

Méthodologie de la programmation



Chapitre 0

Présentation du cours

1. Présentations

2. Langages, Paradigmes et Outils

3. Évaluation et attendus

4. Take away

Organisation du cours

10 semaines, 45h de cours

- ▶ 1h30 **sans machine**
- ▶ 3h mise en pratique **sur machine**

en cas d'absence, prévenir le plus tôt possible.

Tour de table

- ▶ ingénieure en informatique, recherche en traitement automatique des langues
- ▶ <https://alicemillour.github.io/teaching>
- ▶ bureau : A179, am@up8.edu (24h de délai, relance si pas de réponse)

et vous ?

pourquoi cette licence ? expérience de l'informatique ?

Méthodologie de la programmation

apprendre à apprendre à programmer

- ▶ introduction à certains **langages** de programmation
- ▶ introduction à certains **outils** essentiels

notions nouvelles → mots nouveaux : [posez des questions](#)

Informatique

1. <https://fr.wiktionary.org/wiki/informatique>.

Informatique

*Science du traitement autom**atique** de l'**information**.*¹

mot utilisé à partir des années **1960**

1. <https://fr.wiktionary.org/wiki/informatique>.

Ordinateur

2. <https://fr.wiktionary.org/wiki/ordinateur>.

Ordinateur

Du latin *ordinator* : « celui qui met de l'ordre, ordonnateur », Jacques Perret
1955

*Appareil électronique capable, en appliquant des instructions prédéfinies (**programme**), d'effectuer des traitements automatisés de données et d'interagir avec l'environnement grâce à des périphériques (écran, clavier...).*²

2. <https://fr.wiktionary.org/wiki/ordinateur>.

Programmer : faire faire une tâche à l'ordinateur

Algorithme vs. programme

Algorithme vs. programme

1. **définition de la tâche** à résoudre (données \Rightarrow résultat)
« je veux déterminer si un nombre n entre 1 et 100 est divisible par 3 »

Algorithme vs. programme

1. **définition de la tâche** à résoudre (données \Rightarrow résultat)
« je veux déterminer si un nombre n entre 1 et 100 est divisible par 3 »
2. **analyse du problème & conception d'un algorithme**
découpage séquentiel de la tâche et suite d'instructions *en pseudo-langage*
 1. lister les multiples de 3 ($L = [3, 6, 9, \dots, 99]$)
 2. regarder si mon nombre est présent dans la liste L
 3. afficher le résultat... ou encore
 1. calculer le reste de la division euclidienne de n par 3
 2. regarder si le reste est égal à 0
 3. afficher le résultat

Algorithme vs. programme

1. **définition de la tâche** à résoudre (données \Rightarrow résultat)
« je veux déterminer si un nombre n entre 1 et 100 est divisible par 3 »
2. **analyse du problème & conception d'un algorithme**
découpage séquentiel de la tâche et suite d'instructions *en pseudo-langage*
 1. lister les multiples de 3 ($L = [3, 6, 9, \dots, 99]$)
 2. regarder si mon nombre est présent dans la liste L
 3. afficher le résultat... ou encore
 1. calculer le reste de la division euclidienne de n par 3
 2. regarder si le reste est égal à 0
 3. afficher le résultat
3. **écriture du programme** traduction de l'algorithme dans un certain *langage de programmation*

Algorithme vs. programme

1. **définition de la tâche** à résoudre (données \Rightarrow résultat)
« je veux déterminer si un nombre n entre 1 et 100 est divisible par 3 »
2. **analyse du problème & conception d'un algorithme**
découpage séquentiel de la tâche et suite d'instructions *en pseudo-langage*
 1. lister les multiples de 3 ($L = [3, 6, 9, \dots, 99]$)
 2. regarder si mon nombre est présent dans la liste L
 3. afficher le résultat... ou encore
 1. calculer le reste de la division euclidienne de n par 3
 2. regarder si le reste est égal à 0
 3. afficher le résultat
3. **écriture du programme** traduction de l'algorithme dans un certain *langage de programmation*

Algorithme vs. programme

1. **définition du problème** à résoudre (données \Rightarrow résultat)
 2. **analyse du problème & conception d'un algorithme**
découpage séquentiel de la tâche et suite d'instructions *en pseudo-langage*
 3. **écriture du programme** traduction de l'algorithme dans un certain *langage de programmation*
-
- ▶ les étapes 1 et 2 se font **sans** la machine
 - ▶ un problème peut être résolu par différents algorithmes plus ou moins rapides ou efficaces (autre ex. "aller d'un point A à un point B")
 - ▶ un algorithme peut être traduit dans différents langages de programmation

1. Présentations

2. Langages, Paradigmes et Outils

3. Évaluation et attendus

4. Take away

Langages

langage de programmation vs. langage naturel

Points communs

- ▶ suivent un **lexique**, une **syntaxe** et une **sémantique** particulière
- ▶ on n'apprend pas par « essai erreur » : il faut comprendre la logique et **pratiquer** pour être à l'aise

Différences

- ▶ la machine n'a aucune capacité d'invention : une erreur rend le programme **incompréhensible** (analyse sémantique vérifie la *validité* du code)

Langages de programmation

langage interprété vs. langage compilé

► langage *interprété*

1. **code source** + **données d'entrée** → **interpréteur** → **données de sortie**

- traduction en code machine « à la volée » (++) temps d'exécution)
- Ex : Python, Javascript, PHP, Ruby, etc.

