Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE
DE L'ENSEIGNEMENT
SUPÉRIEUR,
DE LA RECHERCHE
ET DE L'INNOVATION

Classes préparatoires aux grandes écoles

Filière scientifique

Voie Biologie, chimie, physique et sciences de la Terre (BCPST)

Annexe 4

Programmes d'informatique

1ère et 2nde années

Programme d'informatique
Filière BCPST
Première et deuxième années

6 décembre 2020

Table des matières

Introduction au programme

Les objectifs du programme Le programme d'informatique de BCPST s'inscrit entre deux continuités : en amont avec les programmes rénovés du lycée, en aval avec les enseignements dispensés dans les grandes écoles, et plus généralement les poursuites d'études universitaires. Il a pour objectif la formation de futurs ingénieures et ingénieurs, vétérinaires, enseignantes et enseignants, chercheuses et chercheurs et avant tout des personnes informées, capables de gouverner leur vie professionnelle et citoyenne en pleine connaissance et maîtrise des techniques et des enjeux de l'informatique et en la nourrissant par les habitudes de la démarche scientifique.
Le présent programme a pour ambition de poser les bases d'un enseignement cohérent et mesuré d'une science informatique encore jeune et dont les manifestations technologiques connaissent des cycles d'obsolescence rapide. On garde donc à l'esprit :

— de privilégier la présentation de concepts fondamentaux pérennes sans s'attacher outre mesure à la description de technologies, protocoles ou normes actuels ;
— de donner aux futurs diplômées et diplômés les moyens de réussir dans un domaine en mutation rapide et dont les technologies qui en sont issues peuvent sauter brutalement d'un paradigme à un autre très différent ;
— de préparer les étudiantes et étudiants à tout un panel de professions et de situations de la vie professionnelle qui les amène à remplir tour à tour une mission d'expertise, de création ou d'invention, de prescription de méthodes ou de techniques, de contrôle critique des choix opérés ou encore de décision en interaction avec des spécialistes ;

Compétences visées Ce programme vise à développer les six grandes compétences suivantes :

analyser et modéliser un problème ou une situation, notamment en utilisant les objets conceptuels de l'informatique pertinents (table relationnelle, graphe, dictionnaire, etc.) ;

imaginer et concevoir une solution, décomposer en blocs, se ramener à des sous-problèmes simples et indépendants, adopter une stratégie appropriée, décrire une démarche, un algorithme ou une structure de données permettant de résoudre le problème ;

décrire et spécifier les caractéristiques d'un processus, les données d'un problème, ou celles manipulées par un algorithme ou une fonction ;

mettre en œuvre une solution, par la traduction d'un algorithme ou d'une structure de données dans un langage de programmation ou un langage de requête ;

justifier et critiquer une solution, en développant des processus d'évaluation, de contrôle, de validation d'un code que l'on a produit ;

communiquer à l'écrit ou à l'oral, présenter des travaux informatiques, une problématique et sa solution ; défendre ses choix ; documenter sa production et son implémentation.

La pratique régulière de la résolution de problèmes par une approche algorithmique et des activités de programmation qui en résultent constitue un aspect essentiel de l'apprentissage de l'informatique. Les exemples ou les exercices d'application peuvent être choisis au sein de l'informatique elle-même ou en lien avec d'autres champs disciplinaires.

Sur les partis pris par le programme Ce programme impose aussi souvent que possible des choix de vocabulaire ou de notation de certaines notions. Les choix opérés ne présument pas la supériorité de l'option retenue. Ils ont été précisés dans l'unique but d'aligner les pratiques d'une classe à une autre et d'éviter l'introduction de longues définitions récapitulatives préliminaires à un exercice ou un problème. De même, ce programme nomme aussi souvent que possible l'un des algorithmes possibles parmi les classiques qui répondent à un problème donné. Là encore, le programme ne défend pas la prééminence d'un algorithme ou d'une méthode par rapport à un autre mais il invite à faire bien plutôt que beaucoup.

Sur les langages et la programmation L'enseignement du présent programme repose sur un langage de manipulation de données (SQL) ainsi que le langage de programmation Python, pour lequel une annexe liste de façon limitative les éléments qui sont exigibles des étudiants. La poursuite de l'apprentissage du langage Python est vue en particulier par les étudiants pour adopter immédiatement une bonne discipline de programmation tout en se concentrant sur le noyau du langage plutôt que sur une interface de programmation applicative (API) pléthorique.

