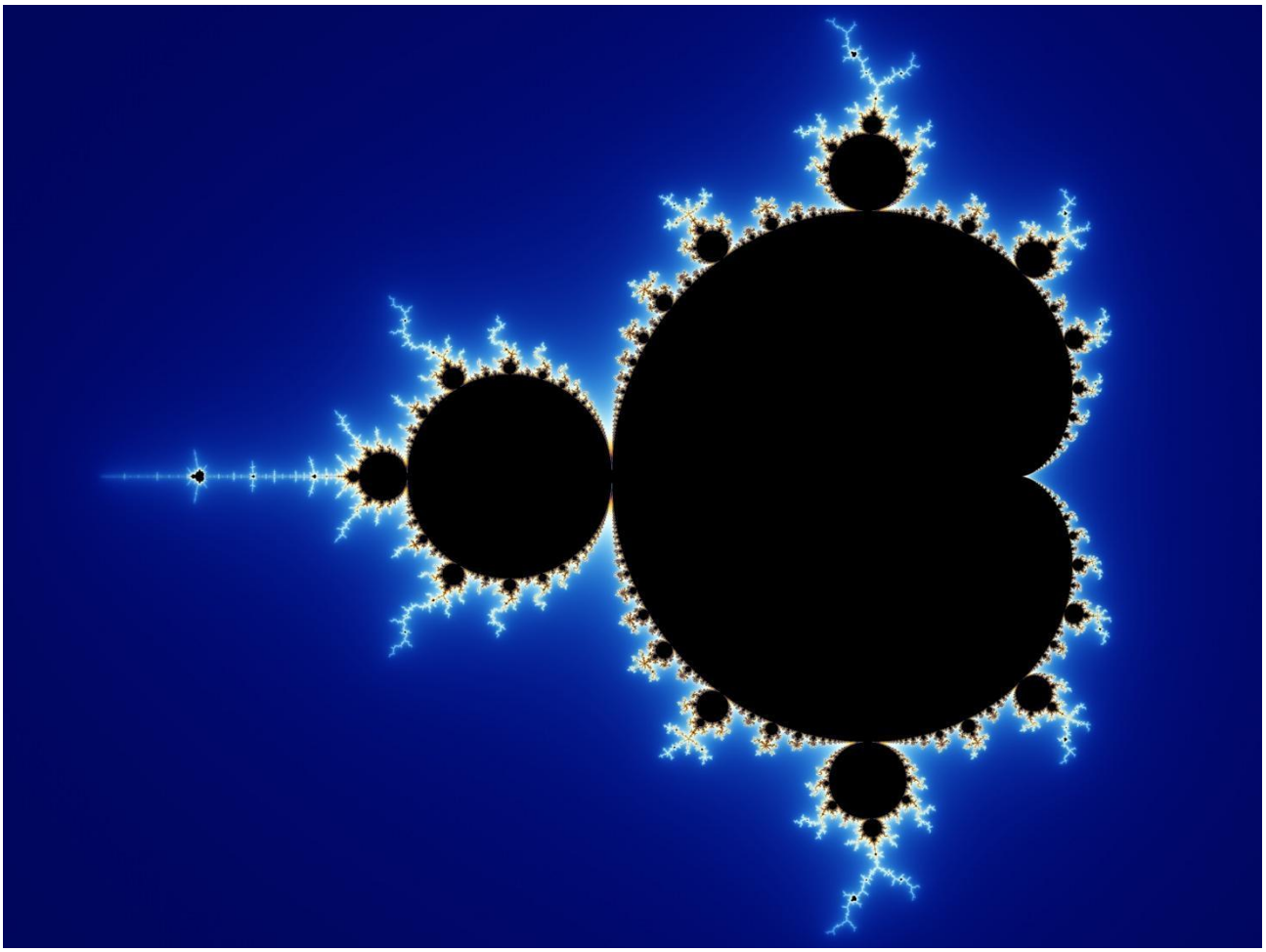
Algorithmique

KIT APPRENANT

Ce projet est cofinancé par l'Union européenne
L'EUROPE S'ENGAGE en région Auvergne-Rhône-Alpes avec le FSE
La Région Auvergne-Rhône-Alpes



L'ensemble de Mandelbrot (source image : Wikipedia)

OBJECTIFS

Objectif Global

L'algorithmique est l'art de décrire formellement la solution à une problématique donnée. Un algorithme a comme qualités principales : l'assurance de terminaison (fin d'exécution), la justesse dans la réponse donnée (correction) et la compatibilité avec toute donnée d'entrée pour laquelle il a été conçu (complétude).

L'algorithmique est un des socles de l'informatique. Même s'il est possible de coder et d'exécuter un programme sans passer par l'algorithmique, celle-ci est indispensable pour répondre aux questions suivantes : la solution est-elle lisible ? portable ? réutilisable ? efficace ?

Avoir des notions d'algorithmique est donc essentiel pour toute personne désirant évoluer dans les métiers liés à la programmation informatique.

Objectifs pédagogiques

Le codage n'est que l'étape finale d'un processus de résolution d'un problème donné. L'objectif de ce module est de vous donner une vision complète de ce processus.

À la fin de ce module, l'objectif est que vous soyez en mesure de :

1. Mettre en place une méthodologie de travail : spécification, conception (préliminaire, détaillée), codage
2. Décrire des algorithmes à l'aide de pseudocode
3. Évaluer la complexité d'un algorithme
4. Créer une bibliothèque Python sous la forme d'un module Python

Démarche pédagogique

Nous allons étudier l'algorithmique au travers d'une problématique donnée : le tri. Ce problème a déjà été largement exploré et nous allons en profiter pour nous approprier tous les enseignements qui découlent de ces années de recherches.

