestion des utilisateurs** :
  - `adduser`, `passwd`, `su`, `sudo`
- **Shell scripting (scripts bash simples)** :
  - Variables, conditions, boucles
  - Écrire un script pour automatiser une tâche simple (ex : sauvegarde)

B. Windows (administration basique)

- **Notions clés** :
  - Gestion des fichiers et dossiers (explorateur, commandes CMD/PowerShell basiques)
  - Gestion des utilisateurs et groupes (Panneau de configuration, `net user`)
  - Services Windows (services.msc)
  - Tâches planifiées (Planificateur de tâches)
- **Commandes CMD/PowerShell de base** :
  - `dir`, `cd`, `copy`, `del`
  - `ipconfig`, `netstat`, `tasklist`
  - Scripts PowerShell simples (variables, boucles)

Étape 2 : Concepts avancés

A. Virtualisation

- **Comprendre les concepts** :
  - Qu'est-ce qu'une machine virtuelle (VM) ?

    - Hyperviseurs : Type 1 (bare-metal) vs Type 2 (hosted)
    - Conteneurs (Docker) vs VM : différences, avantages
- **Pratique** :
    - Installer VirtualBox ou VMware Workstation Player
    - Créer et configurer une VM Linux ou Windows
    - Découvrir Docker : installation, commandes de base (`docker run`, `docker build`, `docker ps`)
    - Écrire un Dockerfile simple pour containeriser une application

B. Cloud computing

- **Bases du cloud** :
    - Modèles de service : IaaS, PaaS, SaaS
    - Modèles de déploiement : public, privé, hybride
- **Découvrir les principaux fournisseurs** : AWS, Azure, GCP
    - Créer un compte gratuit (Free Tier)
    - Découvrir les services clés :
        - Calcul : EC2 (AWS), VM (Azure), Compute Engine (GCP)
        - Stockage : S3 (AWS), Blob Storage (Azure), Cloud Storage (GCP)
        - Bases de données managées
- **Pratique** :
    - Déployer une VM dans le cloud
    - Stocker et récupérer des fichiers dans un bucket
    - Comprendre la gestion des identités et accès (IAM)

Projets simples à faire

- Écrire un script bash qui automatise la sauvegarde d'un dossier
- Créer une VM Linux sous VirtualBox, y installer un serveur web Apache
- Dockeriser une petite application Python simple
- Déployer un serveur web basique sur une VM AWS gratuite

3. Ingénierie logicielle & méthodes

Étape 1 : Méthodes de développement

A. Comprendre les méthodologies agiles

- **Principes de l'agilité** :
  - Priorité à la collaboration, adaptation au changement, livraisons fréquentes
- **Scrum** :
  - Rôles : Product Owner, Scrum Master, Équipe de développement
  - Cérémonies : Sprint Planning, Daily Stand-up, Sprint Review, Sprint Retrospective
  - Artéfacts : Backlog produit, Backlog sprint, Increment
- **Kanban** :
  - Visualisation du flux de travail (tableau Kanban)
  - Limitation du travail en cours (WIP)
  - Amélioration continue

B. Cycle en V vs Agile

- **Cycle en V** :
  - Approche séquentielle, chaque phase doit être terminée avant la suivante
  - Phases : spécifications, conception, développement, tests, maintenance
- **Agile** :
  - Approche itérative et incrémentale
  - Collaboration constante avec le client
  - Livraison fréquente de versions utilisables

C. Outils associés

- **Gestion de projet** : Jira, Trello
- **Gestion de versions** : Git (bases : commit, branch, merge, pull request)
- Pratique : créer un projet simple sur Jira ou Trello et gérer des tâches

---

Étape 2 : Qualité logiciel

A. Concepts clés

- **Tests unitaires** : tester des petites unités de code (fonctions, méthodes) isolément
- **Tests d'intégration** : tester les interactions entre plusieurs modules
- **TDD (Test Driven Development)** : écrire les tests avant le code

B. Outils d'automatisation des tests

- **JUnit** (pour Java) : framework de test unitaire
- **Selenium** : tests fonctionnels automatisés pour applications web
- **Jenkins** : serveur d'intégration continue, automatisation des builds et tests

---

Étape 3 : UML (Unified Modeling Language)

A. Notions de base

- **Diagrammes de cas d'utilisation** : représenter les fonctionnalités du système vues par l'utilisateur
- **Diagrammes de classes** : décrire la structure des classes et leurs relations
- **Diagrammes de séquence** : montrer l'ordre des interactions entre objets dans un scénario

