République Tunisienne
Ministère de l'Education

CURRICULUM
D'INFORMATIQUE
SPECIFIQUE

SECTIONS : SCIENCES EXPERIMENTALES,
SCIENCES TECHNIQUES
ET MATHEMATIQUES

Septembre 2021

NIVEAU : 2EME ANNEE

Aide pédagogique 2021-2022

Domaine d'apprentissage	Savoirs associés	Pistes pédagogiques et directives
Pensée Computationnelle et programmation	• **Identifier les étapes de résolution d'un problème.** ○ Dégager les éléments essentiels permettant la résolution d'un problème. • **Élaborer des solutions sous forme d'algorithmes.** ○ Utiliser des structures de données à bon escient. ○ Utiliser les structures de contrôle adéquates pour résoudre un problème. ○ Traiter essentiellement les notions suivantes : ▪ Les objets (**constantes** et **variables**) ▪ Les types de données (**entier, réel, caractère, booléen, chaîne**). ▪ Les structures simples.	• Etablir des liens et trouver des fils conducteurs entre les différents domaines d'apprentissage rompant ainsi avec l'aspect linéaire du programme. • Il est préconisé de présenter les savoirs associés à travers des projets, des mini-projets ou des activités ayant du sens pour l'apprenant. • Favoriser l'investigation, le questionnement, l'apprentissage expérientiel, l'apprentissage par problème. etc. • Il est fortement recommandé d'opter pour une démarche de création au cours de laquelle les apprenants développent leur autonomie, leur créativité et leur imagination, mais aussi le sens du travail collaboratif. • Donner du sens aux activités, les diversifier et opter pour une démarche interdisciplinaire permettant le décloisonnement entre les divers champs d'apprentissages et l'ouverture de l'informatique sur les autres disciplines. • Favoriser l'exploitation des ressources en ligne et développer la communication. • Inciter à l'innovation et motiver les apprenants pour la créativité. • Inciter les apprenants à faire des échanges autour des solutions proposées et de les partager en ligne. • Il est pertinent de : ○ familiariser l'apprenant à formuler sous forme d'actions des solutions à des problèmes puisés de son vécu. On pourra exprimer ces solutions à

	▪ La structure de contrôle conditionnelle (**Si … Alors ; Si … Sinon**). ▪ La structure de contrôle répétitive (**Pour**). • **Exploiter un environnement de programmation** ○ Implémenter un algorithme en utilisant un langage de programmation. ○ Tester et valider un programme.	l'aide d'un schéma, d'un organigramme, d'une carte heuristique, d'un pseudocode, etc. ○ inviter les apprenants à déterminer les entrées, les sorties et les traitements. ○ inciter les apprenants à vérifier la validité d'une solution donnée par rapport à l'énoncé d'un problème. • Il est recommandé d'inciter les apprenants à : ○ étudier quelques séquences algorithmiques (décrire, comparer, déterminer le rôle, etc.). ○ modifier des algorithmes existants pour changer leurs comportements. ○ corriger les erreurs de logique dans une séquence algorithmique afin de parvenir aux résultats souhaités. ○ évaluer différentes solutions algorithmiques d'un même problème donné. • Il est préconisé : ○ de familiariser l'apprenant avec un environnement de programmation. ○ d'inciter l'apprenant à : • réutiliser des codes sources existants. • modifier un programme existant pour obtenir un résultat différent. • écrire un programme pour résoudre un problème. • Toutes les solutions des problèmes sont implémentées via le langage de programmation **Python**.

Systèmes et technologies Internet	• **Prendre conscience de l'intérêt de la robotique** - Définir la robotique. - Identifier des domaines d'application de la robotique. • **Piloter un objet connecté** - Connaitre les interfaces de l'objet à connecter. - Savoir connecter un objet à l'ordinateur - Programmer des objets simples virtuels ou réels pour réaliser différentes tâches d'une façon innovante.	• Il est nécessaire de présenter la robotique et d'expliquer ses fondements en s'appuyant sur des séquences vidéo, des ressources numériques, des études de cas, etc. • L'apprenant n'est pas appelé à développer une application de commande. • Utiliser **Micro-Python / Arduino** pour programmer la carte. • Traiter les activités telles que : ○ Faire clignoter une diode Led. ○ Feu de carrefour (rouge, vert, orangé).

