LE BULLETIN OFFICIEL DE L'ÉDUCATION NATIONALE

Annexe

Programme de numérique et sciences informatiques de première générale

Préambule

L'enseignement de spécialité de numérique et sciences informatiques du cycle terminal de la voie générale vise l'appropriation des fondements de l'informatique pour préparer les élèves à une poursuite d'études dans l'enseignement supérieur, en les formant à la pratique d'une démarche scientifique et en développant leur appétence pour des activités de recherche.

L'objectif de cet enseignement, non professionnalisant, est l'appropriation des concepts et des méthodes qui fondent l'informatique, dans ses dimensions scientifiques et techniques. Cet enseignement s'appuie sur l'universalité de quatre concepts fondamentaux et la variété de leurs interactions :

- Les **données**, qui représentent sous une forme numérique unifiée des informations très diverses : textes, images, sons, mesures physiques, sommes d'argent, etc.
- Les **algorithmes**, qui spécifient de façon abstraite et précise des traitements à effectuer sur les données à partir d'opérations élémentaires.
- Les **langages**, qui permettent de traduire les algorithmes abstraits en **programmes** textuels ou graphiques de façon à ce qu'ils soient exécutables par les machines.
- Les **machines**, et leurs systèmes d'exploitation, qui permettent d'exécuter des programmes en enchaînant un grand nombre d'instructions simples, assurant la persistance des données par leur stockage, et de gérer les communications. On y inclut les **objets connectés** et les **réseaux**.

À ces concepts s'ajoute un élément transversal : les **interfaces** qui permettent la communication avec les humains, la collecte des données et la commande des systèmes.

Cet enseignement prolonge les enseignements d'informatique dispensés à l'école primaire, au collège en mathématiques et en technologie et, en seconde, l'enseignement commun de sciences numériques et technologie. Il s'appuie aussi sur l'algorithmique pratiquée en mathématiques en seconde.

Il permet de développer des compétences :

- analyser et modéliser un problème en termes de flux et de traitement d'informations ;
- décomposer un problème en sous-problèmes, reconnaître des situations déjà analysées et réutiliser des solutions ;
- concevoir des solutions algorithmiques ;
- traduire un algorithme dans un langage de programmation, en spécifier les interfaces et les interactions, comprendre et réutiliser des codes sources existants, développer des processus de mise au point et de validation de programmes ;
- mobiliser les concepts et les technologies utiles pour assurer les fonctions d'acquisition, de mémorisation, de traitement et de diffusion des informations ;
- développer des capacités d'abstraction et de généralisation.

Cet enseignement a vocation à multiplier les occasions de mise en activité des élèves, **sous des formes variées** qui permettent de développer des compétences transversales :

- faire preuve d'autonomie, d'initiative et de créativité ;
- présenter un problème ou sa solution, développer une argumentation dans le cadre d'un débat ;
- coopérer au sein d'une équipe dans le cadre d'un projet ;
- rechercher de l'information, partager des ressources ;
- faire un usage responsable et critique de l'informatique.

La progression peut suivre un rythme annuel construit autour de périodes spécifiques favorisant une alternance entre divers types d'activités.

Cet enseignement contribue à l'acquisition des compétences numériques qui font l'objet d'une certification en fin de cycle terminal.

Comme tous les enseignements, cette spécialité contribue au développement des compétences orales à travers notamment la pratique de l'argumentation. Celle-ci conduit à préciser sa pensée et à expliciter son raisonnement de manière à convaincre. Elle permet à chacun de faire évoluer sa pensée, jusqu'à la remettre en cause si nécessaire, pour accéder progressivement à la vérité par la preuve. Si ces considérations sont valables pour tous les élèves, elles prennent un relief particulier pour ceux qui choisiront de poursuivre cet enseignement de spécialité en terminale et qui ont à préparer l'épreuve orale terminale du baccalauréat. Il convient que les travaux proposés aux élèves y contribuent dès la classe de première.

Démarche de projet

Un enseignement d'informatique ne saurait se réduire à une présentation de concepts ou de méthodes sans permettre aux élèves de se les approprier en développant des projets applicatifs.