B. Importance

- Facilite la conception claire et la communication entre les membres d'une équipe
- Aide à documenter le système

Étape 4 : DevOps

A. Principes clés

- **Intégration continue (CI)** : automatiser la compilation et les tests dès qu'un code est poussé
- **Déploiement continu (CD)** : automatiser la mise en production des versions validées

B. Outils courants

- **Docker** : conteneurisation des applications
- **Kubernetes** : orchestration des conteneurs Docker à grande échelle
- **Jenkins** : pipeline d'intégration et déploiement continu
- **Ansible** : automatisation des tâches d'administration et déploiement

---

Projets simples à faire

- Créer un projet agile avec un backlog dans Jira/Trello
- Écrire des tests unitaires simples avec JUnit (Java) ou pytest (Python)
- Mettre en place un pipeline CI basique avec Jenkins
- Dockeriser une application simple et la déployer sur Kubernetes minikube

4. Réseaux & cybersécurité

Étape 1 : Bases réseaux à maîtriser

A. Modèle OSI & TCP/IP

- **Modèle OSI** (7 couches) : comprendre le rôle de chaque couche (physique, liaison, réseau, transport, session, présentation, application)
- **Modèle TCP/IP** (4 couches) : liaison, internet, transport, application
- Importance et comparaison des deux modèles

B. Protocoles essentiels

- **HTTP / HTTPS** : fonctionnement du protocole web, sécurisation avec TLS
- **FTP** : transfert de fichiers
- **SMTP** : envoi de mails
- **DNS** : résolution de noms de domaine en adresses IP

C. Adressage IP

- IPv4 : structure, classes, masque de sous-réseau, CIDR
- IPv6 : pourquoi IPv6, format, avantages

D. Configuration de base réseau (Unix/Windows)

- Commandes courantes :
  - `ping` (tester la connectivité)
  - `traceroute` / `tracert` (chemin réseau)
  - `netstat` (connexions réseau actives)
- Modifier configuration IP (ifconfig/ip sous Linux, ipconfig sous Windows)

Étape 2 : Concepts fondamentaux en cybersécurité

A. Triade CIA

- **Confidentialité** : protéger les données contre l'accès non autorisé
- **Intégrité** : garantir que les données ne sont pas modifiées
- **Disponibilité** : assurer que les services et données sont accessibles

B. Sécurité réseau

- Pare-feu : rôle, types (filtrage de paquets, proxy)
- VPN : chiffrement des communications et accès distant sécurisé
- Chiffrement SSL/TLS : sécurisation des échanges (HTTPS)

C. Protocoles sécurisés & authentification

- Protocoles : SSH, IPSec, Kerberos
- Méthodes d'authentification : mots de passe, certificats, OTP

Étape 3 : Introduction à la forensic

- Définition et objectifs de la forensic numérique
- Analyse d'incidents : collecte de preuves, suivi des traces
- Récupération de données : outils et techniques basiques

Étape 4 : Pratique avec outils réseau et sécurité

- **Wireshark** : capture et analyse du trafic réseau
- **Nmap** : scanner de ports et détection de services/vulnérabilités
- Exercices :
    - Capturer des paquets HTTP/HTTPS avec Wireshark
    - Scanner un réseau local avec Nmap
    - Identifier des anomalies simples dans un trafic réseau

Projets simples à réaliser

- Configurer un pare-feu simple sous Linux (iptables)
- Mettre en place un VPN basique (OpenVPN)
- Analyser un fichier PCAP (capture Wireshark) pour détecter une attaque simple
- Scanner un réseau simulé avec Nmap et interpréter les résultats

5. Intelligence artificielle & bases de données

Étape 1 : Intelligence artificielle (IA)

A. Concepts fondamentaux

- **Machine Learning (ML)** :
  - Apprentissage supervisé : entraînement avec données étiquetées (ex : classification, régression)
  - Apprentissage non supervisé : détection de motifs sans étiquettes (ex : clustering)
- **Data Science** :
  - Collecte des données : sources, formats
  - Nettoyage des données : traitement des valeurs manquantes, doublons
  - Analyse exploratoire : statistiques descriptives, visualisation

