AIDE PÉDAGOGIQUE 1ère ANNÉE SECONDAIRE

Domaine d'apprentissage : Technologies Internet		
Compétences disciplinaires à développer	**Pistes pédagogiques**	**Exemples d'activités**
▪ Exploiter les environnements de partage à bon escient.	▪ Participer à des communautés de partage existantes (pour communiquer, consulter, partager, discuter, déposer, répondre, etc.). ▪ Créer des communautés de partage (groupes, espaces de partage, etc.) pour des fins pédagogiques. ▪ Utiliser un environnement de partage dans le travail collaboratif et dans l'individualisation des apprentissages. **Exemples d'outils :** Facebook, Dropbox, Bouquet Google (Drive, Classroom, docs, …), Edmodo, …	▪ Exploiter une page Facebook pour communiquer, consulter, partager, discuter, déposer, répondre, etc. ▪ Créer un espace de stockage avec Google Drive puis l'utiliser pour stocker et partager les différentes productions. ▪ Collaborer pour produire un document avec Google docs définissant les règles d'utilisation et les bonnes pratiques sur les réseaux sociaux.
▪ Utiliser d'une façon avisée les technologies Internet.	▪ Guider les apprenants à adopter une attitude avisée vis-à-vis des communautés de partage (les réseaux sociaux, les plateformes, etc.)	▪ Exploiter des séquences vidéos pour sensibiliser les apprenants aux dangers de l'Internet et de l'usage de certaines applications mobiles.
▪ Rationaliser l'utilisation des applications mobiles.	▪ Sensibiliser les apprenants aux dangers de l'usage de certaines applications mobiles. ▪ Guider les apprenants à adopter une attitude avisée vis-à-vis des applications mobiles.	▪ Étudier des cas et mener des débats afin de guider les apprenants à adopter une attitude avisée vis-à-vis des applications mobiles.

Domaine d'apprentissage : Production numérique		
Compétences disciplinaires à développer	**Pistes pédagogiques**	**Exemples d'activités**
▪ Découvrir les technologies multidimensionnelles. ▪ Concevoir et réaliser des objets multidimensionnels. ▪ Produire des animations interactives.	▪ Distinguer et produire des objets 3D. ▪ Apporter des modifications sur des objets 3D. ▪ Se limiter à des formes géométriques simples pour la création d'objets 3D. ▪ Créer des animations interactives ▪ Utiliser différents supports pour publier les productions réalisées.	▪ Mener une recherche Internet afin de dégager la définition de la modélisation 3D, ses intérêts et ses domaines d'application. ▪ Présenter quelques exemples de produits en 2D et en 3D pour déterminer les caractéristiques de chacun. ▪ Chercher sur Internet des exemples de logiciels 3D et présenter les caractéristiques de chacun (3D Builder, SketchUp, Tinkercad, Blender, …). ▪ Présenter le logiciel à utiliser. ▪ Par groupe, imaginer un objet à manipuler d'une manière 3D (ils peuvent le dessiner) correspondant au projet (robotique). ▪ Apprendre les manipulations de base tout en produisant des objets 3D.

		▪ Créer des animations interactives (Par exemple : Avec SketchUp, on peut utiliser la palette « scène » pour générer des animations puis on utilise un logiciel de montage vidéo).
Domaine d'apprentissage : Programmation et Robotique		
Compétences disciplinaires à développer	**Pistes pédagogiques**	**Exemples d'activités**
▪ Découvrir différents environnements de programmation. ▪ Résoudre des problèmes et exprimer les solutions sous forme de code de programmation. ▪ Tirer profit des notions de programmation dans le domaine de la robotique. ▪ Développer le raisonnement et la pensée logique.	▪ L'initiation à l'utilisation du code se fera à partir d'un programme existant (exécution, exploration du code, modification) puis l'écriture de codes similaires. ▪ Modifier et/ou écrire un code de programmation pour résoudre un problème simple, faisant appel à des structures de contrôle. ▪ Se servir de dispositifs ou de robots pour appliquer des notions de programmation, en mettant à profit différents outils et langages de programmation. ▪ Monter et programmer des robots virtuels pour réaliser différentes tâches d'une façon innovante. ▪ Programmer des robots réels pour réaliser différentes tâches d'une façon innovante.	▪ Rappel des concepts relatifs à la robotique : • Définition et domaines d'application de la robotique. • Caractéristiques de la carte Micro:bit utilisé en 9ème année. ▪ Introduction à la programmation en lignes de code : • Rappeler l'utilisation de la carte Micro:bit (en exploitant le simulateur) dans un environnement de programmation visuelle (Faire quelques activités réalisées en 9ème année). • Utiliser la conversion automatique du code en Micro-Python. • Apporter des modifications en mode visuel puis constater les changements dans le code. • Expliquer brièvement chaque instruction du code. • Apporter des modifications sur le code en Python. ▪ Introduction à l'utilisation de la carte USP 32 : • Présenter la carte ESP32 • Décrire la carte ESP32 (composants, pins, …) tout en faisant la comparaison avec la carte Micro:bit. • Connecter la carte à l'ordinateur (installer le pilote de la carte ESP32 et installer l'environnement de travail : **Micro-Python**). ▪ Programmation de la carte pour réaliser différentes tâches telles que : • Faire clignoter une diode Led. • Feu de carrefour (rouge, vert, orangé). • Varier la luminosité d'une diode Led. • Allumer une diode LED par le toucher d'un pin. • Allumer une diode LED avec un bouton poussoir. ▪ Introduction à la programmation : • Tirer profit de la programmation de la carte USP32 pour introduire les notions de la programmation. • Présenter l'environnement de travail • Résoudre des problèmes simples et exprimer les solutions sous forme de code de programmation.

