Master IWOCS - 1ère année

UFR Sciences et Techniques - Le Havre

Simplification du flux de données AIS

Auteurs :
M. Jérémy Aubourg
M. Isaac Lefebvre

Encadrants :
Pr. Claude Duvallet
Pr. Yoann Pigné

Résumé

Dans le cadre de la validation de notre 1ère année de master d'informatique à l'Université du Havre, nous avons eu l'occasion de réaliser un TPE portant sur la réalisation d'un programme permettant la simplification du flux de données AIS. Notre objectif était de pouvoir réaliser des traitements sur des données issues d'AISHub afin de simplifier la lecture de ces derniers. Dans ce rapport, trouverez les différents outils et technologies utilisés, ainsi que les solutions implémentées.

Mots-clés

AIS | AISHub | Argparse | CSV | Git | Gitlab | Gnuplot | Informatique | Inspect | Pathlib | Polars | Python | Science des données | TPE | Visual Studio Code

Abstract

As part of the validation of our 1st master's degree of computer science at the University of Le Havre, we had the opportunity to realize a independent project on the design of a program allowing the simplification of the AIS data flow. Our objective was to be able to carry out treatments on data resulting from AISHub in order to simplify the reading of the latter. In this report, you will find the different tools and technologies used, as well as the implemented solutions.

Keywords

AIS, AISHub, Argparse, Computer science, CSV, Data science, Git, Gitlab, Gnuplot, Independent project, Inspect, Pathlib, Polars, Python, Visual Studio Code

Table des matières

Table des figures

Table des sigles et abréviations

- AIS : Automatic Identification System.
    - CLI : Command-Line Interface.
    - CSV : Comma-Separated Values.
    - GPL : General Public License.
    - GUI : Graphical User Interface.
    - IDE : Integrated Development Environment.
    - MIT : Massachusetts Institute of Technology.
    - MMSI : Maritime Mobile Service Identity.
    - PFSL : Python Software Foundation Licence.
    - RADAR : RAdio Detection And Ranging.
    - SQL : Structured Query Language.

Introduction

Depuis la fin des restrictions liées à la pandémie de Covid-19, le trafic maritime mondial tourne à plein régime. C'est une des conséquences de la mondialisation : le transport par les eaux représente 80 % des flux de matières premières et de marchandises. Comme notre planète abrite de plus en plus d'humains chaque année, le transport maritime devient de plus en plus dense. Mais comment les différents pays communiquent-ils pour pouvoir avoir différentes informations sur les navires ? Une réponse simple : AIS. Or, ce flux de données est assez conséquent pour les ports maritimes. Par exemple, le premier port mondial, Shanghai, situé en Chine, accueil pas moins de 200 navires par jour. Le traitement de ces données, permet entre autre de connaître des informations pertinentes sur ces navires :

- Le port de départ et d'arrivée, ainsi que les escales ;
- Les marchandises à décharger ;
- La gestion du parc maritime ;
- etc.

La conception d'un programme permettant l'automatisation de ces tâches est donc un atout majeur pour gagner du temps. Il ne faut pas cependant que ce programme remplace un être humain ; un programme peut se tromper et un être humain doit pouvoir toujours vérifier si le résultat du programme est adapté au besoin.

Pour une lecture plus fluide, les mots dont la définition ne sont pas données et/ou leur acronyme sont disponibles, sont en gras.

1 Présentation d'AIS

1.1 Qu'est-ce qu'AIS ?

Un **AIS** (Automatic Identification System) (voir figure 1) est un système d'échanges d'informations utilisé par les navires, les bateaux de plaisances ainsi que les postes de contrôles maritimes. Il s'agit d'un système qui permet de complémenter les informations du **RADAR (RAdio Detection And Ranging)**. Il est obligatoire en particulier pour les navires dépassant les 300 tonnes ([12]).

FIGURE 1 : Exemple d'un AIS

1.2 Les informations échangées

Un navire équipé d'un AIS permet entre autres d'envoyer les informations suivantes :

- Le numéro **MMSI** (Maritime Mobile Service Identity) : un identifiant unique de neuf chiffres ;
- Le statut de navigation : permet de connaître le statut du navire sous forme d'un nombre (amarré, en train de pêcher, en route, ...) ;
- Route sur le fond : son cap ;
- Vitesse sur le fond : sa vitesse ;
- La position : en latitude et longitude ;
- La date avec l'heure précise.

D'autres informations plus ou moins pertinentes peuvent être envoyées en fonction de la classe du navire. En particulier, un AIS sert également à envoyer des messages de détresse à travers le statut de navigation.

1.3 AISHub

AISHub (voir figure 2) ([1]) est un site internet permettant de partager et de regrouper les informations issues d'une antenne AIS soit en configurant l'antenne pour rediriger les données sur le site, soit via leur logiciel AIS Dispatcher. AISHub se veut être un site internet totalement gratuit et adapté aussi bien aux amateurs qu'aux professionnels. Il se veut également être une alternative au populaire MarineTraffic, qui lui, est payant pour accéder aux informations.