B. Pratique en Python

- Bibliothèques clés :
  - **pandas** : manipulation et analyse des données
  - **scikit-learn** : algorithmes ML classiques (régression, arbres de décision, clustering)
  - **TensorFlow** et **PyTorch** : frameworks pour réseaux de neurones et deep learning
- Exemples :
  - Implémenter une classification simple avec scikit-learn
  - Construire un réseau de neurones basique avec TensorFlow

C. IA appliquée

- Cas concrets :
  - Reconnaissance d'image (ex : classification de photos)
  - Traitement du langage naturel (NLP) : analyse de texte, chatbot basique
  - Systèmes de recommandation (ex : recommandations produits)

Étape 2 : Bases de données

A. Concepts essentiels

- **Modèle relationnel (SQL)** : organisation des données en tables avec relations
- **NoSQL** : bases non relationnelles, adaptées aux données non structurées
    - Types : document (MongoDB), clé-valeur (Redis), colonne (Cassandra)

B. Maîtrise de SQL

- Requêtes fondamentales :
    - `SELECT`, `INSERT`, `UPDATE`, `DELETE`
    - Clauses WHERE, ORDER BY, GROUP BY
    - Jointures (INNER, LEFT, RIGHT, FULL)
    - Index : optimisation des requêtes
- Spécificités des SGBD :
    - Oracle : PL/SQL, outils comme SQL*Plus
    - MS SQL Server : T-SQL, SQL Server Management Studio (SSMS)

C. Bases NoSQL

- **MongoDB** : documents JSON, requêtes basiques, agrégation
- **Cassandra** : architecture distribuée, cas d'usage pour grandes données

---

Projets simples à réaliser

- Nettoyer et analyser un jeu de données CSV avec pandas
- Implémenter un algorithme ML simple (ex : classification de fleurs Iris)
- Construire une petite base SQL, faire des requêtes avancées
- Créer une base MongoDB, insérer et interroger des documents

6. Gestion de projet & IT

Étape 1 : Certifications & méthodes

A. PMP (Project Management Professional)

- Principes fondamentaux de gestion de projet
- Les phases du projet : initiation, planification, exécution, suivi/contrôle, clôture
- Livrables clés à chaque phase (charte de projet, plan de projet, rapports d'avancement)

B. ITIL (Information Technology Infrastructure Library)

- Bonnes pratiques pour la gestion des services IT
- Processus ITIL principaux : gestion des incidents, gestion des problèmes, gestion des changements
- Importance de la qualité du service et de la satisfaction utilisateur

C. Outils de pilotage

- **MS Project** : planification de projet, création de tâches, assignation, suivi
- Diagramme de **Gantt** : visualisation temporelle des tâches
- Diagramme de **PERT** : analyse des dépendances et temps critiques

---

Étape 2 : Projets & stages

A. Types de projets

- **PFA (Projet de fin d'année)** : projet court, souvent en équipe, démontrant l'application des connaissances
- **PFE (Projet de fin d'études)** : projet plus complet et approfondi, souvent individuel ou encadré, orienté recherche ou développement
- **Stages d'été** : immersion professionnelle, objectifs de découverte, apprentissage et contribution

B. Organisation des projets

- Phases classiques :
  - Analyse des besoins
  - Conception (modélisation, choix techniques)
  - Réalisation (développement)
  - Tests (unitaires, intégration, validation)
  - Documentation (manuel utilisateur, rapports)

C. Valorisation des expériences

- Rédiger un CV clair, axé sur les réalisations
- Préparer des réponses pour les entretiens (méthode STAR : Situation, Tâche, Action, Résultat)
- Construire et entretenir un réseau professionnel (LinkedIn, événements, alumni)

Étape 3 : Soft & Digital Skills

A. Communication efficace

- Techniques d'expression orale : clarté, articulation, gestion du stress
- Rédaction professionnelle : e-mails, rapports, synthèses

B. Entrepreneuriat & esprit d'initiative

- Comprendre les bases de la création d'entreprise
- Développer un mindset proactif et innovant

C. Employabilité

- Techniques de recherche d'emploi : réseaux, candidatures, suivi
- Préparation aux entretiens : questions fréquentes, mise en situation

D. Management de projet agile & leadership

- Rôles et responsabilités d'un leader agile
- Facilitation d'équipe, résolution de conflits, motivation

Projets simples à réaliser

- Planifier un mini-projet avec MS Project (créer tâches, dépendances, échéances)
- Simuler un entretien d’embauche avec un ami ou mentor
- Participer à un atelier de communication orale
- Monter un petit projet en mode agile avec une équipe