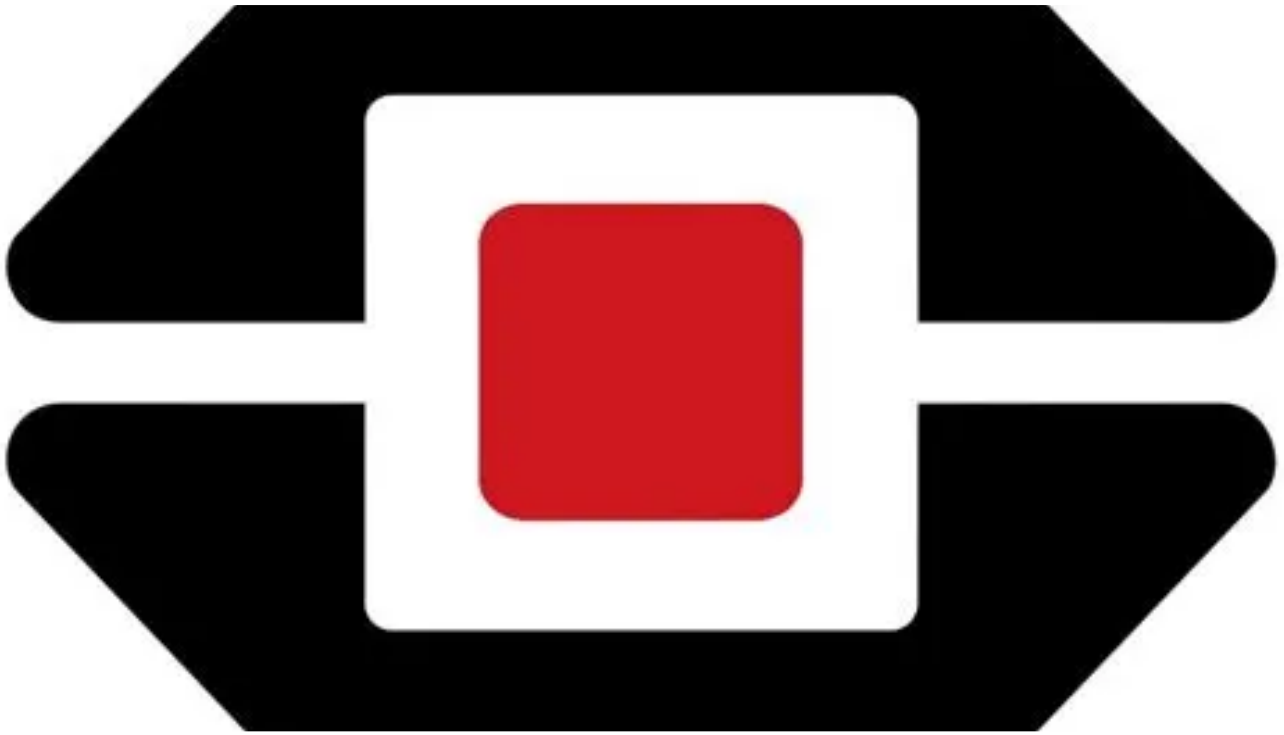




Iàlpeàlr

Introduction à la programmation et à la robotique

Sylvain Léonis

12 chapitres

SUPPORTS DE COURS

<https://cours.tsix.be/cours/introduction-a-la-programmation-et-a-la-robotique>

Version 1 · 12/09/2026

PRÉSENTATION

Programme du cours

Présentation

Notre programme de cours d'introduction à la robotique et à la programmation, utilisant les LEGO Mindstorms, est une initiative éducative innovante conçue pour offrir aux étudiants une expérience d'apprentissage immersive et pratique dans le domaine en constante évolution de la technologie.

Ce cours vise à fournir aux étudiants une introduction complète aux principes fondamentaux de la robotique et de la programmation à travers l'utilisation de LEGO Mindstorms, une plateforme pédagogique reconnue pour son potentiel éducatif et son adaptabilité aux différents niveaux d'apprentissage.

Objectifs pédagogiques

- **Initiation à la robotique** : Les étudiants exploreront les bases de la construction de robots en assemblant des modèles avec les kits LEGO Mindstorms. Ils comprendront les principes de la mécanique, de l'ingénierie et de l'électronique nécessaires à la création de ces robots.
- **Introduction à la programmation** : Les participants apprendront les bases de la programmation en utilisant une interface conviviale pour contrôler leurs robots. Ils acquerront des compétences en codage, en logique et en résolution de problèmes pour donner des instructions aux robots et les faire fonctionner.
- **Développement de compétences pratiques et transversales** : Outre les aspects techniques, ce cours encourage le développement de compétences transférables telles que le travail d'équipe, la pensée critique, la créativité et la résolution de problèmes.
- **Expérience pratique et projets concrets** : Les étudiants mettront en pratique leurs connaissances en participant à des projets concrets. Ils seront amenés à concevoir, construire et programmer des robots pour résoudre des défis spécifiques, favorisant ainsi l'application directe de leurs compétences.

Programme

1. Introduction à la Robotique et à LEGO Mindstorms
 - Présentation du cours, des objectifs et de la plateforme LEGO Mindstorms
 - Assemblage du premier robot simple avec LEGO Mindstorms
2. Fondements de la Programmation

- Introduction aux concepts de programmation (séquences, boucles, conditions)
- Utilisation de l'interface graphique LEGO Mindstorms pour la programmation
- Création de programmes pour le déplacement du robot (avancer, reculer, tourner)

3. Capteurs

- Exploration des différents capteurs (capteur de couleur, capteur de toucher, capteur ultrasonique)
- Approfondissement des concepts de condition et de boucles en programmation
- Formalisation du concept de variable
- Apprentissage de l'utilisation des capteurs pour la détection d'obstacles et la prise de décisions par le robot

4. Projets Pratiques

- Réalisation de projets pratiques guidés, impliquant la résolution de défis spécifiques à l'aide des robots programmés
- Exemples de projets : suivre une ligne, éviter des obstacles, ramasser des objets, etc.

5. Projet Final et Présentations

- Les étudiants travaillent en équipes pour concevoir, construire et programmer un projet final avec LEGO Mindstorms
- Concours final en équipe, visant à concevoir un robot capable d'effectuer un maximum de tâches dans un temps donné

6. Evaluation

- Les étudiants seront évalués sur les résultats de leur robot lors du concours final (en dehors du classement du concours)
- Examen oral individuel visant à objectiver les niveaux de compréhension de chacun en dehors de son intégration dans l'équipe

Sommaire

- Programme du cours
- 1** Introduction
 - 1.1** Présentation
 - 1.2** Robotique
 - 1.3** Programmation
- 2** LEGO Mindstorms
 - 2.1** Introduction
 - 2.2** Environnement
 - 2.3** Programmation
- 3** Programmation
 - 3.1** Algorithme
 - 3.2** Langage
 - 3.3** Syntaxe

CHAPITRE 1

Chapitre 1 : Introduction

Présentation du cours, des objectifs et de la plateforme *LEGO Mindstorms*

Dans cette section, nous introduisons le cours, en précisant ses objectifs pédagogiques et en présentant la plateforme *LEGO Mindstorms*. Nous aborderons l'importance de la robotique dans l'éducation et la vie quotidienne et comment les *LEGO Mindstorms* permettent une approche pratique et interactive de l'apprentissage.

La robotique

La robotique et la domotique représentent des secteurs innovants qui transforment nos interactions quotidiennes avec la technologie. La robotique applique l'automatisation à des tâches variées, améliorant l'efficacité dans de nombreux domaines, tandis que la domotique fusionne confort et technologie pour créer des espaces de vie intelligents. Ce chapitre vous immergera dans ces univers fascinants, posant les bases pour comprendre comment ces technologies évoluent et s'intègrent dans notre quotidien.

La programmation

La programmation est l'art de créer des instructions que les ordinateurs (et les robots) peuvent suivre pour effectuer des tâches spécifiques. C'est le fondement qui permet à toute technologie informatique de fonctionner, des applications mobiles simples aux systèmes complexes de l'intelligence artificielle. Dans ce chapitre, nous allons explorer ce monde et ses applications dans le cadre de la robotique et de la domotique, mais aussi dans d'autres cas de la vie quotidienne et professionnelle.

Assemblage du premier robot simple avec *LEGO Mindstorms*

Nous guiderons les étudiants à travers les étapes de l'assemblage de leur premier robot en utilisant le kit *LEGO Mindstorms*. Cette section comprendra des instructions détaillées, des conseils pratiques et des illustrations pour faciliter la compréhension du processus d'assemblage.

1.1 Présentation

Présentation à venir prochainement.

1.2 Robotique et Domotique : Vers une Synergie Technologique

La robotique et la domotique sont deux branches de la technologie qui, bien qu'ayant des applications différentes, partagent un objectif commun : l'automatisation des tâches pour améliorer la qualité de vie et l'efficacité des processus. Ce chapitre explore comment ces deux domaines s'entremêlent et façonnent l'avenir technologique.

La Robotique : Définition et Domaines d'Application

La robotique est un champ multidisciplinaire qui englobe la conception, la fabrication, et le contrôle de robots. Ces machines autonomes ou semi-autonomes sont capables de réaliser des tâches variées sans intervention humaine directe.

La robotique s'est infiltrée dans presque tous les secteurs d'activité, révolutionnant les méthodes de travail et les services. Examinons de plus près son impact dans quatre domaines clés.

Industrie Manufacturière

Dans le secteur manufacturier, les robots sont devenus indispensables. Ils exécutent avec une précision et une constance inégalées des tâches qui seraient monotones ou dangereuses pour les humains. Les bras robotiques, par exemple, peuvent souder, peindre, assembler, et manipuler des matériaux lourds ou toxiques. Ils contribuent ainsi à :

- **Réduire les Coûts de Production** : Moins d'erreurs et de déchets grâce à une précision accrue.
- **Augmenter la Production** : Fonctionnement continu sans la nécessité de pauses ou de relais.
- **Améliorer les Conditions de Travail** : Réduction des risques d'accidents et des tâches pénibles pour les ouvriers.

Recherche Spatiale

Les robots jouent un rôle crucial dans l'exploration spatiale. Des rovers comme Curiosity sur Mars recueillent des données sur la géologie, l'atmosphère, et la possibilité de vie sur d'autres planètes. Les robots spatiaux permettent :

- **D'explorer des Environnements Hostiles** : Accès à des zones inaccessibles ou dangereuses pour les astronautes.
- **De Réaliser des Expériences Scientifiques** : Conduite d'expériences complexes sans intervention humaine directe.
- **De Collecter des Échantillons** : Prélèvement de roches et de sol pour analyse.

Médecine

La robotique médicale transforme la chirurgie et le soin des patients. Des systèmes comme le robot Da Vinci permettent aux chirurgiens de réaliser des opérations complexes à travers de petites incisions, ce qui réduit les risques d'infection et accélère la convalescence. Les robots médicaux contribuent à :

- **Améliorer la Précision des Interventions** : Mouvements plus précis et stables que la main humaine.
- **Minimiser l'Invasivité des Procédures** : Moins de douleur et de cicatrices pour les patients.
- **Faciliter la Rééducation** : Robots d'assistance pour aider les patients à retrouver leur mobilité.

Services

Les robots de service automatisent des tâches de la vie quotidienne et des services au public. On les trouve dans les hôtels, les aéroports, et même dans certains restaurants. Ils peuvent :

- **Accueillir et Orienter les Clients** : Robots réceptionnistes fournissant informations et direction.
- **Nettoyer et Entretenir des Espaces** : Robots de nettoyage pour les sols, les vitres, et les grands espaces publics.
- **Livrer des Produits** : Drones et robots de livraison qui transportent des colis ou de la nourriture.

En somme, la robotique améliore l'efficacité et la sécurité dans de nombreux secteurs, tout en ouvrant la voie à des innovations qui pourraient transformer radicalement notre manière de vivre et de travailler dans le futur.

Domotique : L'Intégration Technologique au Service du Foyer

La domotique représente l'ensemble des technologies et des dispositifs intégrés au sein de l'habitat pour en automatiser les systèmes et améliorer la qualité de vie des résidents. Elle englobe divers aspects, dont l'éclairage et le chauffage, la sécurité, ainsi que la gestion des appareils ménagers. Explorons ces éléments en détail.

Éclairage et Chauffage

La gestion automatisée de l'éclairage et du chauffage est l'une des fonctionnalités les plus appréciées de la domotique. Elle permet non seulement d'améliorer le confort mais aussi de réaliser des économies d'énergie significatives.

- **Programmation Temporelle** : Les systèmes d'éclairage peuvent être programmés pour s'allumer et s'éteindre à des moments précis de la journée, adaptant l'ambiance lumineuse aux besoins des occupants.
- **Détecteurs de Présence** : L'éclairage peut également être contrôlé par des capteurs de mouvement, s'allumant uniquement en présence d'individus dans une pièce.

- **Thermostats Intelligents** : Pour le chauffage, les thermostats connectés ajustent la température en fonction des habitudes de vie, de la météo extérieure, ou même de la présence effective des personnes dans la maison.

Sécurité

La sécurité est un autre pilier fondamental de la domotique. Les technologies modernes permettent de protéger la maison contre les intrusions, les accidents et les urgences de manière bien plus efficace.

- **Systèmes d'Alarme Connectés** : Ils peuvent informer les résidents ou les services de sécurité en cas d'effraction grâce à des notifications sur smartphone.
- **Caméras de Surveillance** : Elles offrent la possibilité de surveiller à distance ce qui se passe dans et autour de la maison, avec des enregistrements accessibles via internet.
- **Contrôles d'Accès** : Les serrures intelligentes et les portiers vidéo permettent de gérer l'accès à la maison, même à distance, augmentant ainsi la sécurité des occupants.

Appareils Ménagers

Les appareils ménagers connectés transforment la manière dont les tâches domestiques sont effectuées, en apportant une couche d'intelligence et d'interactivité.

- **Réfrigérateurs Intelligents** : Capables de surveiller leur contenu, de suggérer des recettes en fonction des aliments disponibles et même de passer commande pour le réapprovisionnement.
- **Lave-linge et Lave-vaisselle Connectés** : Ils peuvent être programmés pour fonctionner aux heures où l'électricité est moins chère ou lorsque les occupants sont absents, et peuvent alerter les utilisateurs en cas de cycle terminé ou de problème.
- **Systèmes de Cuisine Intégrés** : Ils assistent dans la préparation des repas, avec des fonctions de cuisson précises et des recommandations pour des cuissons optimales.

La domotique, en intégrant ces divers aspects, rend la maison non seulement plus confortable et sécurisée, mais aussi plus réactive et adaptative aux besoins et habitudes de ses occupants. Elle représente une étape importante vers l'habitat du futur, où l'efficacité énergétique et le confort de vie sont en parfaite harmonie grâce à l'intelligence artificielle et à l'Internet des Objets (IoT).