NIVEAU : 3EME ANNEE

Aide pédagogique 2021-2022

Domaine d'apprentissage	Savoirs associés	Pistes pédagogiques et directives
Pensée Computationnelle et programmation	• **Utiliser les structures algorithmiques adéquates pour résoudre un problème.** ○ Les objets (**constantes et variables**). ○ Les types de données (**entier, réel, caractère, booléen, chaîne de caractères**). ○ Les structures simples. ○ Les structures de contrôle (**Si, Selon, Pour, Tant Que et Répéter**). ○ Les tableaux à une dimension. • **Elaborer des solutions algorithmiques modulaires.** ○ Analyser un problème. ○ Acquérir la capacité de décomposer un problème en modules. ○ Identifier les éléments principaux d'un module (**paramètres, résultat, type, portée des objets**).	• Exprimer les solutions, selon les besoins, sous forme d'un organigramme, d'une carte mentale, d'un pseudocode, etc. • Inciter les apprenants à choisir les structures de données et les structures de contrôle adéquates. • Concevoir des solutions algorithmiques. • Choisir des exemples concrets pour montrer les avantages de la décomposition modulaire. • Argumenter et justifier les choix de la modularité. • Inciter les apprenants à écrire des solutions modulaires. • Il est conseillé d'habituer les apprenants à commenter les solutions proposées. • Utiliser des modules prédéfinis et implémenter des modules personnels. • Décrire correctement les entrées / les sorties et le rôle de chaque module.
	• **Utiliser un langage de programmation pour implémenter une solution.** ○ Analyser un programme existant et lui apporter des modifications pour améliorer sa fonctionnalité. ○ Implémenter un algorithme en un programme.	• Les solutions des problèmes sont implémentées via le langage de programmation **Python.** • L'apprentissage se fait à travers un projet ou des mini projets faisant appel essentiellement aux traitements suivants :

	○ Écrire un programme pour résoudre un problème. ○ Tester, corriger, améliorer et valider un programme.	○ Calculs arithmétiques tels que : PGCD, PPCM, nombres premiers, décomposition en facteurs premiers. ○ Tri d'un tableau (une méthode de tri). ○ Recherche d'un élément dans un tableau.
Internet et Nouvelles technologies	• **Prendre conscience de l'intérêt de la robotique** - Définir la robotique. - Identifier des domaines d'application de la robotique. • **Piloter un objet connecté** - Connaitre les interfaces de l'objet à connecter. - Savoir connecter un objet à l'ordinateur - Programmer des objets simples virtuels ou réels pour réaliser différentes tâches d'une façon innovante.	• Il est nécessaire de présenter la robotique et d'expliquer ses fondements en s'appuyant sur des séquences vidéo, des ressources numériques, des études de cas, etc. • L'apprenant n'est pas appelé à développer une application de commande. • Utiliser **Micro-Python** / **Arduino** pour programmer la carte. • Traiter les activités telles que : ○ Faire clignoter une diode Led. ○ Feu de carrefour (rouge, vert, orangé).

NIVEAU : 4EME ANNEE

Les apprentissages à développer en termes de compétences

Domaine d'apprentissage spécifique	Compétences disciplinaires à développer	Savoirs associés (savoir, savoir-faire et savoir-être)	Compétences de vie visées
Pensée Computationnelle et programmation	• *Elaborer des solutions algorithmiques pour résoudre un problème.* • *Exploiter un environnement de programmation pour implémenter une solution.*	• *Utiliser les nouvelles structures algorithmiques pour résoudre un problème.* • *Utiliser un langage de programmation pour développer une solution.* • *Mobiliser les compétences développées dans des mini-projets concrets et variés.*	*La résolution de problèmes* *L'auto-évaluation* *La communication et la collaboration* *L'innovation et la créativité*

