Introduction à la programmation

Durée: 2 Jours Réf de cours: GK840203

Résumé:

Apprenez les bases de la programmation et lancez-vous dans un parcours de développeur.

Le cours « Introduction à la programmation » est idéal pour tous ceux qui souhaitent s'initier au codage. Que vous souhaitiez vous lancer dans une carrière dans le développement logiciel, améliorer vos compétences en résolution de problèmes ou simplement automatiser des tâches quotidiennes, ce cours est fait pour vous.
Vous vous lancerez dans les bases de la programmation en découvrant les types de données, les variables, les opérateurs et le contrôle de flux.
Grâce à un mélange de cours magistraux, d'exercices et de travaux pratiques, vous vous construirez une base solide en programmation. Vous découvrirez également les différences entre les langages compilés et interprétés, et vous vous familiariserez avec les techniques de débogage qui vous aideront à identifier et à corriger les erreurs dans votre code.
À la fin du cours, vous serez capable d'écrire des programmes simples et de comprendre les concepts fondamentaux qui sous-tendent tous les langages de programmation et vous disposerez des compétences et de la confiance nécessaires pour aborder des sujets et des projets de programmation plus avancés.

Mis à jour 17/11/2026

Public visé:

Ce cours est conçu pour être accessible aux débutants, aucune expérience préalable en programmation n'est donc requise.

Objectifs pédagogiques:

- A l'issue de la formation, les participants seront capables de :
- Comprendre l'objectif et l'importance de la programmation dans la résolution de problèmes concrets et l'automatisation de tâches.
- Distinguer les langages de programmation compilés des langages interprétés et identifier leurs cas d'utilisation courants.
- Utiliser les concepts fondamentaux de la programmation, notamment les types de données, les variables, les opérateurs et les expressions, pour écrire des programmes simples.
- Créer des variables pour stocker et gérer différents types de données dans un programme.
- Appliquer des opérateurs arithmétiques, de comparaison et logiques pour effectuer des opérations et évaluer des expressions.
- Construire des instructions conditionnelles, notamment « if », « else if », « else » et « switch », pour gérer la prise de décision dans les programmes.
- Créer des programmes intégrant des conditions imbriquées et enchaînées afin de résoudre des problèmes plus complexes.
- Utiliser des techniques de débogage pour identifier et corriger les erreurs logiques dans des programmes simples.

Pré-requis:

Ce cours est accessible aux débutants, toutefois il est préférable d'avoir une appétence technique. Des connaissances de base en mathématiques et en algorithmes peuvent être utiles.

Test et certification

Après cette formation, nous vous conseillons le(s) module(s) suivant(s):

Pas de recommandation spécifique, dépend du projet professionnel de chaque participant.

Contenu:

Introduction

- Définition et objectif de la programmation
- Qu'est-ce que la programmation et à quoi sert-elle ?
- Comment la programmation permet-elle de résoudre des problèmes concrets ?
- Évolution et bref historique

Langages de programmation et configuration de base de l'environnement

- Que sont les langages de programmation ?
- Définition et objectif des langages de programmation
- Types de langages de programmation : procéduraux, orientés objet, de script, fonctionnels
- Langages compilés vs langages interprétés (de haut niveau)
- Brève présentation des langages courants (Python, JavaScript, C++, etc.)
- Configuration de l'environnement
- Choix d'un éditeur de code (par ex. VS Code, Sublime, environnements de test en ligne)
- Brève présentation de l'installation du runtime ou de l'interpréteur requis pour un langage de programmation (par ex. Python, Node.js, Java)
- Création de fichiers de code et enregistrement avec les extensions appropriées
- Exécution du code depuis la ligne de commande ou depuis des terminaux intégrés
- Démonstration simple de la création et de l'exécution d'un fichier générique « Hello World »
- Structure du code
- Éléments communs d'un programme
- Langages de programmation : syntaxe (règles) et sémantique (signification)
- Qu'est-ce que la syntaxe ?
- Importance de l'indentation
- Commentaires dans le code
- Structure générale d'un programme simple (pseudocode)

Types de données et variables

- Principes fondamentaux des types de données
- Définition et objectif
- Types de données primitifs (nombres, chaînes de caractères, booléens)
- Types de données non primitifs ou composés (aperçu général, par ex. : tableaux/objets)
- Concept de mémoire : comment les données sont stockées (conceptuel)
- Déclaration et initialisation des variables
- Que sont les variables ?