Interfaces Intuitives

Les interfaces utilisateurs sont devenues plus intuitives, permettant une interaction plus naturelle avec la technologie domestique.

- **Commandes Vocales** : Des assistants vocaux comme Alexa ou Google Assistant permettent de contrôler les appareils domestiques par la voix, rendant leur utilisation plus accessible et pratique.

- **Gestes** : Certains systèmes domotiques peuvent reconnaître des gestes spécifiques pour activer ou désactiver des appareils, ou pour régler des paramètres comme l'intensité lumineuse ou le volume sonore.

Connectivité et Internet des Objets (IoT)

La connectivité est au cœur de la domotique moderne, permettant une gestion centralisée et à distance des appareils domestiques.

- **Synchronisation** : Les appareils peuvent communiquer entre eux pour créer des scénarios automatisés. Par exemple, le système d'alarme peut éteindre tous les appareils électriques lorsqu'il est armé.
- **Contrôle à Distance** : Grâce à l'IoT, il est possible de surveiller et de contrôler les systèmes domestiques via des applications sur smartphone ou tablette, que ce soit pour ajuster le thermostat ou vérifier les caméras de sécurité.

Convergence et Synergies

La robotique et la domotique, bien que distinctes, s'entrelacent de plus en plus grâce aux progrès technologiques. Cette convergence est à l'origine de plusieurs innovations qui transforment notre quotidien.

Robots Domestiques

Les robots domestiques sont peut-être l'exemple le plus évident de la fusion entre la robotique et la domotique. Ces appareils autonomes accomplissent des tâches ménagères, réduisant ainsi la charge de travail domestique.

- **Aspirateurs Robots** : Ils cartographient l'espace de vie et naviguent de manière autonome pour nettoyer les sols, s'adaptant aux différents types de surface et évitant les obstacles.
- **Tondeuses Robotisées** : Elles entretiennent les pelouses de manière autonome, en suivant des schémas de tonte prédéfinis et en retournant se charger lorsqu'il le faut.
- **Aides Domestiques Automatisés** : Des robots de plus en plus sophistiqués peuvent désormais accomplir des tâches plus complexes, comme le rangement, la désinfection de surfaces ou l'assistance aux personnes à mobilité réduite.

Domotique Industrielle

La domotique industrielle, également connue sous le nom d'automatisation industrielle, est l'application des principes de la domotique à l'environnement industriel et à ses nombreux robots. Elle vise à optimiser les processus de production, à améliorer la sécurité et à augmenter l'efficacité énergétique des installations industrielles. Voici quelques-uns de ses aspects :

- **Gestion Centralisée des Systèmes** : Des plateformes intégrées permettent de contrôler et de surveiller à distance les machines, les flux de production et l'énergie utilisée dans une usine.
- **Maintenance Prédictive** : Grâce à l'analyse des données collectées par des capteurs intelligents, il est possible de prévoir les défaillances d'équipement avant qu'elles ne surviennent, réduisant ainsi les temps d'arrêt et les coûts de maintenance.
- **Optimisation Énergétique** : Les systèmes d'automatisation peuvent ajuster le fonctionnement des machines en fonction des besoins réels, minimisant la consommation d'énergie et réduisant l'empreinte carbone de l'industrie.

La domotique industrielle s'appuie sur des technologies telles que l'Internet des Objets, l'intelligence artificielle et le Big Data pour créer des usines intelligentes où les processus sont interconnectés, automatisés et optimisés. Cela permet non seulement une production plus agile et personnalisée, mais aussi une meilleure réactivité aux conditions changeantes du marché et une intégration plus poussée de la chaîne d'approvisionnement.

Impact sur la Société et l'Individu

La robotique et la domotique transforment notre société et ont un impact significatif sur la vie individuelle. Ces technologies redéfinissent les interactions quotidiennes avec notre environnement et apportent des changements bénéfiques sur plusieurs plans.

Amélioration de la Qualité de Vie

Les technologies robotiques et domotiques simplifient les tâches ménagères et professionnelles, ce qui se traduit par plusieurs avantages :

- **Gain de Temps** : Les robots domestiques comme les aspirateurs et les tondeuses prennent en charge les corvées, libérant du temps pour des activités plus enrichissantes.
- **Confort Accru** : La domotique permet de personnaliser l'environnement domestique selon les préférences des utilisateurs, avec des réglages automatiques pour l'éclairage, la température et les appareils multimédia.
- **Réduction du Stress** : La gestion automatisée des tâches diminue la charge mentale associée à la planification et à l'exécution des activités domestiques.

Accessibilité et Assistance

La robotique et la domotique offrent des solutions d'assistance qui augmentent l'autonomie des personnes à mobilité réduite ou des seniors :

- **Aides à la Mobilité** : Les fauteuils roulants robotisés et les exosquelettes assistent les déplacements et les efforts physiques.
- **Systèmes d'Alerte Médicale** : Des dispositifs connectés peuvent détecter les chutes ou les urgences médicales et alerter les services d'urgence ou les proches.

- **Assistance Quotidienne** : Des robots peuvent aider à la prise de médicaments, à la préparation des repas ou même offrir une compagnie sociale pour réduire l'isolement.

Développement Durable

L'automatisation contribue également à un mode de vie plus durable en optimisant l'utilisation des ressources :

- **Gestion Énergétique Intelligente** : Les systèmes domotiques réduisent la consommation d'énergie en adaptant le chauffage, l'éclairage et l'utilisation des appareils électriques aux besoins réels.
- **Minimisation des Déchets** : La robotique industrielle améliore l'efficacité des processus de production, réduisant ainsi les déchets et la surproduction.
- **Agriculture de Précision** : Les technologies robotiques permettent une agriculture plus précise et moins gourmande en ressources, avec une gestion optimisée de l'eau, des engrais et des pesticides.

Au-delà des avantages pratiques, la fusion de la robotique et de la domotique soulève des questions éthiques et sociales, telles que la protection de la vie privée, la sécurité des données et l'impact sur l'emploi. Il est impératif que ces technologies soient développées et intégrées de manière responsable pour maximiser leurs bénéfices tout en minimisant les risques potentiels.

Défis et Perspectives

L'adoption croissante de la robotique et de la domotique représente une avancée majeure vers des environnements de vie et de travail plus intelligents. Cependant, cette intégration n'est pas exempte de défis qui doivent être relevés pour assurer une transition harmonieuse et équitable.

Sécurité et Vie Privée

Avec l'augmentation des appareils connectés, la sécurité des données et la vie privée deviennent des préoccupations majeures.

- **Protection des Données** : Les données personnelles collectées par les appareils domestiques et industriels doivent être protégées contre les accès non autorisés et les fuites.
- **Cybersécurité** : Les systèmes doivent être sécurisés contre les cyberattaques qui pourraient prendre le contrôle des appareils et compromettre la sécurité des utilisateurs.
- **Normes et Réglementations** : Le développement et l'application de normes de sécurité strictes sont essentiels pour protéger les consommateurs et leurs informations. Les lois doivent également intervenir pour empêcher les abus, tel que le scandale de Cambridge Analytica, par exemple.

Accessibilité Économique

L'accessibilité économique est un enjeu clé pour que les bénéfices de ces technologies puissent être partagés par tous.

- **Coût des Technologies** : Les coûts initiaux élevés des systèmes robotiques et domotiques peuvent constituer une barrière à l'entrée pour les ménages à faible et moyen revenu.
- **Subventions et Aides** : Des programmes d'aide pourraient être mis en place pour rendre ces technologies plus accessibles, en particulier pour les personnes qui pourraient en bénéficier le plus, comme les seniors et les personnes handicapées.
- **Modèles Économiques Innovants** : Le développement de solutions abordables et de modèles de financement alternatifs pourrait aider à démocratiser l'accès à la domotique et à la robotique.

Éthique et Emploi

L'impact de l'automatisation sur le marché du travail et les considérations éthiques associées sont des sujets de débat important.

- **Remplacement de la Main-d'œuvre** : L'automatisation peut entraîner une réduction des emplois, en particulier dans les secteurs manufacturier et de service.
- **Formation et Reconversion** : Des programmes de formation et de reconversion professionnelle sont nécessaires pour aider les travailleurs à s'adapter aux nouvelles réalités du marché du travail.
- **Questions Éthiques** : Les implications morales de l'utilisation de robots, par exemple dans les soins aux personnes âgées ou dans les décisions liées à la sécurité, doivent être soigneusement considérées.

En conclusion, si les défis associés à la robotique et à la domotique sont importants, les perspectives qu'elles offrent en termes d'innovation et d'amélioration de la qualité de vie sont considérables. Une approche équilibrée qui prend en compte les aspects techniques, sociaux et éthiques est essentielle pour réaliser pleinement le potentiel de ces technologies.

Conclusion

La robotique et la domotique, en pleine expansion, redéfinissent notre interaction avec la technologie. Elles promettent un avenir où les machines nous assistent de manière toujours plus intégrée et naturelle, tout en soulevant des questions importantes sur leur intégration harmonieuse dans la société.

1.3 Programmation : la communication entre l'homme et sa machine

Concepts de Base de la Programmation

La programmation informatique est le processus par lequel on écrit, teste, débogue et maintient le code source des programmes informatiques. Ce code source est écrit dans un langage de programmation qui est compréhensible et exécutable par un ordinateur. Il existe de nombreux langages de programmation, chacun avec ses propres usages et domaines d'application.

Langages et Environnements de Développement

Certains langages sont particulièrement adaptés à la robotique et à la domotique, comme Python pour sa simplicité et sa polyvalence, ou C/C++ pour sa performance et son contrôle sur le matériel. Les environnements de développement intégrés (IDE) facilitent l'écriture et le test du code, et des plateformes comme Arduino ou Raspberry Pi offrent des environnements matériels et logiciels propices à l'apprentissage et à la mise en œuvre de projets en robotique et domotique.

Algorithmique

L'algorithmique est l'étude et la conception des algorithmes, qui sont des ensembles d'instructions pour résoudre des problèmes ou effectuer des tâches. En robotique et domotique, les algorithmes peuvent être utilisés pour le contrôle de mouvement, la navigation, la perception de l'environnement, ou encore la gestion intelligente de l'énergie dans une maison.

Logique de Programmation

La logique de programmation est fondamentale, car elle permet de structurer le code en séquences d'instructions, de décisions (if/else), et de boucles (for/while). Ces structures de contrôle sont essentielles pour créer des programmes qui peuvent réagir aux entrées sensorielles et prendre des décisions autonomes.

Conclusion

La programmation est un outil puissant qui permet de transformer des idées en solutions concrètes applicable par un ordinateur. Elle ouvre la porte à des maisons plus intelligentes et à des robots plus autonomes, capables d'assister les humains dans leurs tâches quotidiennes et professionnelles. En maîtrisant les bases de la programmation, on peut non seulement comprendre mais aussi participer à la création de ces technologies innovantes, les personnaliser à ses besoins ou encore concevoir des solutions encore inexistantes.

CHAPITRE 2

Programmation de LEGO Mindstorms

Introduction aux LEGO Mindstorms

Composants du kit Mindstorms

Découverte des différents éléments qui composent le kit Mindstorms : briques intelligentes, moteurs, capteurs, et connectiques.

Prise en main et configuration

Instructions pour la mise en route d'un kit Mindstorms, incluant le montage de base et la configuration logicielle.

Environnement de développement

Logiciel de programmation Mindstorms

Présentation du logiciel officiel LEGO pour la programmation des robots Mindstorms.

Interface et fonctionnalités

Exploration de l'interface utilisateur du logiciel de programmation Mindstorms et des différentes fonctionnalités offertes.

Premiers pas en programmation

Création d'un premier programme simple pour comprendre les bases de la programmation des robots Mindstorms.

Programmation de base

Blocs de programmation

Introduction aux blocs de programmation de base et à leur utilisation pour contrôler les robots Mindstorms.

Structures de contrôle

Apprentissage de l'utilisation des structures de contrôle telles que les boucles et les instructions conditionnelles dans le contexte des robots Mindstorms.

Capteurs

Utilisation des capteurs LEGO Mindstorms pour recevoir des informations de l'environnement et réagir en conséquence.

Projets avancés

Programmation des moteurs

Techniques avancées pour la programmation des moteurs, incluant la synchronisation et le contrôle précis de la vitesse et de la direction.

Gestion des capteurs avancés

Approfondissement de l'utilisation des capteurs et traitement des données pour des tâches plus complexes.

Conception de projets personnalisés

Planification et conception

Méthodologies pour la planification et la conception de projets robotiques personnalisés avec LEGO Mindstorms.

Débogage et optimisation

Stratégies pour identifier et corriger les problèmes dans les programmes Mindstorms, ainsi que pour optimiser les performances du robot.

Documentation et présentation de projets

Conseils pour documenter efficacement les projets Mindstorms et pour préparer des présentations de ces projets.

Communauté et ressources

Ressources en ligne et support

Présentation des ressources en ligne disponibles, y compris les forums, les tutoriels et la documentation officielle.

Compétitions et événements

Information sur les compétitions de robotique LEGO Mindstorms et les événements communautaires où les participants peuvent présenter leurs créations.

2.1 Introduction aux LEGO Mindstorms

Composants du kit Mindstorms



Brique intelligente EV3



La brique intelligente EV3 est l'élément central des créations LEGO Mindstorms. Elle agit comme le cerveau du robot, lui permettant de traiter les informations, de contrôler les moteurs et de réagir aux capteurs. Nous allons explorer ses caractéristiques techniques et ses fonctionnalités.

Caractéristiques techniques

- **Processeur** : le processeur ARM9 est le moteur de la brique EV3, lui fournissant la puissance nécessaire pour exécuter des programmes complexes et gérer plusieurs tâches simultanément.
- **Mémoire** : avec une mémoire interne suffisante pour stocker des programmes et des données, la brique EV3 peut gérer des projets de toutes tailles. Elle est également extensible via une carte SD.
- **Ports d'entrée et de sortie** : la brique dispose de plusieurs ports d'entrée et de sortie qui permettent de connecter jusqu'à quatre capteurs et quatre moteurs, ouvrant un large éventail de possibilités pour la construction de robots.

