# Présentation du programme d'informatique

UPS

31 mai 2015

## Table des matières

*Ce document est produit par des enseignants d'informatique (hors option informatique) des classes préparatoires en première et seconde année des filières MP, PC, PSI, PT, TPC et TSI. Son but est de proposer l'équivalent des comparatifs des programmes qui sont écrits lors des changements de programme. Comme dans ce cas il n'y a pas réellement de programme antérieur, il s'agit de proposer une lecture du programme en en montrant le contenu et les limites. Nous espérons qu'il permettra d'atteindre un consensus sur la lecture des programmes et, qui sait, aux concepteurs de sujets d'avoir une vision de ces programmes qui leur permette de savoir à quoi s'attendre de la part des candidats.*

*Bien entendu ce texte ne remplace en aucun cas le programme officiel, il se veut un document d'accompagnement. La lecture faite de ce programme n'est pas une lecture juridique, des choix d'interprétation ont été faits. Ces interprétations qui pourraient être lues comme des ajouts, ne contiennent pas de notions nouvelles ; elles ont pour but de rendre cohérentes des parties du programme ou de fluidifier l'expression de ce qui est le plus important, la compréhension des problèmes et de leurs solutions.*

*Ce qui est présenté ici délimite ce qui peut être exigé des étudiants ; quand il est indiqué un choix laissé à l'enseignant cela signifie qu'aucune interprétation particulière ne peut être attendue. Bien entendu l'usage correct de notions non inscrites ici doit être accepté.*

*Le temps consacré à chaque partie peut être partagé comme suit :*

- *60% pour la partie algorithmique et programmation (avec l'introduction),*
- *25% pour la partie ingénierie numérique,*
- *15% pour la partie bases de données.*

# 1 Introduction

## 1.1 Présentation

Les étudiants doivent savoir manipuler en mode "utilisateur" les principales fonctions d'un système d'exploitation et d'un environnement de développement.

## 1.2 Représentation des nombres

- Représentation des entiers en binaire,
  les entiers positifs représentables sur $n$ bits sont $\{0, 1, \ldots, 2^n - 1\}$.
  Les entiers signés représentables sur $n$ bits sont compris entre $-2^{n-1}$ et $2^{n-1}$ sans que la borne incluse soit spécifiée.
- Représentation des nombres réels en mémoire, écriture en virgule flottante normalisée sans les cas particuliers.

Les étudiants doivent connaître le principe des représentations et surtout les limites de celles-ci ; **aucune** norme précise n'est exigible.

# 2 Algorithmique et programmation

Cette partie est celle présentée dans les deux années, les parties spécifiquement introduites en seconde année sont signalées.
L'accent est porté sur le développement raisonné d'algorithmes, leur implantation dans un langage de programmation n'intervient qu'après une présentation organisée de la solution algorithmique, indépendante du langage choisi.
L'écriture des algorithmes doit être modulaire, structurée et commentée.

## 2.1 Outils

L'enseignement se fonde sur un environnement de programmation (langage et bibliothèques) basé sur le langage Python (la version n'est pas imposée). Un environnement de développement efficace doit être choisi et utilisé ; le choix est laissé à chaque enseignant, il n'y a pas d'environnement imposé [1].
L'usage des bibliothèques est présenté, principalement lors des séances de TP, mais **"Aucune connaissance des fonctions des bibliothèques n'est exigible des étudiants"**.
L'étude approfondie de ces divers outils et environnements n'est pas une fin en soi et n'est pas un attendu du programme.

## 2.2 Types de données

**Aucun** autre type de données [2] que ceux énumérés ci-après ne fait partie des attendus du programme.

### 2.2.1 Types simples

- Notion de variable, type et valeur d'une variable.
- Opérations usuelles.
- Différence entre expression et instruction.
- Affectation.

Les types simples à connaître sont

**entiers** : `int, +, -, *, //, %, **`,

**flottants** : `float, +, -, *, /, **`, les fonctions usuelles sont dans le module `math` ou `numpy`,

**booléens** : `bool, and, or, not`. Il n'y a pas de fonction logique bit-à-bit dans le programme.

1. Dans la pratique les logiciels employés majoritairement sont Pyzo/IEP, Spyder et Idle.
Les versions de Python les plus utilisées sont des versions de Python3.
2. Par exemple les dictionnaires ou les files.

### 2.2.2 Tableaux/listes

Les deux notions sont implémentées par le même type (`list`) en Python.
Il n'y a pas lieu de présenter les listes et tableaux abstraits.
La notion de liste chaînée n'est pas au programme.

