République Tunisienne
Ministère de l'Education

PROGRAMME D'INFORMATIQUE

SPECIFIQUE

SECTION : SCIENCES DE L'INFORMATIQUE

Septembre 2022

NIVEAU : 2EME ANNEE

Matière : Informatique

Aide pédagogique 2022-2023

Domaines d'apprentissage	Savoirs associés détaillés	Pistes pédagogiques et directives
Pensée computationnelle et programmation	• Reconnaître les phases de résolution d'un problème. - Lire et comprendre l'énoncé d'un problème afin de dégager les tâches à réaliser. - Dégager les éléments essentiels pour la résolution (les entrées, les sorties et les traitements). - Élaborer une solution sous forme d'un algorithme. - Écrire et exécuter le programme solution sur ordinateur. - Apporter des modifications à la solution (actions correctives, actions évolutives). • Décomposer un problème en modules. - Identifier des sous-problèmes pertinents (modules). - Identifier les éléments principaux d'un module (Type, paramètres, résultat, etc.). - Acquérir la capacité de décomposer un problème en sous-problèmes : décomposition logique. • Exploiter des concepts algorithmiques pour résoudre des problèmes. - Utiliser des structures de données à bon escient : • Dégager les objets nécessaires (variables/constantes) pour résoudre un problème.	• Il est possible de faire appel à des séquences vidéo, des sites internet, divers documents ou d'une situation réelle, … pour dégager les phases de résolution d'un problème. • Il est recommandé d'utiliser des exemples concrets pour montrer les avantages de la décomposition (meilleure lisibilité, diminution de risque d'erreurs, réutilisation de modules dans un ou plusieurs algorithmes, simplicité de l'entretien, favorisation de travail en équipe). • Chaque niveau de décomposition est suivi par l'élicitation (valorisation, argumentation, justification) de sous problèmes. • L'initiation à l'algorithmique peut se faire à partir d'un algorithme existant

	• Distinguer les usages et les particularités de chaque type de données, afin d'utiliser le plus adapté pour déclarer un objet nécessaire dans la résolution d'un problème donné. - Utiliser les structures de contrôle adéquates pour résoudre un problème. • Utiliser les structures simples pour lire des données, pour afficher des informations et pour attribuer une valeur à une variable. • Utiliser les structures conditionnelles pour effectuer des choix en fonction des circonstances. • Utiliser les structures répétitives pour répéter un ensemble d'instructions autant de fois que nécessaire. • Utiliser un langage de programmation pour implémenter une solution. - Traduire un algorithme en un programme exécutable. - Écrire un programme pour résoudre un problème. • Tester une solution implémentée afin de répondre à un besoin spécifique. - Exécuter une solution implémentée. - Modifier un code de programmation existant pour changer le comportement d'un programme.	(structure d'un algorithme et tournage à la main). • Il est nécessaire d'habituer les apprenants à exploiter à bon escient les structures de données (Objets et types) et les structures de contrôle lors de la résolution d'un problème (nombre de variables, nombre d'instructions, structure de contrôle adéquate, etc.) • Les types de données à utiliser sont essentiellement : Entier (octet, Entier, Entier long), Réel (Réel, Réel double) Booléen, Caractère, Chaîne de caractères et Tableau **à une dimension**. • On pourra utiliser des outils d'exécution d'algorithmes tels que "Algobox", "Larp", etc. • Inciter les apprenants à comparer différents algorithmes pouvant résoudre le même problème. • Toutes les solutions des problèmes sont implémentées via le langage de programmation **Python**. • On pourra utiliser des outils tels que Trinket.io et Pencilcode.net • L'initiation à l'utilisation du langage peut se faire à partir d'un programme existant

		(structure d'un programme, exécution et exploration du code). • Il est possible de traduire un algorithme existant en un programme. • Il est utile d'inciter les apprenants à analyser un programme exécutable afin de comprendre les traitements. • Se servir de dispositifs ou de robots pour appliquer des notions de programmation, en mettant à profit différents outils et langages de programmation. • Ecrire un programme en Micro-Python ou Arduino pour programmer une carte Esp32 afin de réaliser différentes tâches. • Il est essentiel d'habituer les apprenants à commenter les solutions.

