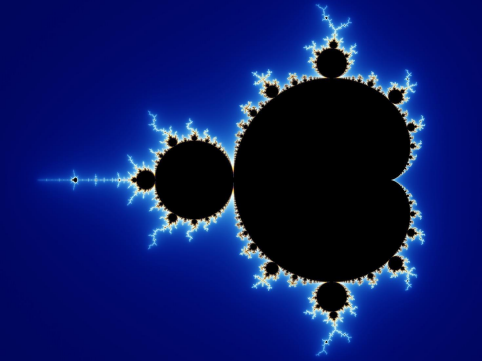
**DATA ANALYST 2024**

Algorithmique 

KIT APPRENANT



**L’ensemble de Mandelbrot**(source image : Wikipedia)

1/14

**DATA ANALYST 2024**

Objectif Global

Algorithmique KIT APPRENANT 

OBJECTIFS

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.

2/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

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.

3/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

Compétences développées

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.

4/14

**DATA ANALYST**

**2024**

Durée

3 jours, soit 21 heures au total.

Algorithmique KIT APPRENANT 

MODALITÉS

Lancement le 21/03/2024 et clôture le 24/03/2024.

Formateurs

Cyril François

(basé sur le travail d’Arturo Mondragon, rédacteur du sujet original)

5/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

**ITÉRATION 1**

Introduction à l’algorithmique

**1.1 — Un peu de théorie 1h** — Individuel

Lisez la ressource ci-dessous. Arrêtez-vous à la diapositive 43 (incluse). 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://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 voussera 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/

6/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

- 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.

7/14

**DATA ANALYST 2024**

RESSOURCES

Algorithmique KIT APPRENANT 

- 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-d2ef00e2 9a63?gi=b75673240d5d

COMPÉTENCES ASSOCIÉES

- Analyse de la complexité d'un algorithme

8/14

**DATA ANALYST 2024**

Algorithmique 

KIT APPRENANT

**ITÉRATION 2** Les tris récursifs

**2.1 — Un peu de théorie 1h** — Individuel

La récursivité est un concept simple à comprendre, mais souvent délicat à mettre en pratique. Pour découvrir la récursivité, lisez attentivement la ressource donnée.

RESSOURCES

- Cours d’introduction à la récursivité

https://moodle.insa-rouen.fr/pluginfile.php/164408/mod\_resource/content/1/07-Recursivite.pdf

**2.2 — Vous reprendrez bien un peu de théorie ? 2h** — Individuel

Vous allez maintenant utiliser le paradigme de récursivité connu sous le nom « Divide-and-conquer » pour implémenter des algorithmes de tri plus avancés : le « quick sort » et le « merge sort ».

Récupérez le jupyter notebook AlgosTriRecursifs.ipynb et regardez attentivement son contenu. Pour chaque algorithme : une vidéo (pédagogique ?) et un pseudocode vous ont été fournis.

⚠ *Sur internet vous trouverez des implémentations différentes à celle proposée dans le notebook.*

Dans un premier temps, il est conseillé de se familiariser avec les principes à partir du notebook. N’hésitez pas à consulter les ressources si nécessaire.

RESSOURCES

- Article Wikipédia pour comprendre le paradigme algorithmique « divide and conquer » : https://fr.wikipedia.org/wiki/Diviser\_pour\_r%C3%A9gner\_(informatique)

- Cours d’introduction au tri récursif

https://moodle.insa-rouen.fr/pluginfile.php/165294/mod\_resource/content/1/TrisRecursifs.pdf - Visualisation des différents algorithmes de tri https://www.youtube.com/watch?v=kPRA0W1kECg

9/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

**2.3 — Le tri fusion 1h30** — Individuel Implémenter le tri fusion.

⚠ *Vérifiez la justesse de vos conditions d’arrêt. Si un bug existe à la condition d’arrêt, votre algorithme ne répondra pas aux caractéristiques d’un bon algo : fin d’exécution. Dans ce cas, arrêtez et relancez jupyter.*

Quelques questions intéressantes à se poser :

● Quelle est la complexité O de l’algorithme ?

● Votre algorithme est-il une procédure ou une fonction ?

**2.4 — Le tri rapide 1h30** — Individuel Implémenter le tri rapide.

Quelques questions intéressantes à se poser :

● Le choix de pivot est-il optimal? Quel est l’enjeu lié à ce choix ?

● Quels autres choix de pivot seraient possibles ?(��bonus : expérimentez)

**2.5 — Création d’un module Python 1h30** — Individuel

Vous allez créer votre premier module python ! Pour commencer, récupérez le fichier sorting.py. Placez le fichier dans le dossier où se trouvent vos *notebooks*.