Fonctionnalités de la brique

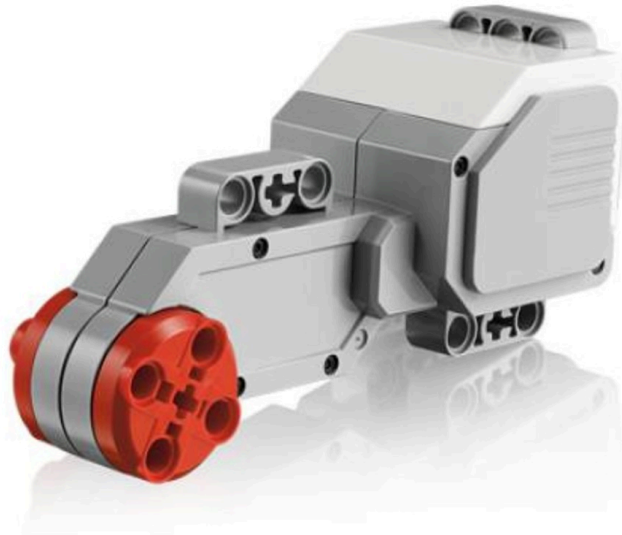
- Écran : l'écran LCD monochrome affiche des informations utiles telles que le statut du robot, les valeurs des capteurs, et plus encore. Il est également utilisé pour naviguer dans les menus de la brique.
- Boutons de navigation : les boutons situés sur le devant de la brique EV3 permettent de naviguer dans les menus, de démarrer et d'arrêter des programmes, et d'interagir avec le robot sans nécessiter un ordinateur.
- Connectivité : la brique EV3 offre plusieurs options de connectivité, y compris USB et Bluetooth, permettant de la programmer à distance et de la connecter à d'autres appareils.
- Logiciel et programmation : la brique EV3 est compatible avec divers langages de programmation et environnements de développement, allant des interfaces graphiques pour débutants aux langages textuels pour utilisateurs avancés.

[Télécharger le logiciel LEGO](#)

Moteurs

Deux types principaux de moteurs sont inclus dans les kits.

Grands moteurs



Les grands moteurs sont souvent utilisés pour les déplacements du robot, comme faire avancer ou tourner les roues. Ils sont conçus pour offrir un bon équilibre entre puissance et vitesse.

Moteurs moyens



Les moteurs moyens, quant à eux, sont utilisés pour des mouvements précis et contrôlés. Ils sont idéaux pour manipuler des objets ou positionner des éléments du robot avec beaucoup de précision.

Chaque moteur présente des caractéristiques de puissance et de précision qui peuvent être exploitées selon les besoins du projet.

Capteurs

Les capteurs sont les yeux et les oreilles du robot Mindstorms. Ils lui permettent de recevoir des informations de son environnement et d'agir en conséquence. Voici quelques-uns des capteurs inclus dans le kit.

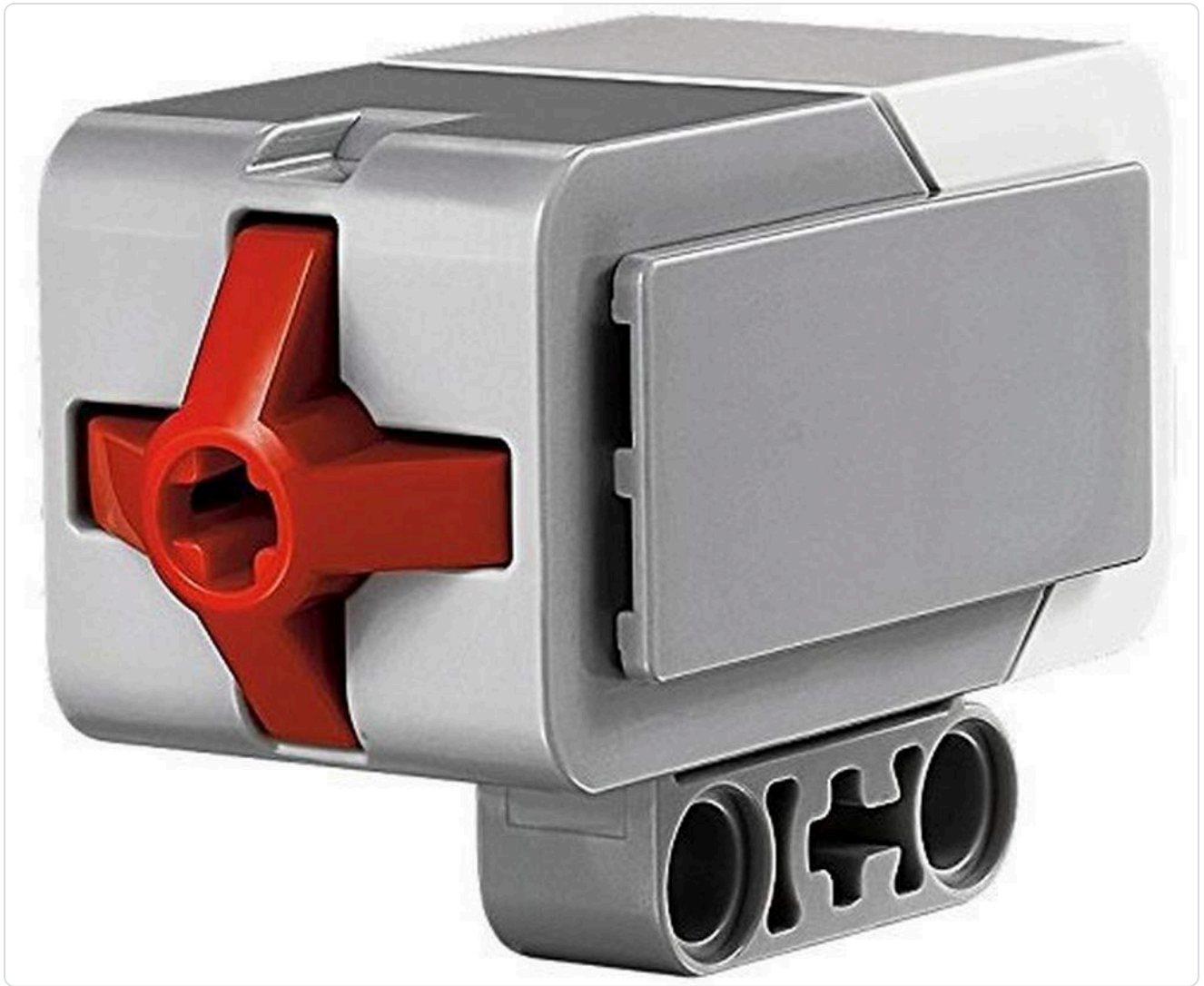
Capteur de couleur

Il détecte la couleur et l'intensité lumineuse, ce qui est utile pour le suivi de ligne ou la classification d'objets par couleur. Il fonctionne soit en mesurant directement la luminosité ambiante, soit en émettant une lumière spécifique qui lui permet de mesurer ce qui est réfléchi par une surface, et en particulier la couleur de celle-ci.



Capteur de toucher

Ce capteur permet au robot de détecter les collisions et les contacts avec des objets, simulant ainsi le sens du toucher. Il s'agit d'un simple bouton, qui est soit en position relâchée, soit en position pressée.



Capteur d'inclinaison

Il mesure l'inclinaison et les changements d'orientation (capteur gyroscopique), permettant au robot de maintenir son équilibre ou de naviguer sur des terrains variés.



Capteur à ultrasons

Le capteur à ultrasons est un outil indispensable pour la navigation et la détection d'objets. En émettant des ondes sonores à haute fréquence qui rebondissent sur les objets, ce capteur est capable de mesurer la distance à laquelle un objet se trouve par rapport au robot. Cette fonctionnalité est particulièrement utile pour éviter les obstacles, mesurer la proximité des objets, ou encore pour des applications de suivi et de cartographie de l'environnement.



D'autres capteurs peuvent également être inclus ou disponibles séparément, comme des capteurs infra-rouge ou de son.



Connectiques et câbles

Pour que tous ces composants puissent communiquer et fonctionner ensemble, ils doivent être correctement connectés à la brique intelligente EV3. Les câbles et connectiques jouent donc un rôle crucial dans la construction de tout robot Mindstorms.

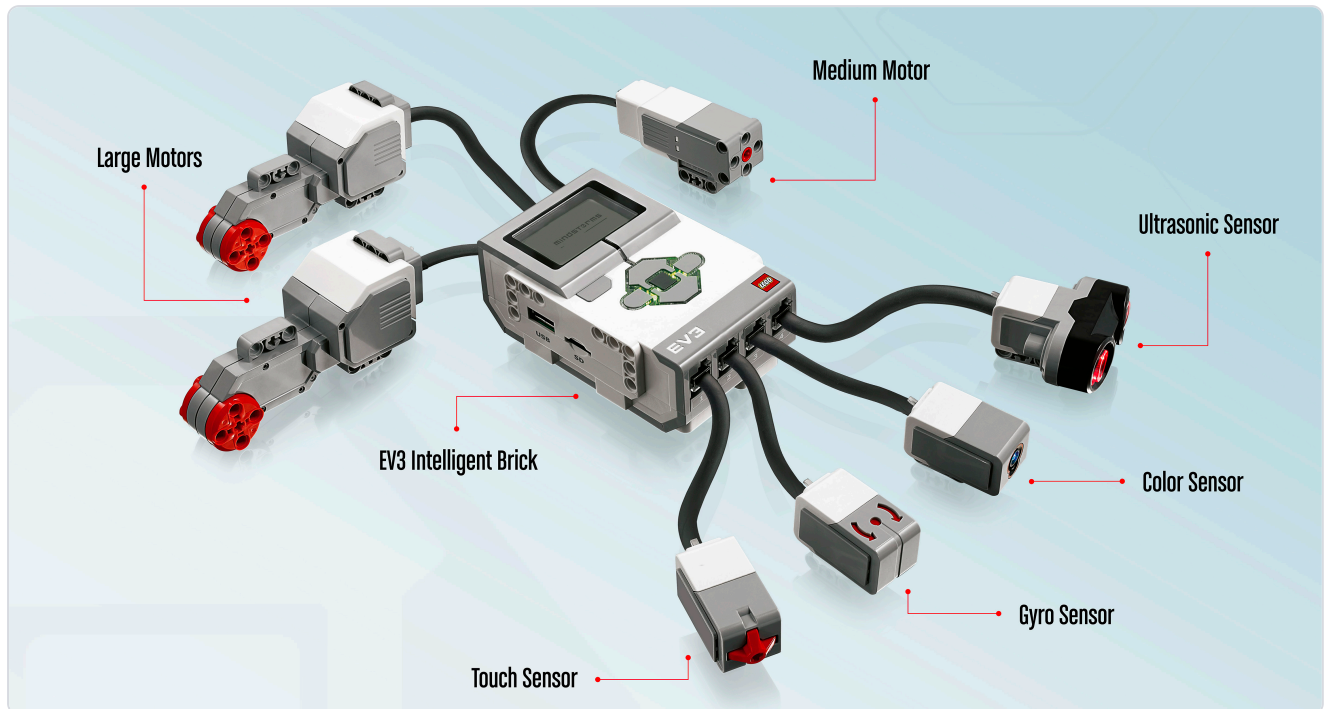
Câbles

Les câbles de différentes longueurs permettent de connecter les moteurs et les capteurs aux ports appropriés de la brique EV3. Ils existent en différentes longueurs, permettant de limiter l'encombrement tout en atteignant les différents objets.

Connectiques

Les connectiques sont conçues pour faciliter l'assemblage des composants, garantissant des connexions fiables et robustes.

Il est important d'organiser le câblage de manière à ne pas entraver les mouvements du robot et à éviter les déconnexions accidentelles.



Prise en main et configuration

L'expérience LEGO Mindstorms commence par l'assemblage et la configuration de votre premier robot. Cette section vous guidera à travers les étapes fondamentales pour construire, programmer et préparer votre robot à l'action.

Montage d'un robot de test

Construire un robot Mindstorms commence par le montage de base. C'est le fondement sur lequel vous allez développer et tester vos compétences en programmation.

Étape 1: Trier les pièces

Organisez les différentes pièces du kit pour faciliter le processus de construction.

Étape 2: Suivre les instructions de montage

Commencez le montage en suivant les instructions fournies étape par étape. Prenez votre temps pour vous assurer que chaque pièce est correctement assemblée.

[Instruction de montage](#)

Étape 3: Tester la construction

Une fois le montage terminé, vérifiez la solidité de la structure et assurez-vous que les moteurs et les capteurs sont bien fixés.

Installation du logiciel

Pour programmer votre robot, vous aurez besoin du logiciel Mindstorms. Ce logiciel offre un environnement de développement intuitif pour créer et tester vos programmes.

Téléchargement du logiciel

Rendez-vous sur le site officiel de LEGO Education pour télécharger la dernière version du logiciel compatible avec votre système d'exploitation.

[Télécharger le logiciel LEGO](#)

Installation sur votre ordinateur

Lancez l'installateur et suivez les instructions pour installer le logiciel sur votre ordinateur.

Découverte de l'interface

Ouvrez le logiciel et familiarisez-vous avec l'interface utilisateur. Explorez les différents menus et options disponibles.

Configuration initiale

La configuration de la brique intelligente EV3 est une étape cruciale pour préparer votre robot à l'exécution de programmes.

Connexion de la brique EV3

Connectez la brique EV3 à votre ordinateur via USB ou Bluetooth pour établir la communication.

Création d'un nouveau projet

Dans le logiciel Mindstorms, créez un nouveau projet et commencez à explorer les blocs de programmation disponibles.

Téléchargement d'un programme de test

Téléchargez et ouvrez ensuite le programme de test, et téléchargez-le sur la brique EV3 pour vérifier que tout fonctionne correctement.

(insérer un lien vers le programme de test)

Premiers pas avec le robot

Une fois le programme téléchargé, déconnectez la brique EV3 et lancez le programme pour voir votre robot en action.

2.2 Environnement de développement

Logiciel de programmation Mindstorms

Le logiciel de programmation LEGO Mindstorms est l'épine dorsale de la création et de la gestion des programmes qui donnent vie aux créations robotiques LEGO. Cet outil puissant permet aux utilisateurs de tous âges de concrétiser leurs idées en robotique de manière intuitive et éducative.

Historique et versions

LEGO Mindstorms a évolué au fil des années, avec plusieurs versions du logiciel qui ont accompagné les différentes générations de kits robotiques. De RIS (Robotics Invention System) à EV3 et maintenant avec la série Spike Prime, chaque itération a apporté des améliorations significatives en termes de fonctionnalités, de facilité d'utilisation et de compatibilité.

Insertion d'image: Une chronologie graphique des différentes versions du logiciel Mindstorms.

Intégration dans l'écosystème Mindstorms

Le logiciel de programmation n'est pas seulement un outil isolé; il fait partie intégrante de l'écosystème Mindstorms, qui comprend le matériel (briques programmables, capteurs, moteurs) et une communauté en ligne active. Il est conçu pour travailler de manière transparente avec ces composants, permettant une expérience utilisateur fluide du déballage à la programmation avancée.

Prise en main de l'interface

L'écran d'accueil

Lorsque vous lancez le logiciel de programmation LEGO Mindstorms, vous êtes accueilli par l'écran d'accueil. Cet écran est conçu pour offrir un accès rapide aux projets récents, à des ressources éducatives, et aux défis de programmation. C'est également à partir de cet écran que les nouveaux utilisateurs peuvent commencer des tutoriels interactifs qui les guident à travers les fonctionnalités de base du logiciel.