Une part de l'horaire de l'enseignement d'au moins un quart du total en classe de première doit être réservée à la conception et à l'élaboration de projets conduits par des groupes de deux à quatre élèves.

Les projets réalisés par les élèves, sous la conduite du professeur, constituent un apprentissage fondamental tant pour la compréhension de l'informatique que pour l'acquisition de compétences. En classe de première comme en classe terminale, ils peuvent porter sur des problématiques issues d'autres disciplines et ont essentiellement pour but d'imaginer des solutions répondant à l'expression d'un besoin ; dans la mesure du possible, il convient de laisser le choix du thème du projet aux élèves eux-mêmes. Il peut s'agir d'un approfondissement théorique des concepts étudiés en commun, d'une application à d'autres disciplines telle qu'une simulation d'expérience, d'un travail sur des données socioéconomiques, du développement d'un logiciel de lexicographie, d'un projet autour d'un objet connecté ou d'un robot, de la conception d'une bibliothèque implémentant une structure de données complexe, d'un problème de traitement d'image ou de son, d'une application mobile, par exemple de réalité virtuelle ou augmentée, du développement d'un site *Web* associé à l'utilisation d'une base de données, de la réalisation d'un interprète d'un mini-langage, d'un programme de jeu de stratégie, etc.

La gestion d'un projet inclut des points d'étape pour faire un bilan avec le professeur, valider des éléments, contrôler l'avancement du projet ou adapter ses objectifs, voire le redéfinir partiellement, afin de maintenir la motivation des élèves.

Les professeurs veillent à ce que les projets restent d'une ambition raisonnable afin de leur permettre d'aboutir.

Modalités de mise en œuvre

Les activités pratiques et la réalisation de projets supposent, pour chaque élève, l'accès à un équipement relié à internet.

Un langage de programmation est nécessaire pour l'écriture des programmes : un langage simple d'usage, interprété, concis, libre et gratuit, multiplateforme, largement répandu, riche de bibliothèques adaptées et bénéficiant d'une vaste communauté d'auteurs dans le monde éducatif est à privilégier. Au moment de la conception de ce programme, le langage choisi est Python version 3 (ou supérieure). L'expertise dans tel ou tel langage de programmation n'est cependant pas un objectif de formation.

Éléments de programme

Le programme est organisé autour de huit rubriques. Il ne constitue cependant pas un plan de cours. Il appartient aux professeurs de choisir leur progression, sans faire de chaque partie un tout insécable et indépendant des autres. Au contraire, les mêmes notions peuvent être développées et éclairées dans différentes rubriques, en mettant en lumière leurs interactions.

Histoire de l'informatique

Cette rubrique transversale se décline dans chacune des sept autres.

Comme toute connaissance scientifique et technique, les concepts de l'informatique ont une histoire et ont été forgés par des personnes. Les algorithmes sont présents dès l'Antiquité, les machines à calculer apparaissent progressivement au XVII^{e} siècle, les sciences de l'information sont fondées au XIX^{e} siècle, mais c'est en 1936 qu'apparaît le concept de machine universelle, capable d'exécuter tous les algorithmes, et que les notions de machine, algorithme, langage et information sont pensées comme un tout cohérent. Les premiers ordinateurs ont été construits en 1948 et leur puissance a ensuite évolué exponentiellement. Parallèlement, les ordinateurs se sont diversifiés dans leur taille, leur forme et leur emploi : téléphones, tablettes, montres connectées, ordinateurs personnels, serveurs, fermes de calcul, méga-ordinateurs. Le réseau internet, développé depuis 1969, relie aujourd'hui ordinateurs et objets connectés.

Contenus	Capacités attendues	Commentaires
Événements clés de l'histoire de l'informatique	Situer dans le temps les principaux événements de l'histoire de l'informatique et leurs protagonistes.	Ces repères historiques seront construits au fur et à mesure de la présentation des concepts et techniques.