Mode d'emploi Ce programme a été rédigé par semestre pour assurer une certaine homogénéité de la formation. Le premier semestre permet d'asseoir les bases de programmation vues au lycée et les concepts associés. L'organisation de la progression au sein des semestres relève de la responsabilité pédagogique de la professeure ou du professeur et le tissage de liens entre les thèmes contribue à la valeur de son enseignement. Les notions étudiées lors d'un semestre précédent sont régulièrement revisitées tout au long des deux années d'enseignement.

1 Programme du semestre 1

Les séances de travaux pratiques du premier semestre poursuivent les objectifs suivants :
- consolider l'apprentissage de la programmation en langage Python qui a été entrepris dans les classes du lycée ;
- mettre en place un environnement de travail ;
- mettre en place une discipline de programmation : spécification précise des fonctions et programmes, annotations et commentaires, jeux de tests ;

La consolidation du langage Python portera principalement sur
- les variables
- les expressions et instructions
- les instructions conditionnelles
- les fonctions
- les instructions itératives
- la manipulation de quelques trsuctures de données

Rappelons que l'annexe liste les éléments du langage Python qui sont exigibles des étudiants.

Le tableau ci-dessous présente les thèmes qui sont abordés lors de ces séances et, en colonne de droite, une liste, sans aucun caractère impératif, d'exemples d'activités qui peuvent être proposées aux étudiants. L'ordre de ces thèmes n'est pas impératif.

Aucune connaissance relative aux modules éventuellement rencontrés lors de ces séances n'est exigible des étudiants.

Thèmes	**Exemples d'activité. *Commentaires.***
Algorithmes élémentaires opérant par boucles simples.	Calculs de sommes et produits. Calculs des termes d'une suite récurrente (ordre 1, ordre supérieur), liste des termes, chaînes de caractères.
Algorithme opérant par boucles dans un tableau unidimensionnel.	Recherche d'un élément. Recherche du maximum, du second maximum. Application aux polynômes : représentation par le tableau de ses coefficients (évaluation, opération, détermination du degré). *Manipulations élémentaires d'un tableau unidimensionnel (indexation, extraction, etc.).*
Lecture et écriture dans un fichier texte.	
Utilisation de modules, de bibliothèques.	Calculs statistiques sur des données. Représentation graphique (histogrammes, etc.).
Algorithmes opérant par boucles imbriquées.	Recherche d'un facteur (ou d'un mot) dans un texte. Recherche des deux valeurs les plus proches dans un tableau. Manipulations élémentaires des tableaux à deux dimensions (indexation et extraction, etc.). Calculs de la somme, du produit et de la transposée d'une matrice. *On en profitera pour introduire la bibliothèque « NUMPY ».*
Recherche dichotomique.	Recherche de valeurs approchées d'une racine d'une équation algébrique. Recherche dichotomique dans un tableau trié. *On met en évidence une accélération du processus.*
Fonctions récursives.	Factorielles, suites récurrentes. Algorithme d'exponentiation rapide. Dessins de fractales. *On évite de se cantonner à des fonctions mathématiques.*
Matrices de pixels et images.	Obtention d'une image en niveaux de gris, image miroir, négatif. Algorithmes de rotation, de réduction ou d'agrandissement. Modification d'une image par convolution : flou, détection de contour, etc. *0n pourra utiliser la bibliothèque « PIL ».*
Tris.	Algorithmes naïfs : tri par insertion, par sélection. Tri par comptage. Application aux statistiques : tri d'une série statistique, recherche de la médiane (éventuellement des quartiles). *0n pourra faire le lien entre le tri par comptage et la recherche des effectifs d'apparition dans une liste.*

2 Programme du semestre 2

On formalise par des leçons et travaux pratiques le travail entrepris au premier semestre concernant la discipline et les méthodes de programmation.

2.1 Méthodes de programmation et dictionnaire

Même si on ne parle pas de preuve d'algorithme, on insistera sur l'importance des tests dans la mise au point des programmes.

Notions	**Exemples d'activité. *Commentaires.***
Identifiants et valeurs. Objets mutables et non mutables, portée d'un identifiant, effets de bord.	On mettra en évidence le phénomène d'aliasing et son impact dans le cas d'objets mutables (listes).
Dictionnaires, clés et valeurs. Usage des dictionnaires en programmation Python. Syntaxe pour l'écriture des dictionnaires. Parcours d'un dictionnaire.	Nombre d'éléments distincts dans une liste, construction d'un index.