- Les méthodes de création de tableaux/listes sont librement choisies par l'enseignant.
- Accès à un élément `t[i]`, en lecture et en écriture.
- Longueur `len(t)`.
- Ajout d'un élément : la méthode `append`.
- Suppression d'un élément : les méthodes utilisées sont librement choisies par l'enseignant.
- Extraction d'une partie d'un tableau/liste `t[2:4]`.
- Les tableaux à deux dimensions sont introduits.
- Au moment de l'introduction de la bibliothèque `numpy` on présente les tableaux numpy de dimension 1 ou 2 et leurs différences avec les tableaux/listes.

La concaténation de deux tableaux/listes par l'opérateur `+` n'est pas un attendu du programme. L'ajout d'éléments successifs doit être présenté avec la méthode `append` et non par l'addition d'un tableau/liste à un élément.
La répétition d'un tableau/liste par le produit de celui-ci et d'un entier n'est pas attendu du programme. Cependant il est possible de présenter la construction d'un tableau/liste conteneur par la construction `[a]*n`.

### 2.2.3 Chaînes de caractères

- Création, `ch = "abcd"`.
- Accès à un élément `ch[i]`, en lecture seulement.
- Longueur `len(ch)`.
- Concaténation `ch1+ch2`.
- Extraction d'une sous-chaîne `ch[2:4]`.

### 2.2.4 Fichiers

- Notion de chemin d'accès.
- Accès à un fichier.
- Lecture depuis un fichier : les méthodes utilisées sont librement choisies par l'enseignant. Fonctions `int` et `float` de construction d'un entier ou d'un flottant à partir d'une chaîne de caractères.
- Écriture dans un fichier. Fonction `str` de construction d'une chaîne de caractères à partir de nombres.

### 2.2.5 Piles (seconde année)

La notion de pile est présentée principalement en raison de son utilisation par la pile d'appels d'une fonction récursive.
Présentation abstraite, fonctions **push** et **pop**.

Implémentation à l'aide de tableaux/listes Python : méthodes `append` et `pop`.

## 2.3 Instructions

- Affectation : `=`.L'affectation simultanée pourra être utilisée, notamment lorsque le résultat d'une fonction se compose de plusieurs valeurs. Aucune connaissance théorique des tuples (n-uplets) n'est cependant exigible
- Affichage : fonction `print`. Aucune connaissance du formatage de l'impression n'est exigible.
- Instructions conditionnelles : `if` et `else`.
  `elif` n'est pas explicitement au programme.
- Instructions itératives : les deux structures doivent être connues.
  1. boucles itératives : `for x in s` où `s` est une structure composée, c'est-à-dire un tableau/liste, une chaîne de caractère, un `range`, un fichier, etc
  2. boucles conditionnelles : `while condition`

  Les énumérateurs ne sont pas au programme.
  Il est possible de sortir d'une boucle par un `return` dans une fonction [3]
- Fonctions.
  — Paramètres et résultats.
  — Portées des variables.
  — Fonctions récursives (seconde année).
  — L'usage de variables globales est déconseillé.

## 2.4 Analyse d'algorithmes

- Les notions de complexité des algorithmes est introduite sur des exemples simples. Les complexités en mémoire ou en temps dans le pire ou le meilleur des cas sont au programme. La complexité moyenne n'est pas au programme.
- La notion d'invariant de boucle est introduite pour s'assurer de la correction de segments itératifs.
- Notion de terminaison d'une boucle.
- Dans le cas des algorithmes récursifs vus en seconde année le programme stipule que les étudiants doivent savoir "comprendre le fonctionnement d'un algorithme récursif" et "s'interroger sur l'efficacité temporelle d'un algorithme". Dans le prolongement des attendus du programme de première année il semble utile de présenter les notions de complexité, de preuve et de terminaison sur des exemples simples d'algorithmes récursifs.

3. Voire par un `break`.

## 2.5 Algorithmes

Le programme contient une liste d'algorithmes que les étudiants doivent savoir expliquer, programmer et modifier selon les contraintes.

### 2.5.1 Première année

- Recherche d'un élément dans un tableau/liste.
- Recherche du maximum dans un tableau/liste.
- Calcul de la moyenne, de la variance des valeurs d'un tableau/liste.
- Recherche par dichotomie dans un tableau/liste trié.
- Recherche par dichotomie d'un zéro d'une fonction continue et monotone.
- Calcul de valeur approchées d'intégrales sur un segment par la méthode des rectangles.
- Calcul de valeur approchées d'intégrales sur un segment par la méthode des trapèzes.
- Recherche d'un mot dans une chaîne de caractères : seul l'algorithme naïf de recherche du mot à chaque position est au programme.