Représentation des données : types et valeurs de base

Toute machine informatique manipule une représentation des données dont l'unité minimale est le bit 0/1, ce qui permet d'unifier logique et calcul. Les données de base sont représentées selon un codage dépendant de leur nature : entiers, flottants, caractères et chaînes de caractères. Le codage conditionne la taille des différentes valeurs en mémoire.

Contenus	Capacités attendues	Commentaires
Écriture d'un entier positif dans une base $b \geqslant 2$	Passer de la représentation d'une base dans une autre.	Les bases 2, 10 et 16 sont privilégiées.
Représentation binaire d'un entier relatif	Évaluer le nombre de bits nécessaires à l'écriture en base 2 d'un entier, de la somme ou du produit de deux nombres entiers. Utiliser le complément à 2.	Il s'agit de décrire les tailles courantes des entiers (8, 16, 32 ou 64 bits). Il est possible d'évoquer la représentation des entiers de taille arbitraire de Python.
Représentation approximative des nombres réels : notion de nombre flottant	Calculer sur quelques exemples la représentation de nombres réels : 0.1, 0.25 ou 1/3.	0.2 + 0.1 n'est pas égal à 0.3. Il faut éviter de tester l'égalité de deux flottants. Aucune connaissance précise de la norme IEEE-754 n'est exigible.
Valeurs booléennes : 0, 1. Opérateurs booléens : and, or, not. Expressions booléennes	Dresser la table d'une expression booléenne.	Le ou exclusif (xor) est évoqué. Quelques applications directes comme l'addition binaire sont présentées. L'attention des élèves est attirée sur le caractère séquentiel de certains opérateurs booléens.
Représentation d'un texte en machine. Exemples des encodages ASCII, ISO-8859-1, Unicode	Identifier l'intérêt des différents systèmes d'encodage. Convertir un fichier texte dans différents formats d'encodage.	Aucune connaissance précise des normes d'encodage n'est exigible.

Représentation des données : types construits

À partir des types de base se constituent des types construits, qui sont introduits au fur et à mesure qu'ils sont nécessaires.

Il s'agit de présenter tour à tour les p-uplets (*tuples*), les enregistrements qui collectent des valeurs de types différents dans des champs nommés et les tableaux qui permettent un accès calculé direct aux éléments. En pratique, on utilise les appellations de Python, qui peuvent être différentes de celles d'autres langages de programmation.

Contenus	Capacités attendues	Commentaires
p-uplets. p-uplets nommés	Écrire une fonction renvoyant un p-uplet de valeurs.	

Tableau indexé, tableau donné en compréhension	Lire et modifier les éléments d'un tableau grâce à leurs index. Construire un tableau par compréhension. Utiliser des tableaux de tableaux pour représenter des matrices : notation a [i] [j]. Itérer sur les éléments d'un tableau.	Seuls les tableaux dont les éléments sont du même type sont présentés. Aucune connaissance des tranches (*slices*) n'est exigible. L'aspect dynamique des tableaux de Python n'est pas évoqué. Python identifie listes et tableaux. Il n'est pas fait référence aux tableaux de la bibliothèque NumPy.
Dictionnaires par clés et valeurs	Construire une entrée de dictionnaire. Itérer sur les éléments d'un dictionnaire.	Il est possible de présenter les données EXIF d'une image sous la forme d'un enregistrement. En Python, les p-uplets nommés sont implémentés par des dictionnaires. Utiliser les méthodes *keys*(), *values* () et *items* ().

Traitement de données en tables

Les données organisées en table correspondent à une liste de p-uplets nommés qui partagent les mêmes descripteurs. La mobilisation de ce type de structure de données permet de préparer les élèves à aborder la notion de base de données qui ne sera présentée qu'en classe terminale. Il s'agit d'utiliser un tableau doublement indexé ou un tableau de p-uplets, dans un langage de programmation ordinaire et non dans un système de gestion de bases de données.