NIVEAU : 4EME ANNEE

Aide pédagogique 2021-2022

Domaine d'apprentissage	Savoirs associés	Pistes pédagogiques et directives
Pensée Computationnelle et programmation	**Utiliser les structures algorithmiques pour résoudre un problème** • ***Utiliser les structures algorithmiques adéquates pour résoudre un problème.*** ○ *Les types de données standards.* ▪ **Entier, réel, booléen, caractère, chaîne.** ○ *Les structures de contrôle conditionnelles.* ▪ **si … alors** ▪ **si … sinon** ▪ **selon**	• Il est important que l'apprenant **conserve une trace écrite du travail réalisé en classe**. Il appartient à l'enseignant de choisir le support le plus adapté à ses élèves. • Il est préconisé de présenter le contenu à enseigner via des **activités** et/ou des **mini-projets** afin de stimuler l'activité, la collaboration et la créativité chez l'apprenant. • Il est judicieux de proposer un contenu motivant ayant un sens pour l'apprenant (situation problème, jeux, simulation, …) et favorisant ainsi l'aspect interdisciplinaire. • Favoriser l'exploitation des ressources en ligne. • Inviter les apprenants à participer à des communautés de développement et de partage de solutions pour une autoformation, pour trouver des réponses à des questionnements ou pour l'enrichir avec leurs productions. • Exprimer les solutions sous forme d'**un pseudocode.** • Habituer les apprenants à dégager, à partir d'un énoncé, les mots clés permettant de dégager les tâches à réaliser et de déterminer les entrées, les sorties et les pistes des traitements nécessaires pour établir une solution à un problème donné. • Dégager les éléments essentiels pour la résolution d'un problème (structures algorithmiques, types de données, traitements, etc.). • Analyser une solution existante et identifier les rôles de différentes structures utilisées. • Inciter les apprenants à expliquer et à argumenter une séquence d'instructions afin de comprendre le traitement.

	▪ **des structures imbriquées** ○ *Les structures de contrôle répétitives.* ▪ **pour … faire** ▪ **répéter ... jusqu'à** ▪ **tant que ...faire** ▪ **des boucles imbriquées** ○ *Les tableaux à une dimension* • ***Elaborer des solutions algorithmiques modulaires.*** ○ *Analyser et modéliser un problème.* ○ *Acquérir la capacité de décomposer un problème en sous-problèmes.* ○ *Identifier les éléments principaux d'un module (l'entête, paramètres,*	• Inviter les apprenants à identifier, pour un problème donné, une solution parmi plusieurs programmes proposés. • Distinguer les usages et les particularités de chaque type de données. • Il est possible de faire appel à des séquences vidéo, des sites internet, divers documents ou d'une situation réelle pour dégager l'utilité de l'utilisation des structures algorithmiques. • Inviter les apprenants à concevoir des solutions algorithmiques. • Inviter les apprenants à corriger une séquence d'instructions ou une solution erronée. • Avantager les échanges et les discussions autour des solutions proposées. • Apporter les modifications nécessaires à un programme existant pour obtenir un résultat différent. • Inciter les apprenants à identifier et à choisir les structures de données et les structures de contrôle adéquates. • Écrire une solution utilisant les structures conditionnelles. • Écrire une solution utilisant les structures itératives. • Argumenter et justifier les choix des différentes structures. • Il est conseillé d'habituer les apprenants à **commenter** les solutions proposées. • L'apprentissage se fait à travers la résolution de problèmes **élémentaires**, par exemple : ○ **traitement simples sur les chaines.** ○ **traitement simples sur les objets numérique.** ○ **somme d'une suite arithmétique.** ○ **recherche d'un élément.** ○ **recherche de nombres premiers.** • Analyser et modéliser un problème, le découper en sous-problèmes plus faciles à résoudre.