### 2.5.2 Seconde année

- Tri dans un tableau/liste à une dimension de nombres :
  — Tri par insertion,
  — Tri rapide (quicksort),
  — Tri fusion.
- Comparaisons des complexités temporelles dans le pire ou le meilleur des cas.
- Application au calcul de la médiane.

Le programme contient une liste de thèmes : ce sont des propositions d'activités facultatives destinées à l'illustration des notion du cours. **Aucune connaissance concernant ces activités ne peut être supposée de la part des élèves.**

# 3 Ingénierie numérique et simulation

La modélisation n'est pas un objectif du programme. Seuls les aspects pratiques sont privilégiés : erreurs d'arrondi, étude empirique de l'influence du pas de discrétisation ou du nombre d'itérations, occupation en mémoire.

## 3.1 Outils

L'environnement de calcul scientifique Scilab est présenté rapidement : il peut être utilisé dans les problèmes de simulation mais ne peut en aucun cas être imposé. Pour le dire autrement dans les problèmes de simulation les candidats doivent avoir le **choix** : ils peuvent programmer en utilisant soit Scilab soit Python avec les bibliothèques numpy et scipy. Ils doivent savoir utiliser les bibliothèques de calcul numérique et la documentation associée.
**Mais aucune connaissance des fonctions des bibliothèques n'est attendue.**
Par contre le type de données des tableaux numpy, `array`, doit être présenté.

## 3.2 Problèmes étudiés

- Résolution d'équations algébriques ou transcendantes à une dimension : méthode par dichotomie, méthode de Newton.
- Calcul de valeur approchée d'une intégrale sur un segment par la méthode des rectangles(*).
- Calcul de valeur approchée d'une intégrale sur un segment par la méthode des trapèzes(*).
- Résolution d'une équation différentielle ordinaire : méthode ou schéma d'Euler. L'application de ce schéma à des systèmes plus complexes, équations d'ordre 2 ou systèmes différentiels, ou l'utilisation d'autres schémas doit être guidée.
- Résolution d'un système linéaire inversible : méthode du pivot de Gauss.

Les algorithmes notés (*) sont placés dans la section algorithmique et programmation.

# 4 Initiation aux bases de données

L'algèbre relationnelle est tout autant exigible que l'écriture de requêtes dans le langage SQL.

## 4.1 Algèbre relationnelle

**Vocabulaire** On présente et emploie le vocabulaire de l'algèbre relationnelle, relations, tuples, attributs, et le vocabulaire des bases de données, tables, entrées, champs, domaines.

**Clef** Seules les clefs primaires sont au programme. On ne parle pas de clefs étrangères.

**Opérateurs unaires élémentaires** La projection $\pi$, la sélection $\sigma$, le renommage $\rho$.

Le renommage est également utilisé pour définir de nouveaux attributs en renommant une expression sur les attributs du schéma.

**Opérations binaires** Union, intersection, différence, produit cartésien et jointure symétrique $\bowtie$.

On présente la division cartésienne sans que sa maîtrise ne soit exigible.

**L'agrégation** Agrégation ${}_{attributs}\gamma_{nom \leftarrow f(attr)}(R)$ où $f$ est une des 5 fonctions, comptage, minimum, maximum, somme et moyenne.

## 4.2 Requêtes SQL

Voici une liste exhaustive des mots clefs et opérateurs à connaître :

- `SELECT, FROM, WHERE, .. JOIN .. ON .. = .., UNION, INTERSECT, EXCEPT, OR, AND, AS` (en renommage d'attribut, comme en nommage)
- `GROUP BY, MAX, MIN, COUNT, SUM, AVG, *`
- les opérateurs servant à définir des expressions sur les champs et sur les résultats des fonctions de groupage : + * - /
- les opérateurs de comparaison > < >= <= =!=.

Les requêtes imbriquées sont présentées également en SQL. En particulier, on indique que le résultat d'une requête imbriquée en SQL qui ne fournit qu'un champ et qu'une entrée peut être identifié avec son unique valeur et utilisé comme tel. Pour la clarté de la rédaction on pourra présenter les requêtes complexes sous la forme
`soustable = SELECT ...` puis
`SELECT ... FROM soustable WHERE ...`

Les autres commandes SQL ne font pas partie du programme.

- On peut présenter d'autres opérateurs afin de simplifier les requêtes comme `DISTINCT`, `IN`, `HAVING`, .... Mais ce ne sont pas des attendus du programme.
- La présentation des cas particuliers de jointures symétriques, `... NATURAL JOIN ...` et `... JOIN ... USING ...`, n'est pas utile.
- Les autres jointures (`LEFT, RIGHT, INNER, OUTER JOIN`) sont à proscrire.
- Les conversions de type, le traitement des dates, ... ne sont pas au programme.