Introduction

UE Informatique, M1 Mathématiques et applications,
EADS, Université d'Aix-Marseille *

2024–2025

1 Faisons les présentations

Le cours est assuré par Lionel Vaux Auclair [1] (le « je » dans la suite du document).

Je suis joignable par courriel (lionel *point* vaux *chez* univ-amu *point* fr), mais la méthode recommandée pour prendre contact pour toute question portant sur le cours est d'écrire sur le forum de discussion sur la page Amétice de l'UE. [2]

2 Contenu et organisation du cours

Ce cours est un cours d'*informatique* dans un cursus de *mathématiques*. On y mobilise donc des concepts et outils de nature informatique (notamment le codage de l'information et la programmation), mais on ne s'interdit pas de s'appuyer sur des connaissances mathématiques (en arithmétique et en algèbre notamment) et les domaines d'application sont à l'intersection des deux disciplines (algorithmique, traitement des données, logique et calculabilité).

Le langage de programmation utilisé pour cet enseignement est Python : il sera donc nécessaire d'en connaître ou acquérir les bases (le niveau attendu est celui de la fin de L3 dans la licence de mathématiques d'AMU) pour suivre les exemples et résoudre certains exercices. Toutefois, les programmes plus conséquents que vous soumettrez pour les évaluations peuvent être écrits dans un autre langage, sous réserve d'acceptation préalable de ma part.

Une partie importante du travail à fournir pour progresser dans l'UE et réussir les évaluations consiste à lire, modifier ou écrire des programmes. Il est de votre ressort de trouver le cadre logiciel qui vous convient le mieux pour cela, mais une solution par défaut peut être l'utilisation de Spyder [3] qui fournit un environnement de développement assez standard en Python.

Qu'il s'agisse de programmation ou d'exercices classiques sur papier, il est important de traiter *tous* les exercices proposés. En cas de difficulté, je vous encourage à poser vos questions et discuter sur le forum : si c'est difficile pour vous ça l'est sans doute pour vos camarades, et tout le monde pourra profiter des éléments supplémentaires issus de la discussion. La participation active à ces échanges sera prise en compte favorablement dans la note de contrôle continu.

1. https://www.i2m.univ-amu.fr/perso/lionel.vaux/
2. https://ametice.univ-amu.fr/course/view.php?id=124246
3. https://www.spyder-ide.org/

2.1 Objectifs

À l'issue de ce cours, on s'attend à ce que les étudiantes et étudiants soient capables de :

— prévoir, modifier et corriger le comportement d'un programme Python de taille raisonnable (quelques lignes ou dizaines de lignes de code) prétendant résoudre une tâche algorithmique ;

— mobiliser leurs connaissances pour résoudre un problème algorithmique portant sur des données brutes (nombres, textes) ou structurées (arbres, graphes), et transcrire cette solution en un programme (en Python ou possiblement dans un autre langage de leur choix) ;

— proposer ou critiquer des choix d'implémentation pour cette dernière tâche : types de données, structuration du programme, appel à des bibliothèques externes, etc.

— mettre en évidence les limites de certains algorithmes ou programmes (notamment, estimer leur complexité en temps et en espace) ;

— identifier des obstacles théoriques à la résolution de certains problèmes grâce à des résultats mathématiques (théorèmes de Rice).

2.2 Calendrier

On suit le calendrier commun du M1 EADS :

— Envoi 1 : 14 octobre (Cours + TD 1)

— Envoi 2 : 2 décembre (Cours + TD 2 + Corrigé du TD 1)

— Envoi 3 : 27 janvier (Cours + TD 3 + Corrigé du TD 2 + DCC 1)

— Envoi 4 : 17 mars (Cours + TD 4 + Corrigé du TD 3 + DCC 2 + Corrigé du DCC 1)

— Envoi 5 : 28 avril (Corrigé du TD4 + Corrigé du DCC 2)

où il faut comprendre que les *Cours + TD* sont représentés par le document principal du cours, qui comprend plusieurs séries d'exercices à réaliser pour consolider les notions ; et que les *DCC* sont des devoirs constitués d'exercices plus longs, à réaliser en autonomie, et qui serviront pour l'établissement des notes de contrôle continu.