Contenus	Capacités attendues	Commentaires
Indexation de tables	Importer une table depuis un fichier texte tabulé ou un fichier CSV.	Est utilisé un tableau doublement indexé ou un tableau de p-uplets qui partagent les mêmes descripteurs.
Recherche dans une table	Rechercher les lignes d'une table vérifiant des critères exprimés en logique propositionnelle.	La recherche de doublons, les tests de cohérence d'une table sont présentés.
Tri d'une table	Trier une table suivant une colonne.	Une fonction de tri intégrée au système ou à une bibliothèque peut être utilisée.
Fusion de tables	Construire une nouvelle table en combinant les données de deux tables.	La notion de domaine de valeurs est mise en évidence.

Interactions entre l'homme et la machine sur le *Web*

Lors de la navigation sur le Web, les internautes interagissent avec leur machine par le biais des pages *Web*.

L'Interface Homme-Machine (IHM) repose sur la gestion d'événements associés à des éléments graphiques munis de méthodes algorithmiques.

La compréhension du dialogue client-serveur déjà abordé en classe de seconde est consolidée, sur des exemples simples, en identifiant les requêtes du client, les calculs puis les réponses du serveur traitées par le client.

Il ne s'agit pas de décrire exhaustivement les différents éléments disponibles, ni de développer une expertise dans les langages qui permettent de mettre en œuvre le dialogue tels que PHP ou *JavaScript*.

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine Événements	Identifier les différents composants graphiques permettant d'interagir avec une application *Web*. Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	Il s'agit d'examiner le code HTML d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript.
Interaction avec l'utilisateur dans une page *Web*	Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton d'une page *Web*.	
Interaction client-serveur. Requêtes HTTP, réponses du serveur	Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre. Distinguer ce qui est mémorisé dans le client et retransmis au serveur. Reconnaître quand et pourquoi la transmission est chiffrée.	Il s'agit de faire le lien avec ce qui a été vu en classe de seconde et d'expliquer comment on peut passer des paramètres à un site grâce au protocole HTTP.
Formulaire d'une page *Web*	Analyser le fonctionnement d'un formulaire simple. Distinguer les transmissions de paramètres par les requêtes POST ou GET.	Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité.

Architectures matérielles et systèmes d'exploitation

Exprimer un algorithme dans un langage de programmation a pour but de le rendre exécutable par une machine dans un contexte donné. La découverte de l'architecture des machines et de leur système d'exploitation constitue une étape importante.

Les circuits électroniques sont au cœur de toutes les machines informatiques. Les réseaux permettent de transmettre l'information entre machines. Les systèmes d'exploitation gèrent et optimisent l'ensemble des fonctions de la machine, de l'exécution des programmes aux entrées-sorties et à la gestion d'énergie.

On étudie aussi le rôle des capteurs et actionneurs dans les entrées-sorties clavier, interfaces graphiques et tactiles, dispositifs de mesure physique, commandes de machines, etc.

Contenus	Capacités attendues	Commentaires
Modèle d'architecture séquentielle (von Neumann)	Distinguer les rôles et les caractéristiques des différents constituants d'une machine. Dérouler l'exécution d'une séquence d'instructions simples du type langage machine.	La présentation se limite aux concepts généraux. On distingue les architectures monoprocesseur et les architectures multiprocesseur. Des activités débranchées sont proposées. Les circuits combinatoires réalisent des fonctions booléennes.
Transmission de données dans un réseau Protocoles de communication Architecture d'un réseau	Mettre en évidence l'intérêt du découpage des données en paquets et de leur encapsulation. Dérouler le fonctionnement d'un protocole simple de récupération de perte de paquets (bit alterné). Simuler ou mettre en œuvre un réseau.	Le protocole peut être expliqué et simulé en mode débranché. Le lien est fait avec ce qui a été vu en classe de seconde sur le protocole TCP/IP. Le rôle des différents constituants du réseau local de l'établissement est présenté.
Systèmes d'exploitation	Identifier les fonctions d'un système d'exploitation. Utiliser les commandes de base en ligne de commande. Gérer les droits et permissions d'accès aux fichiers.	Les différences entre systèmes d'exploitation libres et propriétaires sont évoquées. Les élèves utilisent un système d'exploitation libre. Il ne s'agit pas d'une étude théorique des systèmes d'exploitation.
Périphériques d'entrée et de sortie Interface Homme-Machine (IHM)	Identifier le rôle des capteurs et actionneurs. Réaliser par programmation une IHM répondant à un cahier des charges donné.	Les activités peuvent être développées sur des objets connectés, des systèmes embarqués ou robots.