2.2 Bases de données

On se limite volontairement à une description applicative des bases de données en langage SQL. Il s'agit de permettre d'interroger une base présentant des données à travers plusieurs relations.

Notions	**Exemples d'activité. *Commentaires.***
Vocabulaire des bases de données : tables ou relations, attributs ou colonnes, domaine, schéma de tables, enregistrements ou lignes, types de données.	On présente ces concepts à travers de nombreux exemples. On s'en tient à une notion sommaire de domaine : entier, flottant, chaîne ; aucune considération quant aux types des moteurs SQL n'est au programme. Aucune notion relative à la représentation des dates n'est au programme ; en tant que de besoin on s'appuie sur des types numériques ou chaîne pour lesquels la relation d'ordre coïncide avec l'écoulement du temps. Toute notion relative aux collations est hors programme ; en tant que de besoin on se place dans l'hypothèse que la relation d'ordre correspond à l'ordre lexicographique usuel. `NULL` est hors programme.
Clé primaire, clé étrangère	On se limite au cas où une clé primaire est associée à un unique attribut.
Requêtes `SELECT` avec simple clause `WHERE` (sélection), projection, renommage `AS`. Utilisation des mots-clés `DISTINCT` et `ORDER BY`.	Les opérateurs au programme sont `+`, `-`, `*`, `/` (on passe outre les subtilités liées à la division entière ou flottante), `=`, `<>`, `<`, `<=`, `>`, `>=`, `AND`, `OR`, `NOT`. D'autres mots-clés comme `OFFSET` et `LIMIT` pourront être utilisés mas sleur maîtrise n'est pas au programme.
Jointures T_1 `JOIN` T_2 … `JOIN` T_n `ON` ϕ.	On présente les jointures en lien avec la notion de relations entre tables. On se limite aux équi-jointures : ϕ est une conjonction d'égalités.
Agrégation avec les fonctions `MIN`, `MAX`, `SUM`, `AVG` et `COUNT`, y compris avec `GROUP BY`.	Pour la mise en œuvre des agrégats, on s'en tient à la norme SQL99. Les requêtes imbriquées ne sont pas au programme.
Filtrage des agrégats avec `HAVING`.	

Mise en œuvre
La création de tables et la suppression de tables au travers du langage SQL sont hors programme. La mise en œuvre effective se fait au travers d'un logiciel permettant d'interroger une base de données à l'aide de requêtes SQL comme *MySQL* ou *SQLite*. Récupérer le résultat d'une requête à partir d'un programme n'est pas un objectif. Sont hors programme : la notion de modèle logique *vs* physique, les bases de données non relationnelles, les méthodes de modélisation de base, les fragments DDL, TCL et ACL du langage SQL, les transactions, l'optimisation de requêtes par l'algèbre relationnelle.

2.3 Graphes

Il s'agit de définir le modèle des graphes, leurs représentations et leurs manipulations.
On s'efforce de mettre en avant des applications importantes et si possibles modernes : réseau de transport, graphe du web, réseaux sociaux, bio-informatique. On précise autant que possible la taille typique de tels graphes.

Notions	**Exemples d'activité. *Commentaires.***
Vocabulaire des graphes. Graphe orienté, graphe non orienté. Sommet (ou nœud) ; arc, arête. Boucle. Chemin d'un sommet à un autre. Connexité dans les graphes non orientés. Matrice d'adjacence. Graphe $G = (S, A)$.	On présente l'implémentation des graphes à l'aide de listes d'adjacence (rassemblées par exemple dans une liste ou dans un dictionnaire). On n'évoque ni multi-arcs ni multi-arêtes.
Pondération d'un graphe. Étiquettes des arcs ou des arêtes d'un graphe.	On motive l'ajout d'information à un graphe par des exemples concrets.
Parcours d'un graphe. Parcours en largeur.	*La file sera représentée par une liste. On pourra évoquer le problème de cette représentation naïve en terme d'efficacité mais aucune connaissance sur d'autres représentations plus performante n'est au programme.*

2.4 Méthodes numériques

Cette section propose des sujets de travaux pratiques venant compléter le programme du deuxième semestre.