3 Modalités d'évaluation

Au cours du semestre, vous aurez à rendre deux devoirs de contrôle continu (DCC) qui comprendront chacun une partie programmation conséquente. La méthode recommandée pour le rendu des devoirs est le dépôt électronique directement sur Amétice, incluant des fichiers sources séparés. [4] Il est aussi possible de m'envoyer directement les fichiers, de préférence *via* une plateforme de dépôt. [5]

L'examen terminal (ET) sera un examen classique en fin d'année. La formule pour le calcul de la note est $\max((4ET + CC)/5, ET)$ où CC est la moyenne des deux DCC, possiblement augmentée d'un petit coup de pouce en cas de participation active tout au long du semestre (le contrôle continu ne peut qu'améliorer la note de l'examen).

Même si les DCC influent peu sur la note finale, ils sont conçus pour participer à votre formation en vous proposant des problèmes un peu plus riches que les exercices de TD.

4. Si toutefois vous avez déjà l'habitude d'utiliser Git, vous pouvez aussi me donner un accès en lecture à un dépôt contenant votre travail. Notez d'ailleurs que l'université vous fournit une instance de Gitlab, accessible avec votre compte AMU : `https://etulab.univ-amu.fr/`. (Si vous ne comprenez pas de quoi parle cette note de bas de page, vous pouvez l'ignorer sans risque.)

5. Je vous rappelle qu'AMU met à votre disposition de nombreux services numériques, dont un stockage en ligne et un service d'envoi de fichiers : `https://dirnum.univ-amu.fr/fr/catalogue-services#tab-2551`.

4 Pré-requis et références

Côté mathématique, on s'appuie sur les compétences de base en raisonnement (notamment les différentes formes de récurrence) et calcul, et les connaissances attendues en fin de licence de mathématiques, notamment en arithmétique (division euclidienne, PGCD, primalité, arithmétique modulaire, structure d'anneau de $\mathbf{Z}/n\mathbf{Z}$, structure de corps de $\mathbf{Z}/p\mathbf{Z}$ pour p premier, *etc.*) et en algèbre (les différentes structures algébriques de groupe, anneau, corps, espace vectoriel, *etc.*, et leurs propriétés usuelles).

En informatique, et notamment pour la programmation en Python, le niveau attendu est celui atteint en fin de la licence de mathématiques d'AMU. C'est-à-dire que, même s'il ne s'agit pas de se lancer dans un grand projet de développement de plusieurs milliers de lignes de code, de communiquer sur le réseau, ou de réaliser une interface graphique, on doit posséder les bases du langage :

— maîtriser l'algorithmique de base (calculs, affectations, itération et boucles) ;
— manipuler les chaînes de caractères (`str`) et les types composés usuels (au moins `tuple`, `list`, `dict`) ;
— définir et utiliser des fonctions ;
— comprendre le modèle mémoire au moins de manière élémentaire (c'est-à-dire comprendre comment fonctionne l'affectation d'une valeur à une variable, ou à un élément d'une liste, et avoir une idée de l'ordre d'évaluation des éléments d'une expression) ;
— définir et instancier des classes ;
— importer des modules externes ;
— chercher des informations dans la documentation de la bibliothèque standard.

En cas de lacune, il est de la responsabilité de l'étudiante ou l'étudiant de se documenter et, si la difficulté persiste, de prendre contact avec l'enseignant pour demander de l'aide.

Ces pré-requis étant satisfaits, les supports de cours sont conçus pour être auto-suffisants. Toutefois, si vous souhaitez aller plus loin, on trouve d'excellents livres en français, dont le très classique CLRS : Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest et Clifford Stein, *Algorithmique : cours avec 957 exercices et 158 problèmes*, Paris, Dunod, 2010, 3e éd., xxix+1188 (ISBN 978-2-10-054526-1).