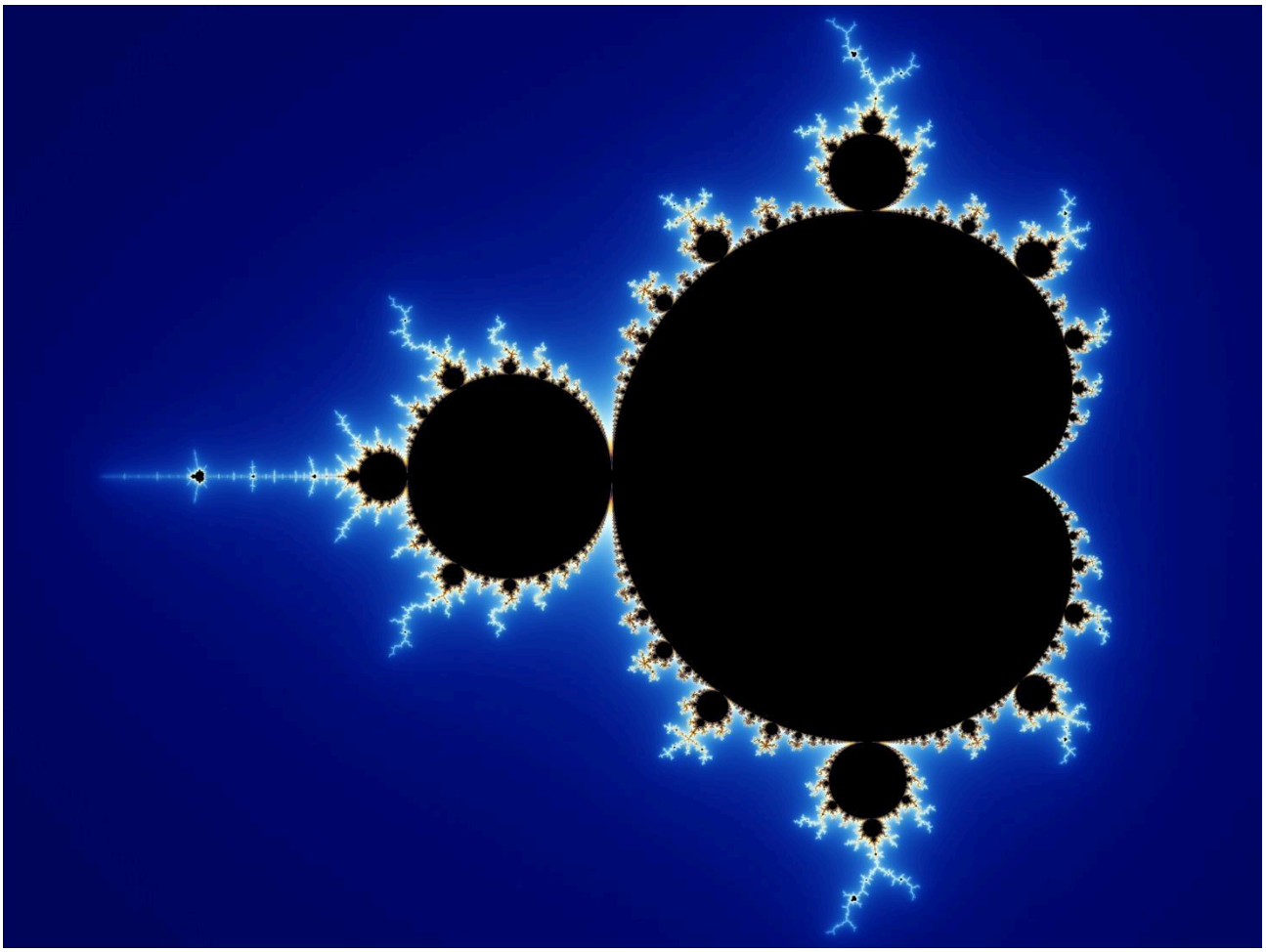
Algorithmique

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
le campus numérique in the ALPS



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**

Critères : Ecrire et lire du pseudo-code
Preuve de travail :
- 1. Écrire du pseudo-code : vous avez écrit le pseudo-code de **l'itération 1.2**
- 2. Lire du pseudo-code : vous triez le paquet de cartes distribué en utilisant le pseudo-code du tri par insertion **(itération 1.4)**

- **Analyse de la complexité d'un algorithme**

Critères : Connaître quelques critères d'évaluation de la complexité d'un algorithme
Preuve de travail :
- 1. Expliquer au formateur la notation "O"
- 2. Évaluer la complexité O du tri par insertion et comparer avec la sortie de la commande magique %prun

- **Créer une bibliothèque en python**

Critères : Organisation de votre code sous forme de module afin de les rendre maintenables et réutilisables
Preuve de travail :
- 1. Vous avez crée un fichier .py et vous l'avez importé avec succès dans un notebook
- 2. Vous savez importer de trois manières différentes la fonction "bubble_sort"

MODALITÉS

Durée

3 jours, soit 21 heures au total.
Lancement le 19/03/2025 et clôture le 21/03/2025.

Formateurs

Jours 1 & 2 > Raph
Jour 3 > Matha

Timothée Gerber & Arturo Mondragon (rédacteurs du sujet original)