NB : Les domaines d'apprentissage se chevauchent.

Exemples de projet : Modélisation d'une chambre avec éclairage. /Modélisation d'une voiture avec éclairage des phares. / Accès à un parking.

AIDE PÉDAGOGIQUE 2ème ANNÉE SCIENCES

Domaine d'apprentissage : Pensée computationnelle et programmation	
Savoirs associés détaillés	**Recommandations**
Les étapes de résolution d'un problème ▪ Analyse d'un problème. • Définition • Représentations utilisées (graphique, textuelle, …) ▪ Écriture d'un algorithme. • Définition d'un algorithme • La forme générale d'un algorithme ▪ Implémentation en un langage de programmation. • Définition d'un langage de programmation • Le langage Python : présentation, utilisation, historique, environnement de développement, …	▪ Insister sur l'obligation de l'étape de l'analyse. ▪ Présenter les différentes formes qu'on peut adopter pour analyser un problème (organigramme, schéma, exemples, …). ▪ Présenter la forme générale d'un algorithme (avec le TDO). ▪ Respecter obligatoirement l'indentation. ▪ Commenter les algorithmes et les programmes. ▪ Installer un environnement de développement (Thonny, …) et présenter son interface.
Les structures de données ▪ Les objets (constantes et variables) • Les constantes : définition, caractéristiques, déclaration, exemples. • Les variables : définition, caractéristiques, déclaration, exemples. ▪ Les types de données • Le type entier : domaine et déclaration. • Le type réel : domaine et déclaration. • Le type booléen : domaine et déclaration. • Le type caractère : domaine, déclaration, opérateurs et les fonctions préfinies. • Le type chaîne de caractères : domaine, déclaration et les fonctions préfinies.	▪ Préciser les règles de nommage d'un objet. ▪ Présenter les opérateurs arithmétiques (+, -, /, *, Div, Mod) ▪ Présenter les opérateurs logiques (NON, ET, OU) ▪ Présenter les opérateurs relationnels (=, ≠, <, ≤, >, ≥, ∈) ▪ Présenter la priorité des opérateurs. ▪ Étudier les fonctions arithmétiques : **arrondi**, **abs**, **aléa**, **ent**, **racinecarré** ▪ Étudier les fonctions préfinies sur les caractères : **chr**, **ord** ▪ Étudier les fonctions préfinies sur les chaînes de caractères suivantes : **long**, **pos**, **convch**, **estnum**, **valeur**, **sous_chaîne**, **effacer**, **majus** ▪ Mentionner pour les chaînes de caractères : • L'indice du premier élément d'une chaîne de caractères. • L'accès à un caractère d'une chaîne • Les façons d'afficher une chaîne de caractères • L'apostrophes ou guillemets à l'intérieur d'une chaîne de caractères
Les structures simples ▪ L'action d'entrée ▪ L'action de sortie ▪ L'affectation	▪ Définir : Instruction et Séquence d'instructions ▪ Présenter pour l'affectation : Définition, types et Initialisation d'une variable ▪ Présenter quelques façons pour formater la sortie
Les structures de contrôle conditionnelles ▪ Définition ▪ La structure de contrôle conditionnelle simple : syntaxe, exemples. ▪ La structure de contrôle conditionnelle complète : syntaxe, exemples.	

Les structures de contrôle itératives ▪ Définition ▪ La structure de contrôle itérative complète : syntaxe, exemples.	▪ Pour la boucle « Pour », insister sur le fait que le nombre de répétitions est connu à l'avance. ▪ La valeur du **pas** peut être **positive** ou **négative**. Par défaut, elle est égale à 1. ▪ Éviter de modifier la valeur du compteur de la structure itérative complète au niveau du traitement.