Recommandations générales

- Avantager les échanges et les discussions autour des solutions proposées.
- Etablir des liens et trouver des fils conducteurs entre les différents domaines d'apprentissage rompant ainsi avec l'aspect linéaire de sa mise en œuvre.
- Il est préconisé de présenter le contenu à enseigner via des projets, des mini-projets ou des activités, ayant un sens pour l'apprenant (jeux, simulation, ...) et stimulant chez lui l'activité, la collaboration et la créativité ; tout en favorisant l'aspect interdisciplinaire.
- L'apprentissage est axé principalement sur la pratique.
- Il est recommandé de consulter des communautés de développement et de partager des solutions (algorithmes ou programmes) dans des espaces de partage créés pour l'échange et l'apprentissage.
- Favoriser l'exploitation des ressources en ligne.
- 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 recommandé d'aborder des problèmes et systématiser leurs résolutions en se basant sur les quatre composantes de la pensée computationnelle (décomposition, reconnaissance de modèles ou de formes, abstraction et algorithme).

Domaine d'apprentissage	Savoirs associés	Pistes pédagogiques et directives
Pensée Computationnelle et programmation	• Exploiter des concepts algorithmiques avancés pour résoudre des problèmes. - Lire et comprendre l'énoncé d'un problème afin de dégager les tâches à réaliser. - Dégager les éléments essentiels pour la résolution d'un problème (structures et types de données, traitements). - Distinguer les usages et les particularités de chaque type de données, afin d'utiliser le plus adapté pour déclarer un objet nécessaire dans la résolution d'un problème donné. - Utiliser des structures de données avancées pour résoudre un problème (Tableau à deux dimensions, Enregistrement ~~et Fichier~~). - Apporter les modifications nécessaires à une solution pour répondre à un besoin.	- 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 de données avancées. - Il est nécessaire d'habituer les apprenants à exploiter à bon escient les structures de données avancées (Objets et types). - habituer les apprenants à appliquer les bonnes pratiques de programmation (nomenclature des objets, commentaire, etc). - L'apprentissage se fait à travers un projet ou des mini projets faisant appel essentiellement aux traitements suivants : - Tri d'un tableau (tri par sélection et tri à bulles) - Recherche d'un élément dans un tableau - (séquentielle et dichotomique) - Calcul arithmétique (PGCD, PPCM, nombres premiers, décomposition en facteurs premiers, etc.)

	- Exploiter des concepts algorithmiques avancés pour résoudre des problèmes d'optimisation et d'approximation • Utiliser un environnement de programmation pour implémenter une solution. - Implémenter un algorithme en un programme exécutable. - Écrire un programme pour résoudre un problème. - Concevoir une interface graphique pour développer des applications simples.	- ~~Traitements sur les fichiers textes et les fichiers typés.~~ - *Des problèmes d'optimisations* - *La recherche du zéro d'une fonction (f(x)=0)* - *La recherche du point fixe d'une fonction (f(x)=x)* - *Des méthodes de calcul d'une valeur approchée de constantes connues (π, e, …)* - Tester le programme solution sur ordinateur. - Commenter une solution. - Apporter des modifications à une solution (actions correctives, actions évolutives). - Il est utile d'inciter les apprenants à analyser un programme exécutable afin de comprendre ses traitements. - Le langage adopté est Python. - La découverte d'une interface graphique peut se faire à partir d'une application existante (fenêtre, zone de texte (TextEdit, Line Edit), bouton (Push Button), bouton radio (Radio Button), liste déroulante (Combo Box), case à cocher (Check Box), étiquette (label)). - La conception d'une interface graphique se fait en utilisant la technique « Glisser–Déposer » (Drag & drop).

		- Utiliser Qtdesigner comme outil de création d'interfaces graphiques. - ~~Il est recommandé d'utiliser des fichiers pour transférer et récupérer des informations.~~ - Tenir compte des contrôles de saisie des champs d'une interface graphique en affichant des messages d'erreurs. - L'apprentissage est axé principalement sur la pratique. - ~~Comme exemples de projets ou de mini projets, on peut citer :~~ o ~~Calculatrice arithmétique~~ o ~~Dictionnaire ou glossaire personnel ordonné selon un critère donné (consultation, ajout, modification, suppression dans un fichier texte ou un fichier typé).~~ o ~~Gestion d'une compétition sportive (bac sport, patinage artistique, course de relais, épreuves combinées, etc.).~~

Recommandations générales
- Avantager les échanges et les discussions autour des solutions proposées. - Établir des liens et trouver des fils conducteurs entre les différents domaines d'apprentissage rompant ainsi avec l'aspect linéaire. - Il est préconisé de présenter le contenu à enseigner via des projets ou de mini-projets afin de stimuler l'activité, la collaboration et la créativité chez l'apprenant. - Il est judicieux de proposer des projets, des mini-projets ou des activités utiles, motivants, ayant un sens pour l'apprenant (jeux, simulation, ...) et favorisant ainsi l'aspect interdisciplinaire. - L'apprentissage est axé principalement sur la pratique. - 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 de données avancées. - Il est recommandé de consulter des communautés de développement et de partager des solutions (algorithmes ou programmes) dans des espaces de partage créés pour l'échange et l'apprentissage. - Favoriser l'exploitation des ressources en ligne. - 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 recommandé d'aborder des problèmes et systématiser leurs résolutions en se basant sur les quatre composantes de la pensée computationnelle (décomposition, reconnaissance de modèles ou de formes, abstraction et algorithme). - Le langage adopté est Python.