Thèmes	**Exemples d'activité. *Commentaires.***
Méthode de dichotomie et méthode des rectangles.	Comparaison avec d'autres méthodes : méthode de Newton, méthode des trapèzes.
Simulation de lois usuelles : Bernoulli, binomiale, hypergéométrique, uniforme. Estimation d'une probabilité, estimation de l'espérance et de la variance.	Simulation d'expériences et de variables aléatoires. Simulation d'une variable aléatoire à l'aide de sa fonction de répartition. *La justification de cette estimation sera donnée en deuxième année*

3 Programme des semestres 3 et 4

Le programme de deuxième année est constitué de séances de travaux pratiques qui poursuivent les objectifs suivants :

— approfondir les connaissances acquises au deuxième semestre notamment sur les bases de données et sur les graphes ;
— programmer des méthodes numériques vues en mathématiques ;
— mettre en application des notions vues en statistiques et probabilités dont les tests statistiques ;
— programmer de nouveaux algorithmes sur des applications en lien avec d'autres disciplines, notamment la biologie.

Comme au premier semestre, les tableaux ci-dessous présentent les thèmes qui sont abordés lors de ces séances et, en colonne de droite, une liste, sans aucun caractère impératif, d'exemples d'activités qui peuvent être proposées aux étudiants. L'ordre de ces thèmes n'est pas impératif.

Aucune connaissance relative aux modules éventuellement rencontrés lors de ces séances n'est exigible des étudiants.

Les deux parties du programme représentent un volume horaire équivalent.

3.1 Méthodes numériques et statistiques

Thèmes	**Exemples d'activités** *Commentaires*
Méthode d'Euler explicite pour la résolution approchée d'équation différentielle ordinaire d'ordre 1.	Modèle logistique, comparaison avec d'autres méthodes de résolution approchée (méthode de Heun par exemple). *L'impact du pas de discrétisation sur la qualité des résultats et sur le temps de calcul est mis en évidence.* *Les résultats obtenus peuvent être comparés avec une résolution exacte ou avec une fonction de résolution approchée fournie par un module.*
Simulation d'une variable aléatoire de loi géométrique à l'aide de la loi de Bernoulli. Estimation de l'espérance, de la loi d'une variable aléatoire à partir de simulation.	On pourra simuler des variables aléatoires à l'aide de la loi uniforme et de la réciproque de la fonction de répartition : cas discret et continu. Exemples de la loi de Poisson, de la loi exponentielle et de la loi normale. Exemples de chaînes de Markov. Fonction de répartition empirique.
Illustration numérique de convergence en lois. Simulation d'une loi de Poisson à l'aide d'une loi binomiale. Intervalle de confiance pour le paramètre d'une loi de Bernoulli.	Illustration du théorème central limite. *La notion théorique d'intervalle de confiance n'est pas au programme*
Approfondissement sur les statistiques : simulation et mécanisme de tests. Simulation de variables aléatoires de lois de khi-2 et de Student. Elaborer d'un test de moyenne, dit de Student, sur un petit échantillon gaussien.	Elaboration pour un test de la moyenne, cas particulier d'une proportion. On affichera les histogrammes obtenus. On pourra visualiser la convergence des lois de Student vers une loi normale. *La connaissance des lois de Student et du khi-2 n'est pas un attendu du programme. On pourra utiliser les simulateurs intégrés dans Python.*

On pourra compléter cette partie par une étude sur un autre sujet ou approfondir un des thèmes précédents. Les thèmes suivants ont été choisis par leur lien possibles avec des applications possible en SVT mais ils ne sont que des exemples proposés.

Exemples de thèmes libres	**Exemples d'activités** *Commentaires*
Complément sur la méthode d'Euler.	Méthode d'Euler explicite pour la résolution approchée de système d'équations différentielles d'ordre 1 et d'équation différentielle d'ordre 2. Modèle de Lotka-Volterra, modèle SIR, système différentiel issu d'une équation de cinétique chimique, équation d'un oscillateur harmonique amorti. *Une programmation vectorisée, même si elle peut être proposée, n'est pas un attendu du programme.*
Méthode du pivot de Gauss.	Résolution d'un système linéaire. *Une version sans recherche de pivot peut être proposée en premier puis une recherche de pivot partiel peut être programmée.* *On pourra mettre en évidence l'impact de la taille du système sur le temps de calcul.* *La programmation sans aide n'est pas un attendu du programme.*
Autour des équations aux dérivées partielles.	Équation de la chaleur, couplage réaction-diffusion. *Aucune connaissance sur les équations aux dérivées partielles n'est exigible.*