Langages et programmation

Les langages de programmation Turing-complets sont caractérisés par un corpus de « constructions élémentaires ». Sans introduire cette terminologie, il s'agit de montrer qu'il existe de nombreux langages de programmation, différents par leur style (impératif, fonctionnel, objet, logique, événementiel, etc.), ainsi que des langages formalisés de description ou de requêtes qui ne sont pas des langages de programmation.

L'importance de la spécification, de la documentation et des tests est à présenter, ainsi que l'intérêt de la modularisation qui permet la réutilisation de programmes et la mise à disposition de bibliothèques. Pour les programmes simples écrits par les élèves, on peut se contenter d'une spécification rapide mais précise.

Contenus	Capacités attendues	Commentaires
Constructions élémentaires	Mettre en évidence un corpus de constructions élémentaires.	Séquences, affectation, conditionnelles, boucles bornées, boucles non bornées, appels de fonction.
Diversité et unité des langages de programmation	Repérer, dans un nouveau langage de programmation, les traits communs et les traits particuliers à ce langage.	Les manières dont un même programme simple s'écrit dans différents langages sont comparées.
Spécification	Prototyper une fonction. Décrire les préconditions sur les arguments. Décrire des postconditions sur les résultats.	Des assertions peuvent être utilisées pour garantir des préconditions ou des postconditions.
Mise au point de programmes	Utiliser des jeux de tests.	L'importance de la qualité et du nombre des tests est mise en évidence. Le succès d'un jeu de tests ne garantit pas la correction d'un programme.
Utilisation de bibliothèques	Utiliser la documentation d'une bibliothèque.	Aucune connaissance exhaustive d'une bibliothèque particulière n'est exigible.

Algorithmique

Le concept de méthode algorithmique est introduit ; de nouveaux exemples seront vus en terminale. Quelques algorithmes classiques sont étudiés. L'étude de leurs coûts respectifs prend tout son sens dans le cas de données nombreuses, qui peuvent être préférentiellement des données ouvertes.

Il est nécessaire de montrer l'intérêt de prouver la correction d'un algorithme pour lequel on dispose d'une spécification précise, notamment en mobilisant la notion d'invariant sur des exemples simples. La nécessité de prouver la terminaison d'un programme est mise en évidence dès qu'on utilise une boucle non bornée (ou, en terminale, des fonctions récursives) grâce à la mobilisation de la notion de variant sur des exemples simples.

Contenus	Capacités attendues	Commentaires
Parcours séquentiel d'un tableau	Écrire un algorithme de recherche d'une occurrence sur des valeurs de type quelconque. Écrire un algorithme de recherche d'un extremum, de calcul d'une moyenne.	On montre que le coût est linéaire.
Tris par insertion, par sélection	Écrire un algorithme de tri. Décrire un invariant de boucle qui prouve la correction des tris par insertion, par sélection.	La terminaison de ces algorithmes est à justifier. On montre que leur coût est quadratique dans le pire cas.
Algorithme des k plus proches voisins	Écrire un algorithme qui prédit la classe d'un élément en fonction de la classe majoritaire de ses k plus proches voisins.	Il s'agit d'un exemple d'algorithme d'apprentissage.
Recherche dichotomique dans un tableau trié	Montrer la terminaison de la recherche dichotomique à l'aide d'un variant de boucle.	Des assertions peuvent être utilisées. La preuve de la correction peut être présentée par le professeur.
Algorithmes gloutons	Résoudre un problème grâce à un algorithme glouton.	Exemples : problèmes du sac à dos ou du rendu de monnaie. Les algorithmes gloutons constituent une méthode algorithmique parmi d'autres qui seront vues en terminale.