Insertion d'image: Capture d'écran de l'écran d'accueil avec des indications sur chaque section et fonctionnalité.

L'écran d'accueil est souvent divisé en plusieurs zones clairement définies :

1. **Zone de projets récents** : Ici, vous pouvez rapidement ouvrir les projets sur lesquels vous avez récemment travaillé, ce qui facilite la reprise de votre progression.
2. **Ressources éducatives** : Cette section fournit des liens vers des guides, des didacticiels et des exemples de projets qui peuvent aider les utilisateurs à améliorer leurs compétences en programmation.
3. **Défis de programmation** : Pour ceux qui cherchent à tester leurs compétences, cette partie propose des défis de programmation qui peuvent être tentés avec des robots Mindstorms.
4. **Tutoriels interactifs** : Des leçons pas à pas sont disponibles pour les débutants, offrant une introduction pratique au logiciel et à la programmation des robots Mindstorms.

Menus

L'interface utilisateur du logiciel Mindstorms est conçue pour être conviviale et accessible. Les menus sont intuitivement disposés, offrant un accès rapide aux fonctions de projet, à la configuration du matériel, aux options de programmation et aux ressources d'aide.

Insertion d'image: Capture d'écran annotée du menu principal du logiciel Mindstorms.

Barres d'outils

Les barres d'outils fournissent un accès instantané aux outils de programmation les plus utilisés. Que ce soit pour ajouter des blocs de code, configurer les connexions Bluetooth ou USB, ou simplement annuler la dernière action, ces outils sont facilement accessibles.

Insertion d'image: Illustration des différentes barres d'outils et de leur utilisation.

Zones de travail

La zone de travail principale est l'endroit où la magie de la programmation prend vie. C'est ici que les utilisateurs peuvent glisser-déposer des blocs de code pour construire leurs programmes. La zone de travail est accompagnée de palettes de blocs de code, reprenant l'ensemble des commandes disponibles.

Insertion d'image: Exemple de projet ouvert dans la zone de travail avec des annotations expliquant chaque composant de l'interface.

Fichier à télécharger: Un fichier de projet d'exemple que les utilisateurs peuvent ouvrir dans leur logiciel pour explorer la zone de travail avec un programme préconstruit.

Fonctionnalités

Le logiciel de programmation LEGO Mindstorms offre une pléthore de fonctionnalités conçues pour faciliter la création, la gestion et l'exécution de programmes pour les robots Mindstorms. Cet environnement riche permet aux utilisateurs de tous niveaux de s'engager dans la robotique éducative et ludique.

Création et gestion de projets

Démarrer un nouveau projet

Pour commencer un nouveau projet, il suffit de sélectionner l'option correspondante dans l'écran d'accueil ou dans le menu "Fichier". Un espace de travail vierge s'ouvre alors, prêt à être peuplé de vos idées de programmation.

Insertion d'image: Capture d'écran montrant le processus de création d'un nouveau projet.

Sauvegarde et chargement

Le logiciel permet de sauvegarder votre travail à tout moment. Vous pouvez également charger des projets existants pour les modifier ou les utiliser comme base pour de nouvelles créations.

Insertion d'image: Illustration du menu de sauvegarde et de chargement des projets.

Blocs de programmation et palettes

Les blocs de programmation

Les blocs de programmation sont les éléments de base avec lesquels les utilisateurs construisent leurs programmes. Chaque bloc représente une action, un contrôle ou une valeur spécifique qui peut être appliquée au robot.

Insertion d'image: Vue d'ensemble des différents types de blocs de programmation.

Utilisation des blocs

L'utilisation des blocs est intuitive : il suffit de les glisser-déposer dans la zone de travail. Les blocs peuvent être connectés pour former des séquences d'instructions qui définissent le comportement du robot. Nous verrons les différents blocs en détails dans la suite de ce cours.

Insertion d'image: Processus de glisser-déposer des blocs pour créer un programme.

Combinaison des blocs pour créer des programmes

Les blocs peuvent être combinés de manière créative pour réaliser des programmes complexes. Les utilisateurs peuvent utiliser des boucles, des conditions et des variables pour construire des algorithmes sophistiqués.

Insertion d'image: Exemple de programme complexe utilisant une combinaison de blocs.

Premiers pas en programmation

La programmation des robots LEGO Mindstorms est une aventure excitante qui débute avec la compréhension des principes fondamentaux. Les exemples pratiques sont un excellent moyen d'acquérir rapidement des compétences de base.

Votre premier programme

La création d'un premier programme simple est une étape cruciale dans l'apprentissage de la programmation. Nous allons créer un programme qui fera avancer le robot, puis reculer.

Étape 1: Démarrage du programme Commencez par ouvrir le logiciel Mindstorms et créez un nouveau projet. Glissez un bloc de démarrage sur la zone de travail.

Étape 2: Faire avancer le robot Ajoutez un bloc de mouvement configuré pour faire avancer le robot pendant une seconde.

Étape 3: Ajouter une pause Insérez un bloc de temporisation pour faire attendre le robot pendant une seconde avant d'exécuter l'action suivante.

Étape 4: Faire reculer le robot Placez un second bloc de mouvement pour faire reculer le robot pendant une seconde.

Étape 5: Arrêter le programme Terminez avec un bloc d'arrêt pour indiquer la fin du programme.

Insertion d'image: Capture d'écran de chaque étape de la création du programme dans le logiciel Mindstorms.

Fichier à télécharger: Un exemple de fichier de projet que les utilisateurs peuvent télécharger pour voir le programme finalisé.

Analyse et compréhension du code

Chaque bloc de programmation génère une partie du code qui contrôle le robot. Le code est purement linéaire, sans aucune option. Cela se voit sous forme graphique, car l'interface montre uniquement une série de bloc connectés les uns à la suite des autres, comme les maillons d'une chaîne.

Le robot suit alors cette chaîne, en appliquant les instructions de chaque maillon.

- Le premier maillon lui indique que c'est le début de la chaîne. Ce bloc est essentiel, car plusieurs chaînes peuvent être présentes et il faut donc indiquer laquelle suivre.
- Le second maillon indique au robot d'avancer vers l'avant durant une seconde.
- Le troisième maillon indique au robot d'attendre pendant une seconde sans rien faire.
- Le quatrième maillon est l'instruction contraire du deuxième : on indique au robot de reculer pendant une seconde. On peut observer que ces deux maillons se ressemblent, seule une option étant différente.

- Le cinquième maillon indique au robot qu'il a fini. Le robot ferme alors le programme et revient au menu principal (comme lorsqu'on ferme une fenêtre sur un ordinateur et que l'on revient au bureau).

Insertion d'image: Diagramme expliquant la logique de chaque bloc et comment ils s'enchaînent pour former le programme.

Exécution et transfert sur le robot

Une fois le programme créé, vous devrez l'exécuter pour vérifier son fonctionnement et ensuite le transférer sur le robot Mindstorms.

Étape 1: Exécution du programme Testez le programme dans le simulateur intégré au logiciel pour vous assurer qu'il fonctionne comme prévu. Cela se fait en conservant le robot connecté et en appuyant sur le bouton "Play".

Étape 2: Transfert sur le robot Connectez le robot à votre ordinateur via USB ou Bluetooth et transférez le programme. Cela se fait en cliquant sur le bouton "Télécharger".

Étape 3: Lancement sur le robot Une fois le programme transféré, démarrez-le sur le robot et observez-le exécuter les actions programmées. Pour cette étape, déconnectez le robot de l'ordinateur et lancez le programme directement depuis la brique intelligente.

Insertion d'image: Instructions visuelles pour l'exécution et le transfert du programme, y compris les icônes et les boutons à utiliser.

2.3 Programmation de base

La programmation de base des robots LEGO Mindstorms repose sur une compréhension claire des blocs de programmation. Ces blocs sont des composants logiciels qui, lorsqu'ils sont assemblés, forment les instructions pour le robot. Nous allons explorer les différentes catégories de blocs et leur utilisation.

Ce chapitre est largement issu de l'aide disponible directement dans le programme Minstorms.

Blocs moteur

Les blocs moteurs sont des outils essentiels pour la commande et la récupération d'informations des moteurs. Ils offrent une variété de fonctionnalités, allant de la rotation du moteur à la surveillance de sa vitesse et de sa position.

Faire tourner le moteur pendant une durée

Il est possible de faire tourner un moteur pour une durée déterminée, en fonction d'un nombre spécifié de rotations, de secondes ou de degrés. La vitesse de rotation est contrôlée par le bloc "Définir la vitesse", avec une valeur par défaut fixée à 75%.

Démarrage du moteur

Pour démarrer le moteur, on le fait tourner dans un sens ou dans l'autre jusqu'à la réception d'une instruction ultérieure ou l'arrêt du programme.

Arrêt du moteur

Commande l'arrêt du moteur. Cela se fait par défaut avec une tentative de maintien de la position actuelle.

Définir la vitesse du moteur

La vitesse d'un moteur peut être définie sur une échelle allant de -100% à 100%. Une valeur négative entraînera une rotation inverse à celle des aiguilles d'une montre. La vitesse par défaut est réglée à 75%.

Maintenir la position à l'arrêt

Ce paramètre détermine le comportement du moteur à la fin de sa commande en cours. Il peut soit se débloquent, soit maintenir activement sa position.

Commande de rotation du moteur à une vitesse spécifiée

Ce bloc fait tourner le moteur à une vitesse déterminée jusqu'à l'interruption par une nouvelle instruction ou la fin du programme. Une valeur de vitesse négative entraînera une rotation en sens inverse des aiguilles d'une montre.

Démarrage du moteur à une puissance spécifiée

Contrairement aux blocs de vitesse, ce bloc fait tourner le moteur à un pourcentage de puissance donné, maintenant un niveau de puissance constant sans régulation. Une valeur de puissance négative entraînera une rotation en sens inverse des aiguilles d'une montre.

Réinitialisation des degrés du moteur

Ce bloc remet à zéro le compteur de degrés du moteur, permettant de suivre la position relative du moteur depuis sa position initiale.

Lecture des degrés parcourus par le moteur

Ce bloc affiche le nombre de degrés que le moteur a parcouru depuis le début du programme ou depuis la dernière réinitialisation. Le compteur de degrés augmente avec la rotation dans le sens des aiguilles d'une montre et diminue dans le sens inverse.

Affichage de la vitesse actuelle du moteur

Ce bloc indique la vitesse actuelle d'un moteur en pourcentage, reflétant la vitesse réelle plutôt que la vitesse programmée.

Blocs de déplacement

Les blocs de déplacement sont conçus pour contrôler deux moteurs de manière synchronisée, facilitant ainsi le déplacement des bases motorisées. Il est important de noter que seuls deux moteurs identiques peuvent être synchronisés, tels que deux gros moteurs.

Déplacement pendant une durée déterminée

Ce bloc permet de déplacer un modèle vers l'avant ou vers l'arrière pour une durée spécifiée, mesurée en secondes, degrés ou rotations. Pour l'associer aux bons moteurs, utilisez le bloc "Définir les moteurs de déplacement" en alignant les ports du modèle avec ceux des gros moteurs, généralement B pour le gauche et C pour le droit. La vitesse des moteurs est réglée via le bloc "Définir la vitesse de déplacement", avec une vitesse par défaut établie à 50%.

Déplacement avec un angle pendant une durée

Ce bloc dirige le modèle pour un temps donné, avec un angle précis, selon le nombre défini de secondes, degrés ou rotations. Les ports des gros moteurs doivent être configurés à l'aide du bloc "Définir les moteurs de déplacement", avec les ports B et C comme valeurs par défaut pour le moteur gauche et droit respectivement. La vitesse est également déterminée par le bloc "Définir la vitesse de déplacement", fixée par défaut à 50%.

Arrêt du déplacement

Ce bloc est utilisé pour stopper immédiatement tout mouvement. Il nécessite la configuration des ports des gros moteurs correspondants via le bloc "Définir les moteurs de déplacement", avec les ports B et C par défaut pour le gauche et le droit.

Configuration des moteurs de déplacement

Ce bloc spécifie quels moteurs sont utilisés pour le déplacement. Le premier port mentionné correspond au moteur gauche, et le second au moteur droit, avec B et C comme réglages par défaut.

Maintien de la position des moteurs de déplacement à l'arrêt

Ce bloc détermine le comportement des moteurs de déplacement à la fin d'une commande. Ils peuvent soit se relâcher, soit maintenir activement la position actuelle à l'arrêt.

Déplacement avec un angle à une vitesse spécifiée pendant une durée

Ce bloc permet de déplacer un modèle à une vitesse et un angle définis, sur une durée spécifiée en secondes, degrés ou rotations. Les ports B et C sont utilisés par défaut pour les gros moteurs, et doivent être configurés via le bloc "Définir les moteurs de déplacement".

Déplacement à une vitesse spécifiée pendant une durée

Ce bloc commande le déplacement d'un modèle à une vitesse déterminée pour chaque moteur, sur une durée exprimée en secondes, degrés ou rotations. La première valeur indique la vitesse du moteur gauche, et la seconde celle du moteur droit. Les ports B et C sont les réglages par défaut pour les gros moteurs et doivent être ajustés avec le bloc "Définir les moteurs de déplacement".

Démarrage du déplacement avec un angle à une vitesse définie

Ce bloc initie le déplacement d'un modèle à une vitesse et un angle précisés, et continue jusqu'à la réception d'une nouvelle commande ou l'arrêt du programme. Les ports B et C sont les valeurs par défaut pour les gros moteurs et doivent être configurés avec le bloc "Définir les moteurs de déplacement".

Démarrage du déplacement à une vitesse définie

Ce bloc actionne le déplacement d'un modèle à une vitesse spécifique pour chaque moteur, et ce, jusqu'à la réception d'une nouvelle commande ou l'arrêt du programme. La vitesse du moteur gauche est définie par la première valeur, tandis que celle du moteur droit est définie par la seconde. Les ports B et C sont les valeurs par défaut pour les gros moteurs et doivent être ajustés à l'aide du bloc "Définir les moteurs de déplacement".

Blocs d'affichage

Les blocs d'affichage sont conçus pour présenter des éléments textuels ou graphiques sur l'écran de la brique EV3.

Affichage d'une image pour une durée limitée

Ce bloc affiche une image sélectionnée sur l'écran de la brique EV3 pour une durée déterminée en secondes.

Affichage permanent d'une image

Ce bloc permet d'afficher une image choisie sur l'écran de la brique EV3. L'image reste visible jusqu'à ce qu'un autre bloc d'affichage la remplace, que l'écran soit effacé, ou que le programme se termine.