► langage *compilé*

1. **code source** → **compilateur** → **code binaire** (fichier *exécutable*)

2. **code binaire** + **données d'entrée** → **SE** → **données de sortie**

- traduction du code *une fois pour toutes* (— temps d'exécution)
- indiqué pour les problématiques de temps réel (JV, aérospatiale, SE)
- Ex : C, C++, Ada, etc.

la différence ne tient pas tant aux langages eux-mêmes mais à *la manière*
dont on les utilise

code source = le(les) source(s) = fichier(s) texte(s) écrit(s) dans
un ou plusieurs langages de programmation

ex : Ctrl-U sur une page web

implémentation des deux algorithmes en Python et en C
(démon)

implémentation des deux algorithmes en Python et en C
(démonstration)

quel est l'interpréteur Python ? Quel est le compilateur C utilisé ?

Paradigme de programmation

modèle théorique qui oriente la façon de concevoir un programme (types d'objets manipulés, structures de contrôle, etc.)

- ▶ concurrente,
- ▶ déclarative,
- ▶ fonctionnelle,
- ▶ impérative,
- ▶ logique,
- ▶ objet,
- ▶ par contrainte,
- ▶ synchrone,
- ▶ événementielle, etc.

Langages utilisés dans ce cours

Ce cours aborde l'informatique au travers de la programmation dans différents langages :

- ▶ Bash (voir le cours de Pratique des Machines)
- ▶ Python
- ▶ C
- ▶ L^AT_EX

Outils

- ▶ documentations,
- ▶ interpréteurs,
- ▶ compilateurs,
- ▶ moteurs de production,
- ▶ gestionnaires de contrôle de version,
- ▶ gestionnaire de paquets,
- ▶ débogueurs,
- ▶ profileurs.

1. Présentations

2. Langages, Paradigmes et Outils

3. Évaluation et attendus

4. Take away

Prenez de bonnes habitudes

...à commencer par vos messages / mails

- ▶ champ objet précis (nom du cours),
- ▶ définissez une **signature** (nom complet, groupe),
- ▶ rien de confidentiel (pensez que c'est une carte postale),
- ▶ ne pas utiliser les majuscule mais *ceci* pour mettre en valeur un mot ou une phrase,
- ▶ pas de langage SMS,
- ▶ soyez bref(ve) et courtois(e),
- ▶ évitez l'humour et l'ironie (même avec un smiley),
- ▶ soyez modéré(e)s dans vos propos : vos messages restent,
- ▶ limitez, si possible, à un sujet le contenu de vos messages,
- ▶ ne transmettez pas un courrier reçu à d'autres personnes sans l'autorisation de l'émetteur,
- ▶ limitez la taille et le nombre de vos pièces jointes (privilégier un lien).

Attendus

- ▶ Correctement écrire du code :
 - indentation propre,
 - nommage cohérent,
 - organisation modulaire des fichiers.
- ▶ Savoir gérer un projet seul ou à plusieurs :
 - utiliser git pour gérer les versions du code source et la collaboration,
 - utiliser Make pour gérer la compilation modulaire.
- ▶ Savoir utiliser les bons outils :
 - connaître les forces et faiblesses de différents langages et paradigmes de programmation,
 - savoir chercher efficacement dans une documentation,
 - utiliser un débogueur voire un profileur.
- ▶ Savoir communiquer sur vos projets :
 - utiliser \LaTeX pour écrire des rapports.

Évaluation

- ▶ Votre évaluation pour ce cours prendra en compte (dans l'ordre d'importance) :
 - le projet,
 - les TPs (potentiellement tous les TPs seront notés!).
- ▶ La propreté du code (**nommage, indentation, organisation**) est importante et sera prise en compte autant pour les TP que pour le projet.

1. Présentations

2. Langages, Paradigmes et Outils

3. Évaluation et attendus

4. Take away

Take away

- ▶ algorithme \neq programme,
- ▶ comprendre le problème à résoudre \neq concevoir un algorithme \neq traduire vers un langage de programmation
progresser commence par comprendre là où ça coince,
- ▶ apprendre à programmer \neq apprendre un langage de programmation.

+ les bonnes pratiques à maîtriser font partie des critères d'évaluation