3.2 Approfondissements des concepts informatiques

Thèmes	**Exemples d'activités** *Commentaires*
Révisions et approfondissements sur les graphes. Parcours en profondeur et parcours en profondeur itéré. Plus court chemin dans un graphe pondéré.	Algorithme de Dijkstra. Problème du voyageur de commerce. Algorithme glouton, colonie de fourmis, recuit simulé. *L'objectif est de reprendre la structure de graphe à travers un approfondissement possible.*
Révisions des tris naïfs et exemple de tri récursif.	Tri fusion ou tri rapide. On pourra visualiser à l'aide de graphique l'accélération du temps de calcul. *La programmation du tri rapide en place n'est pas un objectif du programme. L'objectif sur les deux années est qu'un étudiant sache programmer un tri de son choix de façon autonome.*
Révisions et pprofondissements sur les bases de données : révision sur les jointures et agrégations	*A partir d'une base de données comprenant 3 ou 4 tables ou relations, on approfondit la notion de jointure interne en lien avec la notion d'associations entre entités.* *Pour la notion d'agrégation, on présente quelques exemples de requêtes imbriquées.* *On marque la différence entre WHERE et HAVING sur des exemples.*

L'objectif de cette dernière partie est à travers l'étude d'un thème, sur plusieurs séances, de permettre aux étudiants de gagner en autonomie (choix de la représentation des données, choix du découpage en fonctions, etc).

Les thèmes suivants ont été choisis par leurs applications possibles en lien avec la SVT mais ils ne sont que des exemples proposés.

Exemples de thèmes libres	**Exemples d'activités** *Commentaires*
Informatique pour la génétique.	Recherche de motif : recherche exacte d'une ou plusieurs séquences courtes dans un génome. Algorithme naïf, algorithme de Robin-Karp. Alignement de séquences en présence d'erreurs de séquençage ou de mutations. Distance d'édition, algorithmes de Needleman-Wunsch, de Smith-Waterman. Construction d'arbres phylogénétiques.
Images.	Segmentation d'images : comment partitionner une image en zones connexes ? *La structure union-find peut être introduite dans ce but.*
Apprentissage et classification.	Algorithme des k-moyennes. Classification par k plus proches voisins. *On pourra travailler autant sur des données réelles que sur des données issues de simulations (clusters gaussiens par exemple).*

A Langage Python

Cette annexe liste limitativement les éléments du langage Python (version 3 ou supérieure) dont la connaissance est exigible des étudiants. Aucun concept sous-jacent n'est exigible au titre de la présente annexe.
Aucune connaissance sur un module particulier n'est exigible des étudiants.
Toute utilisation d'autres éléments du langage que ceux que liste cette annexe, ou d'une fonction d'un module, doit obligatoirement être accompagnée de la documentation utile, sans que puisse être attendue une quelconque maîtrise par les étudiants de ces éléments.

Traits généraux

— Principe d'indentation.
— Portée lexicale : lorsqu'une expression fait référence à une variable à l'intérieur d'une fonction, Python cherche la valeur définie à l'intérieur de la fonction et à défaut la valeur dans l'espace global du module.

Types de base

— Opérations sur les entiers (`int`) : `+`, `-`, `*`, `**`, avec des opérandes positifs.
— Opérations sur les flottants (`float`) : `+`, `-`, `*`, `/`, `**`.
— Opérations sur les booléens (`bool`) : **`not`**, **`or`**, **`and`**.
— Comparaisons `==`, `!=`, `<`, `>`, `<=`, `>=`.

Types structurés

— Structures indicées immuables (chaînes de caractères) : `len`, accès par indice positif valide, concaténation `+`, répétition `*`, tranche.
— Listes : création par compréhension `[`*e* **`for`** *x* **`in`** *s*`]`, par `append` successifs ; `len`, accès par indice positif valide ; concaténation `+`, répétition `*`, tranche, copie ; `pop` en dernière position.
— Dictionnaires : création, accès, insertion, `len`, `copy`.

Structures de contrôle

— Instruction d'affectation avec `=`.
— Instruction conditionnelle : **`if`**, **`elif`**, **`else`**.
— Boucle **`while`** (sans **`else`**), **`return`** dans un corps de boucle.
— Boucle **`for`** (sans **`else`**) et itération sur `range(`a`, `b`)`, une chaîne de caractères une liste, un dictionnaire au travers des méthodes `keys` et `items`.
— Définition d'une fonction **`def`** f`(`p_1`, ..., `p_n`)`, **`return`**.

Divers

— Introduction d'un commentaire avec `#`.
— Utilisation simple de `print`, sans paramètre facultatif.
— Importation de modules avec **`import`** *module*, **`import`** *module* **`as`** *alias*, **`from`** *module* **`import`** $f, g, \ldots$