Pistes pédagogiques et directives générales

- 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 les savoirs associés à travers des mini-projets ou des activités ayant du sens pour l'apprenant.
- 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.
- Diversifier les activités 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.
- Mettre l'accent sur le volet algorithmique.
- Inciter les apprenants à faire des échanges autour des solutions proposées et de les partager en ligne.
- Familiariser l'apprenant à formuler sous forme d'actions des solutions à des problèmes puisés de son vécu. On pourra exprimer ces solutions à l'aide d'un schéma, d'un organigramme, d'une carte mentale, 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.

Domaine d'apprentissage : Systèmes et technologies Internet	
Savoirs associés détaillés	**Recommandations**
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 • Présenter la carte ESP32 • Décrire la carte ESP32 (composants, pins, …) tout en faisant la comparaison avec la carte Micro:bit ▪ Savoir connecter un objet à l'ordinateur • Installer le pilote de la carte ESP32 • Installer l'environnement de travail ▪ 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. ▪ Exploiter les pré-acquis des élèves en collèges. ▪ L'apprenant n'est pas appelé à développer une application de commande (ne pas commander la carte à distance). ▪ Utiliser **Micro-Python / Arduino** pour programmer la carte. ▪ Traiter les activités telles que : • Faire clignoter une diode Led. • Feu de carrefour (rouge, vert, orangé). • Varier la luminosité d'une diode Led. • Allumer une diode LED par le toucher d'un pin. • Allumer une diode LED avec un bouton poussoir.

Domaine d'apprentissage : Pensée computationnelle et programmation	
Savoirs associés détaillés	**Recommandations**
Les étapes de résolution d'un problème ▪ Rappel • Analyse d'un problème. • Écriture d'un algorithme. • Implémentation en un langage de programmation.	▪ Insister sur l'obligation de l'étape de l'analyse. ▪ Rappeler les différentes formes qu'on peut adopter pour analyser un problème. ▪ Rappeler la forme générale d'un algorithme (avec le TDO). ▪ Respecter obligatoirement l'indentation. ▪ Commenter les algorithmes et les programmes.
Les structures de données ▪ Rappel • Les objets (constantes et variables). • Les types de données : Entier, Réel, Booléen, Caractère et Chaîne de caractères. ▪ Les tableaux à une dimension.	▪ Rappeler : • les opérateurs arithmétiques (+, -, /, *, Div, Mod), les opérateurs logiques (NON, ET, OU) et les opérateurs relationnels (=, ≠, <, ≤, >, ≥, ∈). • la priorité des opérateurs. • les fonctions arithmétiques : **arrondi**, **abs**, **aléa**, **ent**, **racinecarré** • les fonctions préfinies sur les caractères : **chr**, **ord** • les fonctions préfinies sur les chaînes de caractères suivantes : **long**, **pos**, **convch**, **estnum**, **valeur**, **sous_chaîne**, **effacer**, **majus** ▪ Définir les tableaux à une dimension. ▪ Préciser : • le type des éléments d'un tableau. • le type des indices des éléments d'un tableau. • l'accès à un élément d'un tableau. ▪ Implémenter les tableaux en utilisant la bibliothèque **Numpy**.
Les structures simples ▪ Rappel • L'action d'entrée • L'action de sortie • L'affectation	
Les structures de contrôle conditionnelles ▪ Rappel • Définition • La structure de contrôle conditionnelle simple. • La structure de contrôle conditionnelle complète. ▪ La structure de contrôle conditionnelle généralisée. ▪ La structure de contrôle conditionnelle à choix multiples.	▪ Le sélecteur de la structure « Selon » doit être de type scalaire.

Les structures de contrôle itératives ▪ Rappel • Définition • La structure de contrôle itérative complète. ▪ Les structures de contrôle itératives à condition d'arrêt. • La structure Tant que : Définition, syntaxe, exemples. • La structure Répéter: Définition, syntaxe, exemples.	▪ Pour la boucle « Pour », insister sur le fait que le nombre de répétitions est connu à l'avance. ▪ La valeur du **pas** peut être **positive** ou **négative**. Par défaut, elle est égale à 1. ▪ Éviter de modifier la valeur du compteur de la structure itérative complète au niveau du traitement. ▪ Pour les structures itératives à condition d'arrêt, insister sur le fait que le nombre de répétitions est inconnu à l'avance.
Les sous-programmes ▪ Définition ▪ Les fonctions : définition, syntaxe, déclaration, appel. ▪ Les procédures : définition, syntaxe, déclaration, appel. ▪ Déclaration et portée des objets : • Les objets locaux. • Les objets globaux. ▪ Les paramètres et leurs modes de passage : • Les paramètres formels. • Les paramètres effectifs. • Modes de passage des paramètres.	▪ La modularité peut être traitée dès le début de l'année. ▪ Choisir des exemples concrets pour montrer les avantages de la décomposition modulaire. ▪ Argumenter et justifier les choix de la modularité. ▪ Utiliser des modules prédéfinis et implémenter des modules personnels. ▪ Décrire correctement les entrées / les sorties et le rôle de chaque module. ▪ Une fonction retourne un seul résultat de type simple (entier, réel, booléen, caractère, chaîne).