��*sorting.py est un fichier python quise reconnaît parson extension .py. Ce fichier propose une conception préliminaire du module, c.à.d. lessignatures des procédures/fonctionssont déclarées, mais pasleur implémentation.*

A l’aide d’un éditeur de texte (e.g. gedit, vim ou autre), remplissez chaque signature avec vos implémentations.

Créer un nouveau notebook et importer votre module à l’aide de l’instruction : from sorting import \*

10/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

Toutes les signatures qui se trouvent dans votre fichier sont désormais utilisables depuis n’importe quel *notebook*.

Quelles autres instructions d’import sont possibles ? Quels sont les avantages et inconvénients de chaque méthode d’import?

COMPÉTENCES ASSOCIÉES

- Créer une bibliothèque en python

**Livrables**

➔ Le module sorting.py (à mettre en commentaire de la compétence associée sur Campus Skills)

**2.6 — Comparaison des complexités empiriques 2h30** — Individuel

Testez différentes tailles de tableaux N comme données d’entrée de vos algorithmes de tri. Affichez les résultats sous la forme d’une figure, afin de pouvoir facilement comparer les algorithmes.

⚠ *Pour une comparaison juste, réfléchissez à une manière de tester l’efficacité de chaque algorithme avec les mêmes valeurs d’entrée.*

Quel algorithme semble être le plus efficace ?

Chaque algorithme peut être plus ou moins efficace selon l’état de désordre du tableau d’entrée. Quels sont les différents types de (dés)ordre possible ?

RESSOURCES

- Connaître l’identité d’une instance https://docs.python.org/3/library/functions.html#id COMPÉTENCES ASSOCIÉES

- Analyse de la complexité d'un algorithme

11/14

**DATA ANALYST 2024**

Algorithmique 

KIT APPRENANT

12/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

**ITÉRATION 3 (facultative)**

Plus dur, meilleur, plus rapide, plus fort

⚠ *Cette itération est facultative. Elle n’est pas nécessaire pour valider les compétences du module. C'est du bonus pour ceux qui iraient vite. Assurez-vous d'avoir bien terminé les itérations 1 et 2 et d'avoir validé toutes les compétences avant de vous lancer sur cette troisième itération.*

**3.1 — Recherches supplémentaires 1h** — Individuel

Quel type d'algorithme de tri est utilisé par la méthode sort() de Python ? Renseignez-vous sur cet algorithme, ses forces et ses faiblesses.

Une des propriétés des tris, non étudiée jusqu'à présent, est la stabilité. Qu'est-ce qu'un tri stable ?

**3.2 — Soyez dynamique lorsque vous programmez ! 2h** — Individuel

La suite de Fibonacci commence par les termes u[0] = 1 et u[1] = 1. Pour obtenir les termes suivants, la formule est u[n] = u[n-1] + u[n-2]. Proposez une fonction récursive, qui prend un entier n en entrée et qui retourne la n-ième valeur de la suite de Fibonacci.

Testez votre fonction sur les valeurs [0, 1, 2, 5, 15, 45]. Vous devez obtenir les résultats suivants :

print(fibonacci(0)) # 1

print(fibonacci(1)) # 1

print(fibonacci(2)) # 2

print(fibonacci(5)) # 8

print(fibonacci(15)) # 987

print(fibonacci(45)) # 1836311903

13/14

**DATA ANALYST 2024**

Algorithmique KIT APPRENANT 

Qu'est ce que vous remarquez ? Tout en gardant votre algorithme récursif, réfléchissez à une solution pour optimiser votre fonction. Si vous n'avez pas d'idée, demandez-moi un indice.

**3.3 — C'est en résolvant des problèmes qu'on devient un bon algorithme. Ou un truc dans le genre...**

**6h** — Individuel

Il existe des calendriers de l'avent sous de très nombreuses formes. Un développeur propose une version un peu spéciale, qui demande de résoudre chaque jour un petit problème en deux parties, à l'aide d'un algorithme de votre conception.

Essayez de résoudre le jour 14 du calendrier 2021 : https://adventofcode.com/2021/day/14

Si vous avez terrassé les deux parties du jour 14, premièrement, bravo ! Deuxièmement, cherchez une solution au jour 23 : https://adventofcode.com/2021/day/23

Si vous avez réussi à coder une solution aux jours 14 et 23, vous avez bien mérité une petite pause. Mais si vraiment vous vous ennuyez, il reste tous les autres jours...

14/14