Flux de contrôle (conditions)

- Que sont les conditions ?
- Explication
- Comment les programmes utilisent les conditions pour répondre à différentes situations
- Instructions if/else
- Structure générale (if, else if, else)
- Quand imbriquer et quand enchaîner les « else if »
- Considérations de complexité et organisation du code
- Switch/Case (conceptuel)
- Différences par rapport à « if/else »
- Cas d'utilisation typiques (plusieurs cas distincts)
- Instructions « break » et « default » (comportement et nécessité)

Fonctions

- Introduction aux fonctions
- Que sont les fonctions ?
- Objectif des fonctions (réutilisabilité, organisation du code)
- Comment les fonctions résolvent des problèmes en les décomposant en tâches plus petites.
- Définitions : paramètres vs arguments
- Valeurs de retour vs effets secondaires
- Syntaxe des fonctions
- Exemple de déclaration de fonction en pseudocode (paramètres, corps, instruction de retour)
- Appel de fonctions et passage d'arguments
- Bonnes pratiques en matière de nommage et de portée des fonctions (concept de portée locale vs globale)
- Concept de « champ d'application » dans un programme
- Différence entre les champs d'application

Boucles et itération

- Que sont les boucles ?
- Définition et objectif
- Pourquoi utilise-t-on des boucles (par exemple, pour traiter des ensembles de données, automatiser des tâches répétitives) ?
- Types de boucles (conceptuels)
- Boucles « while » et « do-while » (différences dans la vérification des conditions)
- Boucles « FOR » (itération basée sur un compteur)
- Boucles imbriquées (quand les utiliser et quand les éviter pour limiter la complexité)
- Boucle « WHILE »
- Boucle « DO-WHILE »

Débogage - Principes fondamentaux

- Types d'erreurs (syntaxe, exécution, logique)
- Lecture des messages d'erreur

Utilisation des instructions print/log pour le débogage

- Intérêt du stockage des données dans des variables
- Mots-clés courants de déclaration (par ex. : var, let, const ou équivalents dans d'autres langages)
- Conventions de nommage et bonnes pratiques
- Utilisation des variables
- Mise à jour et réaffectation des valeurs
- Combinaison de variables dans des expressions (par ex : concaténation de chaînes de caractères ou calculs numériques)
- Exemples simples d'utilisation des variables dans un petit extrait de code (pseudocode)

Opérateurs et expressions

- Que sont les opérateurs ?
- Définition et utilité
- Types d'opérateurs
- Opérateurs arithmétiques
- Addition, soustraction, multiplication, division, modulo
- Priorité des opérateurs (ordre des opérations)
- Opérateurs de comparaison
- Supérieur à, inférieur à, égalité vs égalité stricte (différences conceptuelles)
- Opérateurs logiques (AND, OR, NOT)
- Concepts de « vrai » et « faux » au sens général
- Expressions et évaluations
- Combiner des variables et des opérateurs pour former des expressions
- Évaluer une expression pour obtenir un résultat
- Exemples de scénarios (calculs, comparaisons, etc.)

- Boucle inversée
- Contrôle des boucles
- Instructions « break » et « continue » (niveau avancé)
- Boucles infinies (ce qu'elles sont et comment les éviter)

Structures de données de base

- Que sont les structures de données ?
- Définition et objectif
- Pourquoi sont-elles importantes en programmation ?
- Tableaux/listes
- Stockage de plusieurs éléments dans une seule structure
- Tableaux/listes avec des types de données uniques
- Tableaux/listes avec plusieurs types de données
- Indexation et accès aux éléments (conceptuel)

Méthodes pédagogiques :

Un support de cours électronique est remis à chaque participant

Autres moyens pédagogiques et de suivi:

• Compétence du formateur : Les experts qui animent la formation sont des spécialistes des matières abordées et ont au minimum cinq ans d'expérience d'animation. Nos équipes ont validé à la fois leurs connaissances techniques (certifications le cas échéant) ainsi que leur compétence pédagogique.
• Suivi d'exécution : Une feuille d'émargement par demi-journée de présence est signée par tous les participants et le formateur.
• En fin de formation, le participant est invité à s'auto-évaluer sur l'atteinte des objectifs énoncés, et à répondre à un questionnaire de satisfaction qui sera ensuite étudié par nos équipes pédagogiques en vue de maintenir et d'améliorer la qualité de nos prestations.

Délais d'inscription :

• Vous pouvez vous inscrire sur l'une de nos sessions planifiées en inter-entreprises jusqu'à 5 jours ouvrés avant le début de la formation sous réserve de disponibilité de places et de labs le cas échéant.
• Votre place sera confirmée à la réception d'un devis ou """"booking form""""" signé. Vous recevrez ensuite la convocation et les modalités d'accès en présentiel ou distanciel.
• Attention, si cette formation est éligible au Compte Personnel de Formation, vous devrez respecter un délai minimum et non négociable fixé à 11 jours ouvrés avant le début de la session pour vous inscrire via moncompteformation.gouv.fr.

Accueil des bénéficiaires :

• En cas de handicap : plus d'info sur globalknowledge.fr/handicap
• Le Règlement intérieur est disponible sur globalknowledge.fr/reglement