Domaines d'apprentissage	Savoirs associés	Pistes pédagogiques et directives
Pensée computationnelle et programmation	• ***Exploiter des concepts algorithmiques avancés pour résoudre des problèmes :*** ○ *Faisant appel à des structures de données* ○ *Faisant appel à des méthodes de tri* ○ *Faisant appel à la récursivité*	- Rappeler les notions de tableau à deux dimensions, enregistrement, fichier texte et fichier typé à travers la résolution de problème exploitant ces structures de données. - Des traitements sur les fichiers tels que le tri, l'insertion d'un élément, la suppression d'un élément, le décalage… doivent être effectués en mémoire centrale. - Rappeler les deux méthodes de tri : Le tri par sélection et le tri à bulles. - Traiter la méthode de tri par insertion et la méthode de tri shell - On pourra traiter d'autres algorithmes de tri (tri par création, tri par comptage, ...). - Montrer, quand c'est possible, le passage d'une formulation itérative à une formulation récursive. - Ne traiter que le cas de récursivité simple (ni croisée, ni indirecte) sur des problèmes naturellement récursifs (factorielle, palindrome, PGCD …)

	○ *Récurrents et arithmétiques*	- On traitera divers problèmes en axant sur la relation de récurrence d'ordre 2 et plus (suites, triangle de pascal, le nombre d'or, …) - On traitera essentiellement : - Calcul du factoriel, PGCD, PPCM, nombre premier, décomposition en facteurs premiers. - La suite de Fibonacci - Les conversions entre bases de numération - Les calculs de C (n,p) et de A(n,p)
	○ *D'optimisation et d'approximation*	- On traitera essentiellement : - Des problèmes d'optimisations - La recherche du zéro d'une fonction (f(x)=0) - La recherche du point fixe d'une fonction (f(x)=x) - Des méthodes de calcul d'une valeur approchée de constantes connues (π, e, …) - Calcul d'aires (rectangles, trapèzes)
	• ***Utiliser un environnement de programmation pour implémenter une solution.*** ○ *Implémenter un algorithme en un programme exécutable.* ○ *Écrire un programme pour résoudre un problème.*	- Il est utile d'inciter les apprenants à analyser un programme exécutable afin de comprendre les traitements. - Il est essentiel d'habituer les apprenants à commenter les solutions. - Apporter des modifications à une solution (actions correctives, actions évolutives). - Tester le programme solution sur ordinateur.

	- Concevoir une interface graphique pour développer des applications simples.	- Le langage adopté est Python. - Rappeler les composants d'une interface graphique à partir d'une application existante (fenêtre, Label, ~~liste déroulante (Combo Box)~~, Zone de texte (Text Edit, Line Edit), bouton (Push Button), bouton radio (Radio Button), ~~Case à cocher (Check Box)~~). - La conception d'une interface graphique se fait en utilisant la technique « Glisser–Déposer » (Drag & drop). - Utiliser Qtdesigner comme outil de création d'interfaces graphiques. - Il est recommandé d'utiliser des fichiers pour transférer et récupérer des informations. - Traiter les composants d'une interface graphique (List Widget, Table Widget) pour récupérer des informations à partir des fichiers textes et typés. - L'apprentissage est axé principalement sur la pratique.

Recommandations générales
- Avantager les échanges et les discussions autour des solutions proposées. - 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 de données avancées. - Il est judicieux d'utiliser la pédagogie active et de traiter divers problèmes de la vie courante (mathématiques, physiques, économies …) - L'apprentissage est axé principalement sur la pratique. - Il est recommandé d'aborder des problèmes et systématiser leurs résolutions en se basant sur les quatre composantes de la pensée computationnelle (décomposition, reconnaissance de modèles ou de formes, abstraction et algorithme). - Il est recommandé de consulter des communautés de développement et de partager des solutions (algorithmes ou programmes) dans des espaces de partage créés pour l'échange et l'apprentissage. - Favoriser l'exploitation des ressources en ligne. - 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. - Le langage adopté est Python.