	résultat, type du module, portée des objets, appel).	• Choisir des exemples concrets pour montrer les avantages de la décomposition modulaire. • Élaborer des solutions algorithmiques modulaires. • Inciter les apprenants à écrire des solutions modulaires. • Argumenter et justifier les choix de la modularité. • Utiliser diverses techniques de décomposition (Ascendante, descendante, etc.). • Utiliser des modules prédéfinis et implémenter des modules personnels. • Ecrire des solutions modulaires à un problème donné et réutiliser des modules déjà programmés. • Décrire correctement les entrées / les sorties et le rôle de chaque module. • Choisir les données appropriées pour **tester** chaque module. • Inscrire le développement des programmes dans un travail collaboratif. • L'apprentissage se fait à travers la résolution de problèmes modulaires. • Comme exemples de projets ou de mini projets, on peut citer : o **somme des carrés des premiers entiers naturels** o **tri d'un tableau (au moins deux différents principes de tri).** o **calcul de combinaison.** o **triangle de Pascal** o **somme des premiers entiers naturels** o **recherche d'un élément dans un tableau.** o **calcul arithmétique (PGCD, PPCM, nombres premiers, décomposition en facteurs premiers, etc.)** • Collaborer efficacement au sein d'une équipe dans le cadre d'un projet.

	Exploiter un langage de programmation pour implémenter une solution ○ *Implémenter un algorithme en un programme.* ○ *Écrire un programme pour résoudre un problème.* ▪ **Les types de données standards.** ▪ **Les structures de contrôle.** ▪ **Les tableaux à une dimension.** ○ *Tester, corriger, améliorer et valider un programme sur machine.* ○ *Utiliser les fonctions prédéfinies de base*	• Les solutions des problèmes sont implémentées cette année via le langage de programmation **Python**. • L'apprentissage est axé principalement sur la pratique. • Se familiariser avec l'environnement de programmation. • Écrire un programme en respectant les contraintes de la syntaxe. • Repérer l'erreur de syntaxe dans un programme existant. • Tester, corriger et modifier un programme. • Utiliser ses capacités de raisonnement, ses connaissances sur le langage de programmation, ses savoirs faire et à des outils variés pour améliorer sa solution. • Savoir s'autoévaluer et être capable de décrire ses intérêts, ses compétences et ses acquis. • Amener l'apprenant à effectuer une recherche et à présenter les structures de données et de contrôle en python. • Il est nécessaire d'habituer les apprenants à exploiter à bon escient les structures de données (Objets et types). • Habituer les apprenants à appliquer les bonnes pratiques de programmation (nomenclature des objets, indentation de la solution, commentaire, etc.). • Lors de l'implémentation, l'apprenant devrait se conformer aux recommandations de l'enseignant quant à l'usage des méthodes, étapes, formules… • Comme exemples de projets ou de mini projets, on peut citer : ○ **nombre paire ou impaire.** ○ **résolution d'une équation du second degré.** ○ **nombre d'occurrences du maximum dans un tableau.** ○ **nombres amis.** ○ **nombres d'Amstrong.**

	▪ **sqrt, abs, around, trunc, randint, floor, chr, ord, int** ▪ **upper, lower, find, replace, count, len, val, str, isalpha, isdigit, eval, slice, find** ○ *Concevoir une interface graphique pour développer des applications simples.* ▪ **Utiliser les objets graphiques (widgets) les plus usuels (boîte de dialogue, zone texte, label, bouton, bouton radio, liste, liste déroulante, case à cocher).**	• L'interface graphique est utilisée avec le langage de programmation Python pour développer des applications simples. • Installer le logiciel **Qtdesigner** et les bibliothèques nécessaires. • Concevoir et utiliser une interface utilisateur graphique (GUI) • La découverte d'une interface graphique peut se faire à partir d'une application existante. • Initier les apprenants au principe de la programmation événementielle. • La conception d'une interface graphique peut se faire en utilisant la technique « Glisser–Déposer » et la programmation des objets se fait à l'aide du langage python. • Amener les apprenants à importer, installer et utiliser des bibliothèques. • Motiver les apprenants par la création des jeux. • Comme exemples de projets ou de mini projets, on peut citer : ○ **convertisseur de devises.** ○ **convertisseur de bases.** ○ **message en morse.** ○ **réalisation d'une calculette arithmétique.** • L'usage de l'interface graphique devrait être fait dans le but d'améliorer l'ergonomie de quelques programmes, mais en aucun cas un objet principal d'apprentissage. • L'ergonomie concerne les composants graphiques simples d'une fenêtre permettant la saisie des données et l'affichage des résultats (bouton, label …).