Pistes pédagogiques et directives générales

- 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.
- 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.
- Inciter les apprenants à écrire des solutions modulaires.
- 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 :
  - 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.

Domaine d'apprentissage : Systèmes et technologies Internet	
Savoirs associés détaillés	**Recommandations**
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 • Présenter la carte ESP32 • Décrire la carte ESP32 (composants, pins, …) tout en faisant la comparaison avec la carte Micro:bit ▪ Savoir connecter un objet à l'ordinateur • Installer le pilote de la carte ESP32 • Installer l'environnement de travail ▪ 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. ▪ Exploiter les pré-acquis des élèves en collèges. ▪ L'apprenant n'est pas appelé à développer une application de commande (ne pas commander la carte à distance). ▪ Utiliser **Micro-Python / Arduino** pour programmer la carte. ▪ Traiter les activités telles que : • Faire clignoter une diode Led. • Feu de carrefour (rouge, vert, orangé). • Varier la luminosité d'une diode Led. • Allumer une diode LED par le toucher d'un pin. • Allumer une diode LED avec un bouton poussoir.

AIDE PÉDAGOGIQUE 4ème ANNÉE SCIENCES

Domaine d'apprentissage : Pensée computationnelle et programmation	
Savoirs associés détaillés	**Recommandations**
Rappel : ▪ **Les étapes de résolution d'un problème** • Analyse d'un problème. • Écriture d'un algorithme. • Implémentation en un langage de programmation. ▪ **Les structures de données** • Les objets (constantes et variables). • Les types de données simples : Entier, Réel, Booléen, Caractère et Chaîne de caractères. • Les tableaux à une dimension. ▪ **Les structures simples** • L'action d'entrée • L'action de sortie • L'affectation ▪ **Les structures de contrôle conditionnelles** • La structure de contrôle conditionnelle simple. • La structure de contrôle conditionnelle complète. • La structure de contrôle conditionnelle généralisée. • La structure de contrôle conditionnelle à choix multiples. ▪ **Les structures de contrôle itératives** • La structure de contrôle itérative complète. • Les structures de contrôle itératives à condition d'arrêt. ▪ **Les sous-programmes** • Les fonctions • Les procédures • Déclaration et portée des objets • Les paramètres et leurs modes de passage	▪ Distinguer les usages et les particularités de chaque type de données. ▪ Rappeler : • les opérateurs arithmétiques (+, -, /, *, Div, Mod), les opérateurs logiques (NON, ET, OU) et les opérateurs relationnels (=, ≠, <, ≤, >, ≥, ∈). • la priorité des opérateurs. • les fonctions arithmétiques : **arrondi**, **abs**, **aléa**, **ent**, **racinecarré** • les fonctions préfinies sur les caractères : **chr**, **ord** • les fonctions préfinies sur les chaînes de caractères suivantes : **long**, **pos**, **convch**, **estnum**, **valeur**, **sous_chaîne**, **effacer**, **majus** ▪ Rappeler les tableaux à une dimension : • le type des éléments d'un tableau. • le type des indices des éléments d'un tableau. • l'accès à un élément d'un tableau. • Implémentation avec les **listes** ou la bibliothèque **Numpy**. ▪ 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. ▪ Traiter des problèmes utilisant des structures de contrôle **imbriquées**. ▪ Choisir des exemples concrets pour montrer les avantages de la décomposition modulaire. ▪ Élaborer des solutions algorithmiques modulaires. ▪ Argumenter et justifier les choix de la modularité. ▪ Utiliser des modules prédéfinis et implémenter des modules personnels. ▪ Décrire correctement les entrées / les sorties et le rôle de chaque module.

Conception d'interfaces graphiques • Utiliser les objets graphiques (widgets) les plus usuels (boîte de dialogue, zone texte, label, bouton).	▪ 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. ▪ Comme exemples d'activités, 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 …).

Pistes pédagogiques et directives générales

- 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.
- 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.

- Inviter les apprenants à concevoir des solutions algorithmiques sous forme de pseudocodes.
- 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.
- Inviter les apprenants à identifier, pour un problème donné, une solution parmi plusieurs programmes proposés.
- 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.
- 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 chaînes.
    - 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.
- 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 :
    - somme des carrés des premiers entiers naturels
    - tri d'un tableau (au moins deux différents principes de tri).
    - calcul de combinaison.
    - somme des premiers entiers naturels
    - recherche d'un élément dans un tableau.
    - calcul arithmétique (PGCD, PPCM, nombres premiers, décomposition en facteurs premiers, etc.)