Dans un premier temps, nous allons acquérir un peu de méthodologie en voyant quelles sont les étapes de création d'un algorithme. Nous nous attarderons en particulier sur l'utilisation du pseudo-code et la notion de complexité.

Ensuite, la seconde itération nous permettra de consolider ces notions, en étudiant un nouveau concept : la récursivité. Nous mesurerons aussi la complexité de tous les algorithmes implémentés jusque-là afin de les comparer.

Et s'il vous reste du temps et de l'énergie, vous pourrez essayer d'appliquer ce que vous aurez appris sur d'autres problèmes dans l'itération 3. Elle est entièrement facultative et il est recommandé de la démarrer uniquement si vous avez bien compris tous les concepts et que vous avez validé toutes les compétences.

COMPÉTENCES

Les compétences à acquérir sont au nombre de **3**. Voici la liste :

- Méthodologie de conception d'une solution informatique
- Analyse de la complexité d'un algorithme
- Créer une bibliothèque en python

Le détail des compétences se trouve sur Campus Skills. N'hésitez pas à y jeter un œil pour savoir comment valider ces compétences.

MODALITÉS

Durée

3 jours, soit 21 heures au total.
Lancement le 08/03/2023 et clôture le 10/03/2023.

Formateurs

Raphaël Bacher
Timothée Gerber & Arturo Mondragon (rédacteurs du sujet original)

ITÉRATION 1

Introduction à l'algorithmique

1.1 — Un peu de théorie

1h — Individuel

Lisez la ressource ci-dessous. N'hésitez pas à faire vos propres recherches s'il y a des notions que vous ne comprenez pas.

RESSOURCES

- Un cours qui pose les bases de l'algorithmique
  https://openclassrooms.com/fr/courses/7527306-decouvrez-le-fonctionnement-des-algorithmes

Une approche plus théorique pour ceux qui préfèrent :
https://moodle.insa-rouen.fr/pluginfile.php/161180/mod_resource/content/1/Rappels.pdf

---

1.2 — Place à la pratique

2h — Par îlot

Un jeu de cartes et un problème vous seront donnés. Travaillez en équipe, par îlot, afin de proposer une conception préliminaire et détaillée (c.à.d. écrivez les signatures, et le pseudocode associé à votre solution) suite à l'analyse du problème.

⚠ *Le langage pseudocode n'a pas été normalisé, à vous de choisir la norme qui rend votre algorithme le plus lisible, compréhensible pour vous et auprès des autres.*

⚠ *Il est conseillé de réaliser votre conception détaillée avec une numération basée sur l'index zéro. Cela vous sera utile lors du passage au codage, car Python est un langage qui utilise cette convention.*

RESSOURCES

- Un pense-bête pour le langage pseudocode
  https://www.dropbox.com/s/rvpbmqsp6ty649i/Algorithmique-2012-SynthesePC.pdf?dl=0
- Pourquoi indexer des éléments d'un tableau à partir de zéro ? Voici une lettre écrite par le mathématicien et informaticien Dijkstra en 1982. Dijkstra est notamment reconnu pour ses travaux sur les graphes. Un algorithme de recherche du plus court chemin porte son nom.
  https://www.cs.utexas.edu/users/EWD/ewd08xx/EWD831.PDF

- Le commentaire de l'utilisateur qblock résume la lettre de Dijkstra https://www.reddit.com/r/programming/comments/elr06/dijkstra_why_numbering_should_start_at_zero/
- Article wiki sur la numération basée sur l'index zéro https://en.wikipedia.org/wiki/Zero-based_numbering

COMPÉTENCES ASSOCIÉES

- Méthodologie de conception d'une solution informatique

Livrables

➔ Le pseudocode répondant au problème posé (à mettre en commentaire de la compétence associée sur Campus Skills)

1.3 — En Python, ça donne quoi ?

2h — Individuel

Après validation par le formateur de votre conception, réalisez le codage de votre algorithme avec le langage Python.

1.4 — Les tris classiques

5h — Individuel

Récupérez le jupyter notebook `AlgosTriClassiques.ipynb`. Vous allez maintenant programmer quelques algorithmes de tri classiques :

- Bubble sort
- Insertion sort
- Selection sort

Pour chaque algorithme, êtes-vous capable de :

- Déterminer si votre méthode est une fonction ou une procédure ?
- Calculer la complexité O de chaque algorithme ?

- Analyser l'exécution de vos algorithmes à l'aide des « magic commands ». `%prun` ou `%lprun` pourrait être utile pour identifier la complexité.

RESSOURCES

- Cours d'introduction aux tableaux et au tri https://moodle.insa-rouen.fr/pluginfile.php/162384/mod_resource/content/1/Tableau.pdf
- Visualisation des algorithmes de tri https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
- La notion de complexité (à partir de la diapositive 44) https://moodle.insa-rouen.fr/pluginfile.php/161180/mod_resource/content/1/Rappels.pdf
- La notation grand O https://www.youtube.com/watch?v=v4cd1O4zkGw
- Les « magic commands » https://towardsdatascience.com/magic-commands-for-profiling-in-jupyter-notebook-d2ef00e29a63?gi=b75673240d5d

COMPÉTENCES ASSOCIÉES

- Analyse de la complexité d'un algorithme