Affichage de texte sur une ligne spécifique

Ce bloc affiche le texte indiqué à une ligne précise de l'écran de la brique EV3. Chaque ligne a une hauteur de 10 pixels. Le texte demeure affiché jusqu'à ce qu'il soit remplacé par un autre bloc d'affichage, que l'écran soit nettoyé ou que le programme s'arrête.

Affichage de texte à des coordonnées définies avec une police spécifique

Ce bloc affiche le texte spécifié avec la police choisie aux coordonnées XY déterminées sur l'écran de la brique EV3. Le texte reste à l'écran jusqu'à ce qu'il soit remplacé par un autre bloc d'affichage, que l'écran soit effacé, ou que le programme se termine.

Effacement de l'affichage

Ce bloc efface tout contenu présent sur l'écran de la brique EV3.

Définition du témoin d'état

Ce bloc configure le témoin d'état de la brique EV3 en définissant sa couleur et son mode de clignotement. Les options disponibles incluent vert, rouge, orange, clignotement vert, clignotement rouge et clignotement orange.

Blocs son

Les blocs son sont utilisés pour émettre des sons à travers le haut-parleur de la brique EV3, offrant ainsi une interaction auditive pendant l'exécution des programmes.

Jouer un son jusqu'à la fin

Ce bloc joue intégralement le son sélectionné sur la brique EV3 jusqu'à ce qu'il se termine complètement.

Démarrer un son

Ce bloc démarre le son choisi sur la brique EV3 et permet à la séquence de programmation de se poursuivre sans attendre la fin du son.

Jouer un bip pendant une durée déterminée

Ce bloc émet un bip sonore sur la brique EV3 pour une durée précise, mesurée en secondes.

Commencer à jouer un bip

Ce bloc joue un bip sur la brique EV3 qui continuera jusqu'à ce qu'une nouvelle commande soit donnée au haut-parleur ou que le programme arrive à son terme.

Arrêter tous les sons

Ce bloc interrompt immédiatement tous les sons qui sont en train d'être joués sur la brique EV3.

Régler le volume

Ce bloc permet de définir le volume de sortie du son sur la brique EV3, avec un niveau par défaut réglé à 100%.

Blocs événements

Les blocs événements, ou blocs chapeau, sont les éléments de base qui déclenchent l'exécution d'une pile de programmation en réponse à des événements spécifiques. Ils sont toujours placés en tête de pile et ne peuvent pas être suivis d'autres blocs chapeau.

Les blocs événement se déclenchent uniquement lorsque l'évènement se produit. Tant que la situation ne change pas, il ne se re-déclenchent pas.

Au lancement du programme

Ce bloc exécute une séquence d'instructions attachées dès le démarrage du programme. Le programme peut être lancé via le bouton "Télécharger et exécuter" ou directement depuis la brique EV3.

Quand la couleur est détectée

Ce bloc réagit lorsque le capteur de couleur identifie une couleur prédéfinie, comme le noir, le bleu, le vert, le jaune, le rouge, le blanc, ou le marron. Il s'active une seule fois pour chaque détection de la couleur spécifiée.

Lors d'une pression sur le capteur tactile

Ce bloc s'exécute quand le capteur tactile est pressé ou relâché, se déclenchant uniquement à l'activation ou la désactivation du capteur.

Quand la distance est mesurée

Ce bloc intervient lorsque la distance mesurée par le capteur à ultrasons par rapport à un objet répond à un critère spécifique, qu'elle soit inférieure, supérieure, égale, ou qu'elle change par rapport à une distance donnée en centimètres ou en pouces.

Lorsque la proximité est évaluée

Ce bloc se déclenche quand la proximité d'un objet par rapport au capteur infrarouge atteint un seuil défini, que ce soit en dessous, au-dessus, égal ou variant plus que la proximité spécifiée.

Quand le bouton de la télécommande infrarouge est pressé

Ce bloc agit lorsque le capteur infrarouge détecte une pression sur un bouton de la télécommande configuré sur un canal donné. Il s'active seulement lors de la pression du bouton.

Quand l'angle est mesuré par le capteur gyroscopique

Ce bloc est exécuté quand l'angle mesuré par le capteur gyroscopique est inférieur, supérieur, égal, ou change par rapport à un angle spécifié.

Lorsque le bouton de la brique EV3 est pressé

Ce bloc répond à la pression ou au relâchement d'un bouton spécifique sur la brique EV3. Il ne se réactive pas tant que l'état du bouton ne change pas.

Quand une condition booléenne est vraie

Ce bloc exécute les instructions attachées uniquement quand une condition booléenne prédéfinie est vraie.

À la réception d'un message

Ce bloc lance les instructions attachées lorsqu'un message spécifique est reçu, soit par le biais du bloc "Diffuser le message", soit par "Diffuser le message et attendre".

Diffuser le message

Ce bloc envoie un message spécifié et permet l'exécution immédiate du bloc suivant dans la pile, sans attendre que les autres piles de programmation réagissent au message.

Diffuser le message et attendre

Ce bloc diffuse un message spécifié et attend que toutes les piles de programmation qui écoutent ce message aient terminé leur exécution avant de continuer avec le bloc suivant.

Quand le chronomètre dépasse une valeur

Ce bloc se déclenche quand le chronomètre atteint ou dépasse une valeur donnée. Il ne s'active qu'une seule fois pour l'événement spécifié.

Blocs de contrôle

Les blocs de contrôle sont des éléments fondamentaux en programmation qui permettent de gérer le flux d'exécution d'un programme. Ils introduisent des concepts tels que l'attente, les boucles et les instructions conditionnelles, permettant ainsi de créer des algorithmes plus complexes et dynamiques.

Attendre pendant une durée définie

Ce bloc suspend l'exécution de la pile de programmation pour une durée spécifiée en secondes, qui peut être exprimée par un nombre entier ou décimal.

Algorithmique associée : Ce bloc implémente une temporisation, créant une pause dans le programme avant de poursuivre les instructions suivantes.

Attendre que la condition soit vraie

Ce bloc met en pause l'exécution jusqu'à ce qu'une condition booléenne spécifiée devienne vraie.

Algorithmique associée : Ce bloc crée une boucle d'attente active, souvent appelée "busy wait", qui vérifie continuellement si une condition est remplie pour continuer l'exécution.

Répéter un nombre défini de fois

Ce bloc répète l'exécution d'une série de blocs internes pour le nombre de fois indiqué avant de reprendre la suite de la pile de programmation.

Algorithmique associée : Ce bloc implémente une boucle for, une structure de contrôle qui répète les instructions un nombre précis de fois.

Répéter indéfiniment

Ce bloc crée une boucle qui répète indéfiniment les blocs internes jusqu'à ce qu'une action externe, comme l'appui sur le bouton d'arrêt ou l'exécution d'un bloc d'arrêt, intervienne.

Algorithmique associée : Ce bloc représente une boucle while avec une condition toujours vraie, ce qui crée une boucle infinie.

Répéter jusqu'à ce que la condition soit vraie

Ce bloc répète l'exécution des blocs internes jusqu'à ce que la condition booléenne spécifiée soit vraie, puis continue avec le reste de la pile de programmation.

Algorithmique associée : Ce bloc implémente une boucle while qui continue tant que la condition n'est pas satisfaite.

Si la condition est vraie, alors exécuter

Ce bloc exécute les instructions internes uniquement si la condition booléenne est vraie, avant de poursuivre avec le reste de la pile de programmation.

Algorithmique associée : Ce bloc représente une instruction conditionnelle if, exécutant des instructions seulement si une condition est remplie.

Si la condition est vraie, alors exécuter, sinon exécuter autre chose

Ce bloc exécute un ensemble d'instructions si la condition booléenne est vraie et un autre ensemble si elle est fausse, avant de continuer l'exécution du programme.

Algorithmique associée : Ce bloc introduit une structure if-else, permettant d'exécuter des instructions alternatives en fonction de la véracité d'une condition.

Arrêter les autres piles de programmation

Ce bloc arrête l'exécution de toutes les piles de programmation en cours, à l'exception de celle qui contient ce bloc.

Algorithmique associée : Ce bloc permet de gérer la concurrence entre différentes piles de programmation, en forçant l'arrêt des autres processus.

Arrêter l'exécution

Ce bloc arrête l'exécution de la pile de programmation actuelle, ou de toutes les piles, ou même de tout le programme.

Algorithmique associée : Ce bloc agit comme une instruction de sortie, terminant l'exécution de la pile de programmation ou du programme entier.

Blocs capteurs

Les blocs capteurs sont utilisés pour recueillir des informations à partir des différents capteurs connectés à la brique EV3. Ils jouent un rôle crucial dans la perception de l'environnement par le robot.

Intensité de la lumière réfléchie

Question sur l'intensité de la lumière réfléchie

Ce bloc renvoie `true` si l'intensité de la lumière réfléchie mesurée par le capteur de couleur correspond aux critères spécifiés (supérieure, égale ou inférieure à un pourcentage donné).

Mesure de l'intensité de la lumière réfléchie

Ce bloc indique l'intensité de la lumière réfléchie en pourcentage, ajustée en fonction de l'étalonnage du capteur de couleur.

Étalonnage de l'intensité de la lumière réfléchie

Ce bloc permet d'ajuster la sensibilité du capteur de couleur en définissant les seuils minimums et maximums pour l'intensité de la lumière réfléchie.

Réinitialisation de l'étalonnage de l'intensité de la lumière réfléchie

Ce bloc rétablit les paramètres d'étalonnage du capteur de couleur à leurs valeurs par défaut.

Détection de couleur

Attente d'une couleur spécifique

Ce bloc met en pause la pile de programmation jusqu'à ce que le capteur de couleur détecte la couleur demandée, parmi un ensemble de couleurs reconnues.

Question sur la couleur détectée

Ce bloc renvoie `true` si le capteur de couleur identifie la couleur spécifiée parmi celles qu'il peut reconnaître.

Identification de la couleur détectée

Ce bloc fournit la couleur actuellement détectée par le capteur de couleur, exprimée sous forme numérique, où chaque numéro correspond à une couleur spécifique.

- 0 : Aucune couleur
- 1 : Noir
- 2 : Bleu
- 3 : Vert
- 4 : Jaune
- 5 : Rouge
- 6 : Blanc

- 7 : Marron

Intensité de la lumière ambiante

Question sur l'intensité de la lumière ambiante

Ce bloc renvoie `true` si l'intensité de la lumière ambiante mesurée par le capteur de couleur est conforme aux critères spécifiés.

Mesure de l'intensité de la lumière ambiante

Ce bloc affiche l'intensité de la lumière ambiante captée par le capteur de couleur en pourcentage.

Capteur tactile

Attente d'une pression

Ce bloc suspend l'exécution du programme jusqu'à ce que le capteur tactile soit pressé ou relâché.

Question sur l'état du capteur tactile

Ce bloc renvoie `true` si le capteur tactile est actuellement pressé.

Capteur à ultrasons

Attente d'une distance spécifique

Ce bloc met en pause la pile de programmation jusqu'à ce que la distance mesurée par le capteur à ultrasons corresponde à un critère donné (inférieure, supérieure ou égale à une valeur spécifiée).

Question sur la distance mesurée

Ce bloc renvoie `true` si la distance mesurée par le capteur à ultrasons est conforme aux critères définis.

Mesure de la distance

Ce bloc indique la distance entre le capteur à ultrasons et un objet, en centimètres ou en pouces.

Capteur infrarouge

Attente d'une proximité spécifique

Ce bloc suspend l'exécution du programme jusqu'à ce que la proximité mesurée par le capteur infrarouge réponde à un critère défini.

Question sur la proximité

Ce bloc renvoie `true` si la proximité avec un objet mesurée par le capteur infrarouge est conforme aux critères spécifiés.

Mesure de la proximité

Ce bloc fournit la proximité d'un objet détectée par le capteur infrarouge, exprimée en pourcentage.

Attente de l'activation d'un bouton de balise

Ce bloc met en pause la pile de programmation jusqu'à ce que le capteur infrarouge détecte un appui sur un bouton de la télécommande, configuré sur un canal donné.

Question sur l'état du bouton de balise

Ce bloc renvoie `true` si le capteur infrarouge détecte que le bouton de balise spécifié est pressé sur le canal choisi.

Identification du bouton de balise pressé

Ce bloc indique quel bouton de la télécommande infrarouge est pressé, selon le canal spécifié, avec une valeur numérique correspondant à chaque combinaison de boutons.

- 0 : Pas de bouton (balise inactive)
- 1 : Bouton supérieur gauche
- 2 : Bouton inférieur gauche
- 3 : Bouton supérieur droit
- 4 : Bouton inférieur droit
- 5 : Les deux boutons supérieurs
- 6 : Bouton supérieur gauche et bouton inférieur droit
- 7 : Bouton supérieur droit et bouton inférieur gauche
- 8 : Les deux boutons inférieurs
- 9 : Balises actives
- 10 : Les deux boutons gauche
- 11 : Les deux boutons droits

Capteur gyroscopique

Attente d'un angle spécifique

Ce bloc suspend l'exécution du programme jusqu'à ce que l'angle mesuré par le capteur gyroscopique corresponde à un critère défini.

Question sur l'angle mesuré

Ce bloc renvoie `true` si l'angle mesuré par le capteur gyroscopique est conforme aux critères spécifiés.

Mesure de l'angle

Ce bloc affiche l'angle actuel mesuré par le capteur gyroscopique en degrés.

Réinitialisation de l'angle

Ce bloc remet à zéro la mesure de l'angle du capteur gyroscopique.

Mesure de la vitesse angulaire

Ce bloc indique la vitesse angulaire du capteur gyroscopique, exprimée en degrés par seconde.

Bouton de la brique EV3

Attente de l'activation d'un bouton de la brique

Ce bloc met en pause la pile de programmation jusqu'à ce que le bouton spécifié sur la brique EV3 soit pressé ou relâché.

Question sur l'état du bouton de la brique

Ce bloc renvoie `true` si le bouton spécifié sur la brique EV3 est actuellement pressé.

Identification du bouton de la brique pressé

Ce bloc fournit l'état du bouton de la brique EV3, exprimé en nombre, où chaque numéro correspond à un bouton spécifique.

- 0 : Pas de bouton
- 1 : Bouton gauche
- 2 : Bouton central
- 3 : Bouton droit
- 4 : Bouton supérieur

- 5 : Bouton inférieur

Chronomètre

Mesure du temps écoulé

Ce bloc indique le temps écoulé en secondes depuis le début du programme. Le chronomètre est réinitialisé à chaque redémarrage du programme.

Réinitialisation du chronomètre

Ce bloc remet à zéro le chronomètre. La valeur actuelle du chronomètre peut être consultée à tout moment avec le bloc correspondant.

Blocs opérateurs

Les blocs opérateurs sont essentiels en programmation, car ils permettent d'effectuer des opérations mathématiques et logiques avec des valeurs numériques et booléennes.

Sélection aléatoire d'un nombre

Ce bloc choisit un nombre aléatoirement dans une plage définie, incluant les valeurs extrêmes. Par exemple, pour une plage de 1 à 3, il peut renvoyer 1, 2 ou 3.

Addition

Ce bloc additionne deux valeurs numériques et renvoie le résultat.

Soustraction

Ce bloc soustrait la seconde valeur de la première et renvoie le résultat.

Multiplication

Ce bloc multiplie deux valeurs numériques et renvoie le résultat.

Division

Ce bloc divise la première valeur numérique par la seconde et renvoie le résultat.

Comparaison inférieure

Ce bloc renvoie `true` si la première valeur numérique est inférieure à la seconde.

Comparaison égale

Ce bloc renvoie `true` si la première valeur numérique est égale à la seconde.

Comparaison supérieure

Ce bloc renvoie `true` si la première valeur numérique est supérieure à la seconde.

Opérateur logique ET

Ce bloc renvoie `true` si les deux valeurs booléennes sont `true`.

Opérateur logique OU

Ce bloc renvoie `true` si au moins une des deux valeurs booléennes est `true`.

Opérateur logique NON

Ce bloc renvoie `true` si la valeur booléenne est `false`.

Concaténation de chaînes

Ce bloc réunit deux valeurs textuelles et renvoie le résultat. Par exemple, pour "Hello" et "World", il renvoie "HelloWorld". Pour inclure un espace, "Hello " et "World" donneront "Hello World".

Longueur d'une chaîne

Ce bloc indique le nombre de caractères présents dans la chaîne de caractères spécifiée.

Modulo

Ce bloc renvoie le reste de la division de la première valeur numérique par la seconde. Par exemple, 10 divisé par 3 renvoie un reste de 1.

Arrondi

Ce bloc arrondit le nombre spécifié au nombre entier le plus proche selon les règles d'arrondi standard.

Fonctions mathématiques

Ce bloc applique la fonction mathématique spécifiée au nombre donné et renvoie le résultat.

Blocs variables

Les blocs variables sont des composants essentiels de la programmation qui permettent de stocker et de manipuler des données sous forme de variables, de listes et de structures personnalisées.

Variable

Affichage de la valeur d'une variable

Ce bloc affiche la valeur actuelle d'une variable. Une instance de ce bloc est générée pour chaque variable créée, portant le nom de la variable correspondante.

Définition d'une variable

Définir la variable

Ce bloc assigne une valeur spécifiée à une variable, qui peut être un nombre ou une chaîne de caractères.

Modification d'une variable

Modifier la variable

Ce bloc change la valeur d'une variable par la quantité spécifiée. Si la variable est une chaîne de caractères plutôt qu'un nombre, elle est traitée comme ayant une valeur de 0 et est donc mise à jour avec la nouvelle valeur spécifiée.

Listes

Affichage de la valeur d'une liste

Ce bloc indique les données contenues dans une liste. Une version de ce bloc est créée pour chaque liste avec son nom spécifique.

Ajout d'un élément à une liste

Ce bloc ajoute un élément, qui peut être un texte ou un nombre, à la fin d'une liste spécifiée.

Suppression des éléments d'une liste

Ce bloc efface tous les éléments contenus dans une liste donnée.

Remplacement d'un élément dans une liste

Ce bloc remplace l'élément à une position donnée dans une liste par un nouvel élément, qui peut être un texte ou un nombre.

Valeur d'un élément dans une liste

Ce bloc renvoie la valeur de l'élément situé à une position spécifique dans une liste.

Longueur d'une liste

Ce bloc donne le nombre total d'éléments présents dans une liste.

Création de blocs personnalisés

Bloc définir

Ce bloc permet à l'utilisateur de créer des blocs personnalisés. Il s'agit du groupe de blocs qui sont attachés à ce bloc "Définir".

Mon bloc

Ce bloc exécute la fonctionnalité définie par l'utilisateur à travers le bloc "Définir".

CHAPITRE 3

Programmation

Algorithmique

Qu'est-ce qu'un algorithme ?

Dans cette section, nous explorerons la définition d'un algorithme, son rôle crucial en programmation, et comment il est utilisé pour résoudre des problèmes.

Notions de base de l'algorithmique

Nous aborderons les concepts élémentaires de l'algorithmique, incluant les variables, les types de données, ainsi que les opérations et expressions fondamentales.

Structures de contrôle

Cette partie introduira les structures conditionnelles telles que `if` et `else`, ainsi que les différentes boucles utilisées en programmation, comme `for` et `while`.

Pseudo-code

Nous apprendrons à rédiger des algorithmes en pseudo-code, un outil essentiel pour conceptualiser et planifier la logique d'un programme avant sa mise en œuvre.

Langages de programmation

Introduction aux langages de programmation

Nous présenterons les divers langages de programmation existants et leurs champs d'application spécifiques. Nous nous concentrerons sur le Python, qui sera utilisé dans ce cours.

Syntaxe et paradigmes de programmation

Syntaxe et sémantique

Nous expliquerons l'importance de la syntaxe et de la sémantique dans les langages de programmation et comment ces deux aspects influencent la compréhension du code.

Paradigmes de programmation

Nous offrirons une vue d'ensemble des différents paradigmes de programmation, y compris la programmation impérative, orientée objet et fonctionnelle.

Programmation structurée

Fonctions et procédures

Nous verrons comment utiliser et définir des fonctions et des procédures afin de structurer et d'organiser le code de manière efficace.

Portée des variables

Nous examinerons la notion de portée des variables, en distinguant les variables locales des variables globales, et l'importance des espaces de noms.

Modularité et réutilisation du code

Cette partie traitera des principes de modularité et des techniques favorisant la réutilisation du code pour créer des programmes plus maintenables et évolutifs.

Bonnes pratiques

Commentaires et documentation

Nous soulignerons l'importance des commentaires et de la documentation pour assurer la clarté et la pérennité du code.

Conventions de codage

Nous apprendrons les conventions de codage standard qui contribuent à la lisibilité et à la maintenance du code.

Tests et validation

Nous introduirons les concepts de tests unitaires et de validation du code pour garantir sa fiabilité et sa qualité.

3.1 Algorithmique

Qu'est-ce qu'un algorithme ?

Un algorithme est une série d'instructions ordonnées et non ambiguës permettant de résoudre un problème ou d'effectuer une tâche. Imaginez une recette de cuisine : elle liste les étapes à suivre pour préparer un plat. De la même manière, un algorithme guide l'ordinateur à travers les étapes nécessaires pour parvenir à un résultat spécifique.

Insérer une image conceptuelle d'un algorithme, comme un diagramme de flux simplifié illustrant les étapes d'un processus.

Exemple en Python :

```
# Algorithme pour ajouter deux nombres
def additionner(a, b):
    resultat = a + b
    return resultat

print(additionner(5, 7))
```

PYTHON

Notions de base de l'algorithmique

Avant de plonger dans la création d'algorithmes complexes, il est essentiel de comprendre les fondements de l'algorithmique.

Variables

Une variable en programmation est une abstraction qui représente un emplacement de mémoire. Cet emplacement de mémoire est utilisé pour stocker des valeurs qui peuvent être modifiées au cours de l'exécution d'un programme. En d'autres termes, une variable est comme une boîte dans laquelle nous pouvons mettre des données, les modifier et les récupérer lorsque nécessaire.

Insérer une image représentant une variable comme une boîte avec une étiquette, qui contient une valeur.

Exemple en Python :

```
# Création d'une variable
nombre = 10 # La valeur 10 est stockée dans la mémoire et la variable 'nombre' y fait référence
```

PYTHON

Logique de mémoire

Quand une variable est créée, le système d'exploitation alloue un espace de mémoire pour stocker sa valeur. Cette allocation est gérée en arrière-plan, et le développeur n'a pas besoin de connaître l'emplacement physique de la mémoire. Cependant, il est important de comprendre que chaque variable a une adresse mémoire unique, et c'est ainsi que l'ordinateur peut suivre où les données sont stockées et comment y accéder.

On peut comparer cela à une boîte aux lettres. L'adresse permet de savoir quelle boîte aux lettres accéder, et le contenu de la boîte aux lettres est la valeur qu'on a stockée. Le système d'exploitation joue alors le rôle de carnet d'adresse. Il retient que le nom de la variable pointe vers une adresse donnée. Du point de vue du développeur, seul le nom dans le carnet d'adresse est donc important.

Insérer une image montrant la mémoire d'un ordinateur avec des adresses et des variables pointant vers ces adresses.

Exemple en Python :

```
# Assignment d'une nouvelle valeur à une variable
nombre = 15 # La valeur précédente est remplacée par 15 dans le même emplacement de
mémoire
```

PYTHON

Types de données

Les types de données sont essentiels en programmation car ils indiquent à l'ordinateur la nature des données qu'il doit manipuler. Chaque type de donnée a un ensemble de caractéristiques qui définissent les opérations qui peuvent être effectuées avec ces données et la quantité de mémoire nécessaire pour les stocker.

Entiers (`int`)

Les entiers, ou `int`, sont des nombres sans partie décimale. Ils peuvent être positifs ou négatifs et sont souvent utilisés pour compter ou itérer sur des éléments, comme dans une boucle.

Exemple en Python :

```
# Un entier
nombre_de_pommes = 5
```

PYTHON

Nombres à virgule flottante (`float`)

Les nombres à virgule flottante, ou `float`, sont des nombres réels qui incluent une partie décimale. Ils sont utilisés lorsque des calculs précis sont nécessaires, comme en ingénierie ou en sciences.

Exemple en Python :

```
# Un nombre flottant
pi = 3.14159
```

PYTHON

Chaînes de caractères (`str`)

Les chaînes de caractères, ou `str`, sont des séquences de caractères utilisées pour représenter du texte. Elles sont entourées de guillemets simples ou doubles et peuvent contenir des lettres, des chiffres, des espaces et des symboles.

Exemple en Python :

```
# Une chaîne de caractères
message = "Bonjour tout le monde!"
```

PYTHON

Booléens (`bool`)

Les booléens, ou `bool`, sont un type de donnée simple qui ne peut prendre que deux valeurs : `True` (vrai) ou `False` (faux). Ils sont particulièrement utiles dans les structures de contrôle pour prendre des décisions en fonction de conditions.

Exemple en Python :

```
# Un booléen
lampe_allumee = True
```

PYTHON

Ces types de données affectent la manière dont les données sont stockées en mémoire et la façon dont les opérations sont effectuées sur ces données. Choisir le type de donnée approprié pour une variable est une décision importante qui peut affecter la précision et l'efficacité d'un programme.

Opérations et expressions

Les opérations en programmation sont des actions qui manipulent des données. Elles peuvent être arithmétiques, logiques, de comparaison, ou d'affectation, entre autres. Les expressions, quant à elles, sont des combinaisons d'opérateurs et de valeurs (qui peuvent être des variables ou des constantes) formant une suite logique qui, une fois évaluée, produit une nouvelle valeur.

Opérations arithmétiques

Les opérations arithmétiques incluent l'addition (+), la soustraction (-), la multiplication (*), la division (/), et le modulo (%), qui retourne le reste d'une division.

Exemple en Python :

PYTHON

```
# Opérations arithmétiques
a = 10
b = 3
somme = a + b           # Résultat: 13
difference = a - b      # Résultat: 7
produit = a * b         # Résultat: 30
quotient = a / b        # Résultat: 3.333...
reste = a % b           # Résultat: 1
```

Opérations logiques

Les opérations logiques sont des opérations qui appliquent des règles de logique booléenne aux valeurs `True` (vrai) et `False` (faux). Ces opérations sont fondamentales en programmation, car elles permettent de contrôler le flux d'exécution du code en fonction de conditions multiples. Les trois opérateurs logiques de base sont `AND`, `OR`, et `NOT`.

Insérer une image illustrant un circuit logique avec des portes `AND`, `OR`, et `NOT`.

Opérateur logique `AND`

L'opérateur `AND` retourne `True` si et seulement si tous les opérandes sont `True`. Voici sa table de vérité :

	True	False
True	True	False
False	False	False

Opérateur logique `OR`

L'opérateur `OR` retourne `True` si au moins un des opérandes est `True`. Voici sa table de vérité :

	True	False
True	True	True
False	True	False

Opérateur logique `NOT`

L'opérateur **NOT** est un opérateur unaire qui inverse la valeur de vérité de l'opérande. Voici sa table de vérité :

A	NOT A
True	False
False	True

Exemple en Python :

```
# Opérations logiques
vrai = True
faux = False
resultat_and = vrai and faux # Résultat: False
resultat_or = vrai or faux   # Résultat: True
resultat_not = not vrai      # Résultat: False
```

PYTHON

Les tables de vérité sont un outil précieux pour comprendre le résultat des opérations logiques. Elles permettent de prévoir le résultat d'une expression logique, quelles que soient les valeurs de vérité des opérandes. En programmation, les opérations logiques sont souvent utilisées dans les structures de contrôle pour tester des conditions complexes et déterminer le chemin d'exécution d'un programme.

Opérations de comparaison

En programmation, il est crucial de distinguer l'opérateur d'affectation **=** de l'opérateur de comparaison d'égalité **==**. Bien que ces deux opérateurs puissent sembler similaires, ils ont des fonctions très différentes.

Opérateur d'affectation =

L'opérateur **=** est utilisé pour assigner une valeur à une variable. Lorsqu'on utilise cet opérateur, la valeur de droite est calculée et stockée dans la variable située à gauche de l'opérateur.

Exemple en Python :

```
# Affectation de la valeur 10 à la variable 'nombre'
nombre = 10
```

PYTHON

Dans cet exemple, **nombre** est une variable à laquelle on affecte la valeur **10**. Après l'exécution de cette ligne de code, la variable **nombre** contiendra **10**.

Opérateur de comparaison d'égalité ==

L'opérateur `=` est utilisé pour comparer deux valeurs afin de déterminer si elles sont égales. Si les valeurs sont égales, l'expression retourne `True` ; sinon, elle retourne `False`.

Exemple en Python :

PYTHON

```
# Comparaison de la variable 'nombre' avec la valeur 10
resultat = (nombre == 10) # Résultat: True
```

Dans cet exemple, `(nombre == 10)` est une expression de comparaison qui vérifie si la valeur de la variable `nombre` est égale à `10`. Comme `nombre` a été précédemment affecté à `10`, l'expression retourne `True`.

Autres opérateurs de comparaison

En plus de l'égalité, il existe d'autres opérateurs de comparaison qui permettent de tester des relations différentes entre les valeurs :

- La non-égalité (`≠`) vérifie si deux valeurs sont différentes.
- Le supérieur (`>`) vérifie si la valeur de gauche est supérieure à celle de droite.
- L'inférieur (`<`) vérifie si la valeur de gauche est inférieure à celle de droite.
- Le supérieur ou égal (`≥`) vérifie si la valeur de gauche est supérieure ou égale à celle de droite.
- L'inférieur ou égal (`≤`) vérifie si la valeur de gauche est inférieure ou égale à celle de droite.

Expressions

Une expression peut être simple, comme une seule valeur ou une variable, ou complexe, combinant plusieurs opérations. Les expressions sont évaluées par l'ordinateur pour produire une nouvelle valeur, qui peut ensuite être utilisée ou affectée à une variable.

Exemple en Python :

PYTHON

```
# Expression complexe
resultat = (a + b) * (a - b)
```

Dans cet exemple, l'expression `(a + b) * (a - b)` est d'abord évaluée en effectuant l'addition et la soustraction, puis en multipliant les résultats de ces deux opérations.

Les opérations et expressions sont au cœur de la logique de programmation. Elles permettent aux programmeurs de définir la manière dont les données sont traitées et les règles selon lesquelles les décisions sont prises au sein d'un programme.

Structures de contrôle

Les structures de contrôle sont des éléments de syntaxe qui dirigent le flux d'exécution du code selon des conditions logiques ou des séquences répétitives.

Conditionnelles

Les instructions conditionnelles permettent de diriger l'exécution de certaines parties d'un programme en fonction de conditions booléennes. Les structures conditionnelles de base sont `if`, `elif` (else if), et `else`, signifiant `si`, `sinon si` et `sinon`.

`if`

L'instruction `if` (si) est utilisée pour exécuter un bloc de code seulement si une condition spécifiée est vraie.

Exemple en Python :

```
age = 20
if age ≥ 18:
    print("Vous êtes majeur.")
```

PYTHON

Dans cet exemple, le message ne sera affiché que si la valeur de `age` est supérieure ou égale à 18.

`elif`

L'instruction `elif`, abréviation de "else if" (sinon si), permet de vérifier plusieurs conditions successives. Si la condition pour `if` est fausse, le programme vérifie la condition suivante dans `elif`.

Exemple en Python :

```
age = 16
if age ≥ 18:
    print("Vous êtes majeur.")
elif age == 17:
    print("Vous serez bientôt majeur.")
```

PYTHON

Dans cet exemple, si `age` est 17, le message "Vous serez bientôt majeur." sera affiché.

`else`

L'instruction `else` (sinon) capture tous les cas qui ne répondent pas aux conditions spécifiées par `if` et `elif`. C'est une sorte de filet de sécurité qui permet d'exécuter un bloc de code si toutes les autres conditions sont fausses.

Exemple en Python :

PYTHON

```
age = 16
if age ≥ 18:
    print("Vous êtes majeur.")
elif age = 17:
    print("Vous serez bientôt majeur.")
else:
    print("Vous êtes mineur.")
```

Dans cet exemple, si `age` est inférieur à 17, le message "Vous êtes mineur." sera affiché.

Boucles

Les boucles sont des structures qui répètent un bloc de code tant qu'une condition donnée est vraie. Les deux types de boucles les plus courants sont `for` et `while`, signifiant `pour` et `pendant que`.

`for`

La boucle `for` (pour) est utilisée pour itérer sur une séquence (qui peut être une liste, une chaîne de caractères, ou une plage de nombres).

Exemple en Python :

PYTHON

```
for i in range(5):
    print(i)
```

Dans cet exemple, `print(i)` est exécuté pour chaque valeur de `i` de 0 à 4.

`while`

La boucle `while` (pendant que) exécute un bloc de code tant que la condition spécifiée est vraie.

Exemple en Python :

PYTHON

```
compteur = 0
while compteur < 5:
    print(compteur)
    compteur += 1
```

Dans cet exemple, le bloc de code continuera à s'exécuter jusqu'à ce que `compteur` soit égal à 5.

Pseudo-code

Le pseudo-code est une manière informelle de noter un algorithme. Il utilise des termes de langage courant pour décrire les opérations, ce qui le rend accessible même à ceux qui ne connaissent pas de langage de programmation spécifique.

Exemple de pseudo-code :

```
POUR chaque nombre de 1 à 5 FAIRE  
    AFFICHER nombre  
FIN POUR
```

Équivalent en Python :

```
for nombre in range(1, 6):  
    print(nombre)
```

PYTHON

Dans ce cours, nous nous concentrerons sur la syntaxe du langage Python (nous reviendrons sur ce choix dans la suite). Le pseudo code permet de construire la logique en dehors d'autres contextes, ce qui est pratique au début. Toutefois, il s'agit d'une syntaxe à apprendre comme n'importe quel langage de programmation. Dès lors, puisqu'il est nécessaire d'apprendre une nouvelle façon d'écrire, nous avons considéré plus intéressant de directement apprendre une syntaxe utilisable sur un ordinateur.

3.2 Langages de programmation

Introduction aux langages de programmation

Les langages de programmation sont des moyens de communication avec les ordinateurs pour leur donner des instructions précises. Chaque langage a ses propres règles syntaxiques et s'adapte à différentes tâches. Dans ce chapitre, nous découvrirons brièvement divers langages, tels que Python, Java, C++, et JavaScript, en mettant l'accent sur Python pour sa simplicité et sa polyvalence.

Python est particulièrement apprécié pour son code lisible et sa syntaxe claire, ce qui en fait un choix idéal pour les débutants. Il est également très utilisé dans des domaines variés tels que le développement web, l'analyse de données, l'intelligence artificielle, et bien sûr, la robotique.

Histoire des langages de programmation

L'histoire des langages de programmation est riche et variée, reflétant l'évolution des technologies informatiques et des besoins des utilisateurs. Voici un aperçu des moments clés et des étapes importantes des langages de programmation les plus influents.

Naissance de la programmation

- **1940s-1950s** : Les premiers ordinateurs sont programmés avec des langages de bas niveau comme l'assembleur, spécifiques à chaque machine.
- **1957** : Fortran (Formula Translation) est développé par IBM, devenant le premier langage de haut niveau largement adopté, principalement pour les calculs scientifiques et l'ingénierie.
- **1959** : COBOL (Common Business-Oriented Language) est créé pour les applications d'entreprise, mettant l'accent sur la gestion des fichiers et des données.

Évolution et diversification

- **1960s** : Le langage ALGOL (Algorithmic Language) influence de nombreux langages, notamment avec l'introduction de la notion de blocs de code.
- **1970** : Pascal est introduit par Niklaus Wirth, promouvant la programmation structurée et servant d'outil pédagogique.
- **1972** : C, développé par Dennis Ritchie chez Bell Labs, apporte une combinaison de puissance, d'efficacité et de portabilité, servant de base à de nombreux systèmes d'exploitation, dont UNIX.

Période de consolidation

- **1980s** : Avec l'avènement des ordinateurs personnels, des langages comme BASIC et ses dérivés, notamment Visual Basic, deviennent populaires pour leur facilité d'utilisation.
- **1983** : Bjarne Stroustrup crée le C++, ajoutant des fonctionnalités orientées objet au C.
- **1991** : Python est lancé par Guido van Rossum, offrant un langage à la fois puissant et facile à lire, qui gagne rapidement en popularité.

L'ère de l'Internet

- **1995** : Java est publié par Sun Microsystems, avec la promesse de "Write Once, Run Anywhere", gagnant rapidement en popularité pour les applications web et mobiles.
- **1995** : JavaScript est créé en seulement dix jours par Brendan Eich de Netscape Communications, pour permettre aux pages web d'être interactives.
- **1995** : PHP est créé par Rasmus Lerdorf pour le développement web, et devient un choix standard pour la création de sites dynamiques.

Nouveau millénaire, nouvelles tendances

- **2000s** : Avec l'essor d'Internet, JavaScript s'établit comme la pierre angulaire du développement web côté client, et des frameworks comme Angular, React et Vue.js voient le jour.
- **2009** : Le langage Go est introduit par Google, visant à combiner la simplicité de la programmation avec la performance des langages compilés.
- **2014** : Swift est lancé par Apple pour remplacer Objective-C avec un langage plus sûr et plus moderne pour le développement iOS et macOS.

Aujourd'hui et demain

- **2020s** : Les langages continuent d'évoluer avec les besoins des développeurs et des utilisateurs. Python et JavaScript restent parmi les langages les plus populaires, notamment pour l'apprentissage automatique, le développement web et les applications mobiles.
- **Tendance actuelle** : Les langages de programmation se concentrent de plus en plus sur la facilité de maintenance, la sécurité, et l'interopérabilité entre les systèmes.

Les langages de programmation ont parcouru un long chemin depuis les premiers jours de l'informatique, et l'histoire continue de s'écrire avec l'émergence de nouvelles technologies et paradigmes.

Assembleur



Le langage assembleur, souvent simplement appelé "assembleur", est un type de langage de programmation de bas niveau utilisé pour écrire des programmes qui sont très proches de l'instruction set architecture (ISA) d'un ordinateur. Chaque type de processeur a son propre langage assembleur spécifique, adapté à sa configuration de registres et à ses opérations.

L'histoire de l'assembleur remonte aux premiers jours de l'informatique, dans les années 1940 et 1950, lorsque les programmeurs écrivaient des instructions en langage machine sous forme de séries de chiffres. Cela était fastidieux et sujets à erreurs. Pour simplifier ce processus, les programmeurs ont commencé à utiliser des mnémoniques pour représenter les opérations, donnant naissance aux premiers assembleurs.

Les assembleurs ont permis aux programmeurs de travailler avec des codes plus lisibles, qui étaient ensuite convertis en langage machine par un programme appelé assembleur. Au fil du temps, les langages de haut niveau ont été développés pour offrir une abstraction plus élevée et

une meilleure portabilité, mais l'assembleur reste utilisé pour des tâches spécifiques nécessitant un contrôle direct et précis du matériel.

Avantages et inconvénients

Avantages

- **Contrôle total** : L'assembleur donne aux programmeurs un contrôle complet sur le matériel, permettant une optimisation fine des performances et de l'utilisation des ressources.
- **Efficient** : Les programmes écrits en assembleur peuvent être extrêmement rapides et légers, car il n'y a pas de surcharge liée à l'abstraction ou à l'interprétation.
- **Essentiel pour certains domaines** : L'assembleur est souvent utilisé dans le développement de firmware, de systèmes embarqués et de situations où la manipulation directe du matériel est nécessaire.

Inconvénients

- **Complexité** : Programmer en assembleur est complexe et exigeant, avec une courbe d'apprentissage raide, surtout pour ceux qui sont habitués aux langages de haut niveau.
- **Maintenance difficile** : Les programmes en assembleur peuvent être difficiles à maintenir et à comprendre, car ils manquent de structures de haut niveau comme les boucles et les fonctions.
- **Non portable** : Le code assembleur est spécifique à une architecture de processeur donnée et doit être réécrit pour fonctionner sur un matériel différent.
- **Développement plus lent** : Écrire en assembleur prend généralement plus de temps que d'utiliser des langages de haut niveau en raison de la nécessité de gérer manuellement les détails de bas niveau.
- **Possibilité d'erreurs** : Le risque d'erreurs est plus élevé en assembleur, car les programmeurs doivent gérer explicitement les registres, les adresses de mémoire et les conventions d'appel.

Bien que l'utilisation de l'assembleur ait diminué avec l'avènement des langages de haut niveau, il conserve une place importante dans la programmation système et est un outil précieux pour comprendre en profondeur le fonctionnement interne des ordinateurs.

Fortran



Fortran, un acronyme de "FORMula TRANslation", est un langage de programmation de haut niveau qui a révolutionné l'informatique en permettant une écriture de code plus accessible et plus proche du langage humain. Développé par IBM dans les années 1950, avec la première version (Fortran I) publiée en 1957, il a été conçu principalement pour les applications scientifiques et l'ingénierie, où des calculs complexes et des traitements de données de grande taille sont nécessaires.

Fortran a introduit de nombreuses innovations dans la programmation, telles que l'utilisation de boucles et de tests conditionnels, et il a établi de nombreux standards pour les langages de programmation ultérieurs. Au fil des décennies, Fortran a été mis à jour avec de nouvelles versions, dont Fortran II, Fortran IV, Fortran 77, Fortran 90/95, Fortran 2003, et Fortran 2008, chacune ajoutant de nouvelles fonctionnalités et améliorations pour répondre aux besoins changeants des scientifiques et des ingénieurs.

Avantages et inconvénients

Avantages

- **Optimisation pour les calculs** : Fortran est particulièrement efficace pour les calculs numériques et les opérations sur les tableaux, ce qui en fait un choix privilégié pour les applications de calcul scientifique.
- **Performance** : Les compilateurs Fortran sont réputés pour générer un code très optimisé, ce qui est crucial pour les applications de haute performance.
- **Ancienneté et fiabilité** : Fortran a une longue histoire d'utilisation dans des domaines critiques, ce qui a permis de tester et de fiabiliser le langage au fil des ans.
- **Portabilité** : Les programmes Fortran peuvent être facilement portés entre différentes plateformes et systèmes d'exploitation.

Inconvénients

- **Syntaxe obsolète** : Les versions plus anciennes de Fortran ont une syntaxe qui peut sembler dépassée par rapport aux langages de programmation modernes.
- **Moins intuitif** : Pour les nouveaux programmeurs, Fortran peut être moins intuitif et plus difficile à apprendre que des langages plus récents.
- **Écosystème limité** : Il y a moins de bibliothèques et d'outils modernes disponibles pour Fortran par rapport à des langages plus récents, ce qui peut limiter son utilisation dans certains domaines.
- **Communauté réduite** : La base d'utilisateurs de Fortran est plus petite que celle de langages plus modernes, ce qui signifie moins de support communautaire et moins de ressources d'apprentissage.
- **Applications limitées** : Bien que Fortran soit excellent pour les calculs scientifiques, il n'est pas aussi bien adapté aux applications de développement web, mobile ou de bureau.

Malgré certains de ses inconvénients, Fortran reste un langage important dans les domaines de la science et de l'ingénierie, où des calculs intensifs sont nécessaires, et il continue d'être utilisé et développé pour répondre aux besoins de ces communautés spécialisées.

C++



C++ est un langage de programmation orienté objet créé par Bjarne Stroustrup en tant qu'extension du langage C. Le développement de C++ a commencé en 1979 chez Bell Labs, Stroustrup cherchant à ajouter des fonctionnalités de programmation orientée objet à C sans sacrifier sa vitesse ou sa basse abstraction matérielle. La première version officielle, C++ 1.0, a été publiée en 1985, et la première norme internationale, ISO/IEC 14882, a été adoptée en 1998.

C++ a été conçu pour être un langage "hybride", permettant la programmation procédurale, orientée objet et même générique. Au fil des ans, il a été largement adopté pour le développement de logiciels systèmes, de jeux vidéo, de moteurs graphiques et d'applications nécessitant de hautes performances. Il continue d'être développé et mis à jour, avec des normes modernes comme C++11, C++14, C++17 et C++20, qui ont introduit de nombreuses nouvelles fonctionnalités et améliorations pour faciliter la programmation moderne.

Avantages et inconvénients

Avantages

- **Performance** : C++ est un langage compilé qui génère du code machine optimisé, offrant des performances élevées.
- **Contrôle du système** : C++ donne aux programmeurs un contrôle précis sur les ressources système et la gestion de la mémoire.
- **Compatibilité avec C** : C++ est compatible avec le langage C, permettant aux développeurs d'utiliser des bibliothèques C existantes.
- **Programmation multi-paradigme** : Supporte la programmation procédurale, orientée objet et générique.
- **Utilisation étendue** : C++ est utilisé dans des applications où les performances et la rapidité d'exécution sont critiques, comme les jeux vidéo, les applications de bureau, les systèmes embarqués et les applications temps réel.

Inconvénients

- **Complexité** : La richesse des fonctionnalités de C++ peut le rendre complexe et difficile à apprendre pour les débutants.
- **Gestion de la mémoire** : La gestion manuelle de la mémoire est puissante mais peut conduire à des erreurs difficiles à détecter, comme les fuites de mémoire et les dépassements de tampon.
- **Absence de vérification à l'exécution** : C++ manque de certaines vérifications à l'exécution présentes dans des langages plus modernes, ce qui peut conduire à des erreurs de sécurité.
- **Temps de compilation** : Les gros projets en C++ peuvent avoir des temps de compilation très longs comparés à d'autres langages.
- **Moins de fonctionnalités de haut niveau** : Par rapport à des langages comme Python ou Java, C++ offre moins de fonctionnalités de haut niveau intégrées, ce qui peut augmenter la charge de travail pour certaines tâches.

C++ demeure un choix privilégié pour le développement de logiciels qui nécessitent une gestion fine des ressources et des performances maximales, malgré sa courbe d'apprentissage abrupte et sa complexité accrue par rapport à des langages plus récents.

Python



Python est un langage de programmation interprété, interactif et orienté objet, créé par Guido van Rossum. Son développement a commencé à la fin des années 1980 et la première version (Python 1.0) a été publiée en 1991. L'objectif principal était de combler le fossé entre le shell Unix et le langage C, tout en offrant une syntaxe facile à lire et à apprendre.

Au fil des années, Python a évolué avec plusieurs versions majeures qui ont ajouté de nombreuses fonctionnalités et améliorations. Python 2.0, sorti en 2000, a introduit des fonctionnalités telles que la collecte de déchets pour la gestion automatique de la mémoire et le support Unicode. Python 3.0, sorti en 2008, était une refonte majeure conçue pour rectifier certaines erreurs de conception fondamentales, bien que cela ait introduit des problèmes de compatibilité avec les versions antérieures.

Python a gagné en popularité en raison de sa simplicité et de sa polyvalence, devenant un langage de choix dans divers domaines tels que le développement web, l'analyse de données, l'intelligence artificielle, l'enseignement et bien sûr, la robotique.

Avantages et inconvénients

Avantages

- **Facilité d'apprentissage** : Python a une syntaxe claire et une structure simple, ce qui le rend accessible aux débutants.
- **Polyvalence** : Il est utilisé dans de nombreux domaines, y compris le développement web, la science des données, l'automatisation, et plus encore.
- **Grande communauté** : La communauté Python est vaste et active, fournissant un excellent support et de nombreuses ressources d'apprentissage.
- **Bibliothèques étendues** : Python bénéficie d'un riche écosystème de bibliothèques et de frameworks qui facilitent le développement dans de nombreux domaines.
- **Productivité** : Sa syntaxe concise et son typage dynamique permettent de développer rapidement des prototypes et des applications.

Inconvénients

- **Performance** : En tant que langage interprété, Python est souvent plus lent que les langages compilés comme C ou Java.
- **Consommation de mémoire** : Python peut être plus gourmand en mémoire que d'autres langages, ce qui peut être un inconvénient pour les applications à ressources limitées.
- **Parallélisme** : En raison du Global Interpreter Lock (GIL), Python n'est pas idéal pour les tâches nécessitant un calcul parallèle intensif sur plusieurs threads.
- **Versionnage** : La transition de Python 2 à Python 3 a créé une fracture dans la communauté et des problèmes de compatibilité entre les versions.
- **Dépendances** : La gestion des dépendances et des environnements peut parfois être complexe, bien que des outils comme virtualenv et pip aient facilité ce processus.

Malgré ses inconvénients, Python reste l'un des langages de programmation les plus populaires et continue de croître grâce à sa facilité d'utilisation et à sa large applicabilité.

Java



Java est un langage de programmation orienté objet conçu pour être portable et sécurisé. Développé initialement par Sun Microsystems en 1991, il a été officiellement lancé en 1995. Java a été créé avec la vision "Write Once, Run Anywhere" (WORA), ce qui signifie que le code Java compilé (bytecode) peut s'exécuter sur n'importe quelle plateforme disposant d'une machine virtuelle Java (JVM) sans nécessiter de recompilation.

L'un des grands tournants pour Java a été son adoption par les navigateurs web comme l'un des premiers langages à permettre le développement d'applications web interactives via des applets. Au fil des ans, Java s'est solidifié comme un choix majeur pour le développement d'applications d'entreprise, d'applications serveur, de logiciels mobiles (notamment Android), et dans le monde des systèmes embarqués et de l'Internet des Objets (IoT).

Java a évolué à travers de nombreuses versions, ajoutant constamment de nouvelles fonctionnalités pour améliorer la performance, la sécurité et la fonctionnalité du langage. Oracle Corporation a acquis Sun Microsystems en 2010, prenant ainsi la responsabilité de la poursuite du

développement de Java.

Avantages et inconvénients

Avantages

- **Portabilité** : Grâce à la JVM, le code Java peut être exécuté sur n'importe quel système d'exploitation sans modification, ce qui le rend extrêmement portable.
- **Performance** : Avec l'introduction de la compilation Just-In-Time (JIT), Java offre des performances proches de celles des langages compilés nativement.
- **Orienté objet** : Java encourage une conception propre et modulaire du code, facilitant la maintenance et l'évolutivité des applications.
- **Sécurité** : Java est conçu avec un modèle de sécurité intégré qui limite les classes auxquelles un processus peut accéder.
- **Multithreading** : Java fournit une gestion intégrée du multithreading, permettant de créer des applications hautement interactives et réactives.

Inconvénients

- **Consommation de mémoire** : Les applications Java peuvent être plus gourmandes en mémoire par rapport aux applications écrites dans des langages comme C ou C++.
- **Vitesse de démarrage** : Les applications Java peuvent prendre plus de temps à démarrer en raison du temps nécessaire pour lancer la JVM et effectuer la compilation JIT.
- **Interface utilisateur** : Les applications de bureau Java ont souvent été critiquées pour leur apparence et leur convivialité, qui peuvent être inférieures à celles des applications natives.
- **Verbosité** : Java est souvent critiqué pour sa verbosité, ce qui peut rendre le code plus long et plus fastidieux à écrire par rapport à des langages plus concis comme Python.

Java reste l'un des langages de programmation les plus utilisés et continue d'être un choix privilégié pour de nombreux développeurs et entreprises à travers le monde, en raison de sa robustesse, de sa sécurité et de sa portabilité.

JavaScript



JavaScript, souvent abrégé en JS, est un langage de programmation dynamique qui a été créé en 1995 par Brendan Eich, alors ingénieur chez Netscape Communications. À l'origine, il a été développé sous le nom de Mocha, puis renommé en LiveScript, et finalement en JavaScript. Le but était de permettre aux développeurs web de rendre les pages HTML interactives et de réaliser des tâches côté client sans avoir à recharger la page web.

JavaScript a rapidement gagné en popularité et est devenu un composant standard des navigateurs web. Avec l'introduction d'AJAX (Asynchronous JavaScript and XML) au début des années 2000, JavaScript a permis de créer des applications web plus dynamiques et réactives. Plus récemment, l'écosystème JavaScript s'est considérablement élargi avec l'arrivée de Node.js, qui permet d'exécuter JavaScript côté serveur, et de divers frameworks et bibliothèques tels que React, Angular et Vue.js, qui facilitent le développement d'applications web complexes.

Avantages et inconvénients

Avantages

- **Ubiquité** : JavaScript est supporté par tous les navigateurs web modernes sans avoir besoin de plugins supplémentaires.
- **Full-stack** : Avec Node.js, les développeurs peuvent utiliser JavaScript à la fois pour le développement côté client et serveur, permettant une stack technologique unifiée.
- **Asynchrone** : JavaScript gère nativement les opérations asynchrones, ce qui est idéal pour les tâches non bloquantes, comme les requêtes réseau.
- **Communauté et écosystème** : Une vaste communauté de développeurs et un large éventail de bibliothèques et de frameworks sont disponibles, ce qui facilite le développement rapide d'applications.
- **Flexibilité** : JavaScript est un langage flexible avec un typage dynamique et des fonctions de première classe, permettant une grande variété de paradigmes de programmation.

Inconvénients

- **Sécurité** : Étant donné que le code JavaScript s'exécute généralement côté client, il peut être exposé à des failles de sécurité et à des attaques malveillantes.
- **Gestion de la mémoire** : JavaScript ne permet pas de contrôler directement la mémoire, ce qui peut conduire à des problèmes de performance dans les applications de grande envergure.
- **Compatibilité entre navigateurs** : Bien qu'elle se soit améliorée, la compatibilité entre différents navigateurs peut encore poser problème, nécessitant parfois des polyfills ou du code spécifique au navigateur.
- **Performance** : Pour des tâches très exigeantes en termes de calcul, JavaScript peut être moins performant que d'autres langages compilés ou optimisés pour de telles opérations.
- **Complexité** : La nature asynchrone de JavaScript et son modèle de concurrence basé sur des événements peuvent rendre le code difficile à comprendre et à déboguer pour les développeurs moins expérimentés.

JavaScript continue d'être un pilier du développement web et son rôle s'étend désormais au-delà des navigateurs, avec des applications dans le développement de serveurs, d'applications mobiles et de bureau, et même dans le domaine de l'Internet des Objets (IoT).

PHP



PHP, qui signifie initialement "Personal Home Page", est un langage de script open source côté serveur conçu pour le développement web. Créé par Rasmus Lerdorf en 1994, PHP a été développé pour faciliter la création de pages web dynamiques. Lerdorf a écrit plusieurs scripts en C pour maintenir son CV en ligne et pour suivre le trafic sur sa page web, et il a fini par les combiner pour créer PHP/FI (Forms Interpreter).

Au fil des années, PHP a évolué de cet ensemble de scripts à un langage de programmation complet avec PHP 3, co-développé par Lerdorf et deux autres développeurs, Zeev Suraski et Andi Gutmans, qui a introduit le support des bases de données et une meilleure intégration avec les serveurs web. PHP 4 et PHP 5 ont introduit respectivement le Zend Engine I et II, améliorant les performances et ajoutant des caractéristiques orientées objet.

PHP est devenu l'un des langages de programmation les plus utilisés sur le web, avec une large base d'utilisateurs et de puissants frameworks tels que Laravel, Symfony et CodeIgniter qui facilitent le développement d'applications web robustes.

Avantages et inconvénients

Avantages

- **Facilité d'utilisation** : PHP est connu pour sa syntaxe simple et ses vastes capacités de documentation, ce qui le rend accessible aux débutants.
- **Intégration facile** : PHP est bien intégré avec les serveurs web et les systèmes de gestion de bases de données les plus populaires.
- **Large communauté** : Une grande communauté de développeurs signifie beaucoup de ressources, de frameworks, de bibliothèques et de support.

- **Rapide pour le développement web** : PHP est conçu spécifiquement pour le développement web, ce qui rend la création de sites web dynamiques rapide et relativement facile.
- **Hébergement économique** : La plupart des fournisseurs d'hébergement web supportent PHP, souvent à un coût faible ou nul.

Inconvénients

- **Incohérences dans le langage** : PHP a été critiqué pour avoir une syntaxe incohérente et une organisation de fonction peu intuitive.
- **Performance** : Bien que PHP 7 ait considérablement amélioré la vitesse, PHP peut être plus lent que certains autres langages de programmation, en particulier pour les applications très performantes.
- **Sécurité** : PHP a une histoire de vulnérabilités de sécurité, bien que cela puisse souvent être attribué à des pratiques de programmation médiocres plutôt qu'au langage lui-même.
- **Réputation** : PHP souffre parfois d'une réputation négative, en partie à cause du code de mauvaise qualité écrit par des développeurs inexpérimentés.
- **Concurrence des technologies modernes** : Avec l'émergence de plateformes côté serveur basées sur JavaScript comme Node.js, PHP fait face à une concurrence accrue.

Malgré ses inconvénients, PHP continue d'être largement utilisé pour le développement web, en particulier pour les projets qui nécessitent un déploiement rapide et peu coûteux.

3.3 Syntaxe et paradigmes de programmation

Syntaxe et sémantique

La syntaxe d'un langage de programmation est l'ensemble des règles qui définissent la structure du code. La sémantique, quant à elle, est le sens des instructions. Une syntaxe correcte sans une sémantique appropriée ne donnera pas le résultat attendu.

Prenons l'exemple d'une simple instruction en Python :

```
print("Bonjour le monde!")
```

PYTHON

La syntaxe ici inclut l'utilisation de parenthèses et de guillemets. La sémantique est que cette instruction affiche le texte "Bonjour le monde!" à l'écran.

Insérer une image illustrant un code avec une syntaxe correcte mais une sémantique erronée, par exemple une boucle qui ne se termine jamais.

Paradigmes de programmation

Un paradigme de programmation est une manière d'aborder la programmation en suivant certaines philosophies et modèles. Les principaux paradigmes sont :

- **Programmation impérative** : centrée sur la description détaillée des étapes que l'ordinateur doit suivre pour atteindre un état désiré.
- **Programmation orientée objet (POO)** : basée sur la définition d'objets qui contiennent à la fois des données et des méthodes pour manipuler ces données.
- **Programmation fonctionnelle** : se concentre sur l'utilisation de fonctions pour traiter des données et éviter les changements d'état et les données mutables.

En Python, on peut utiliser ces différents paradigmes :

PYTHON

```
# Impératif
a = 1
b = 2
somme = a + b
print(somme)

# Orienté objet
class Additionneur:
    def __init__(self, a, b):
        self.a = a
        self.b = b
    def calculer(self):
        return self.a + self.b

additionneur = Additionneur(1, 2)
print(additionneur.calculer())

# Fonctionnel
def ajouter(a, b):
    return a + b

print(ajouter(1, 2))
```

Insérer une image illustrant les concepts de la POO, comme les classes et les objets.

Chaque paradigme a ses avantages et est plus adapté à certains types de problèmes. Dans ce cours, nous utiliserons principalement la programmation impérative et orientée objet en Python pour construire des programmes et contrôler des robots.