FIGURE 2 : Le logo d'AISHub

2 Le projet de TPE (Travaux Personnels Encadrés)

2.1 Expression du besoin

Le projet demandé était la réalisation d'un programme permettant de manipuler et de traiter des fichiers **CSV (Comma-Separated Values)** pour en extraire des données pertinentes à partir de fonctions. La fonction principale, permettait de réaliser le trajet d'un navire d'un point A à un point B, avec les escales. En l'état, la route donnée était uniquement pour un CSV, signifiant que pour avoir le trajet complet, il fallait « recoller » les différents CSV. D'autres fonctions auxiliaires pouvaient être disponibles comme le tri par colonne ou encore un filtre personnalisable.

Par ailleurs, le projet devait avoir une interface utilisateur pour pouvoir interagir avec ce dernier. Que ce soit en **CLI (Command-Line Interface)** ou en **GUI (Graphical User Interface)**, elle devait permettre la possibilité de choisir les fichiers à traiter en énumérant directement le fichier à traiter ou bien des dossiers ou des expressions rationnelles. Le choix de la fonction devait être également possible, pour une future extension par d'autres étudiants après notre départ.

2.2 Les contraintes

Nous travaillons avec des CSV qui ont été conçus à travers un projet précédent, nommé AIS-parser. Ce projet, permet de transposer des informations issues d'une base de données à des CSV. Originellement, ce projet a été développé en **Python** ([10]). De ce fait, une première contrainte s'est jointe : il faille que le nouveau projet soit une « continuité » de ce qui a été fait précédemment et soit non seulement simple à utiliser, mais également simple dans sa structure pour une éventuelle extension. La première contrainte est donc l'utilisation du langage Python.

Pour réaliser de la **science des données**, inutile de réinventer la roue : Python propose de nombreuses **bibliothèques** pour mener à bien cela. Cependant, **Pandas** ([8]), la bibliothèque la plus populaire, n'a pas été retenu pour ce projet. Il fallait pouvoir trouver quelque chose de nouveau, permettant entre autre de surpasser Pandas et de faire de la **parallélisation**.

3 Solutions

3.1 Outils

3.1.1 Python

Python (voir figure 3) est un langage de programmation interprété et multi-paradigme et régie sous la **licence PSFL (Python Software Foundation License)**. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est également doté d'un type dynamique fort et d'un ramasse-miettes pour la gestion de la mémoire. Python est similaire à **Perl** ou encore **Ruby**. Crée en 1991 par Guido van Rossum, Python va rapidement devenir un langage incontournable pour la science des données en raison de son nombre gigantesque de bibliothèques.

Python peut être implémenté également en **C** avec CPython, **Java** avec Jython et **C#** avec IronPython. Il a été conçu pour optimiser la productivité des développeurs. Possédant des outils de haut niveau et une syntaxe simple à utiliser, Python est apprécié par certains pédagogues pour sa facilité d'utilisation. Il est également conseillé pour les débutants en programmation pour se faire la main.

En 2021, Python était le langage le plus utilisé dans le monde selon Githut ([5]).

FIGURE 3 : Le logo de Python

3.1.2 Visual Studio Code

Visual Studio Code (voir figure 4) ([14]) est un éditeur de code **libre** et **open-source** et régie sous la **licence MIT (Massachusetts Institute of Technology)**. Développé par Microsoft et lancé en 2015, Visual Studio Code est basé sur **Electron**. Visual Studio Code présente de nombreuses fonctionnalités pour fonctionner comme un **IDE** (Integrated Development Environment) :

- un **débogueur** ;
- la mise en couleur syntaxique ;
- l'**IntelliSense** ;
- les **snippets** ;
- la refactorisation du code ;
- l'intégration de Git ;
- des extensions.

La force de Visual Studio Code se trouve principalement dans son catalogue d'extensions, permettant d'ajouter des fonctionnalités plus ou moins complexes, de la reconnaissance d'un langage de programmation à un tout nouveau débogueur ou encore l'intégration de **Maven**. D'après le sondage de 2021 de Stack Overflow ([13]), 71,06 % des sondés utilisent Visual Studio Code, devenant ainsi l'un des logiciels les plus populaires pour éditer du code.

FIGURE 4 : Le logo de Visual Studio Code

3.1.3 Git et Gitlab

Git (voir figure 5) ([4]) est un logiciel gratuit et open-source de gestion de version régie sous **licence GPL (General Public License)**. Crée par Linus Torvald et développé par la Software Freedom Conservancy en 2005, Git est le gestionnaire de version le plus populaire, avec plus de 12 millions de personnes l'utilisant en 2016. Git peut être utilisé en corrélation avec Gitlab, une plateforme **DevOps** lancé en 2014 et développé en Ruby, permettant le suivi d'un projet comme la gestion de son code, l'intégration continue ou encore le développement de méthodes agiles comme Scrum. Gitlab (voir figure 5) ([6]) est similaire à **Github**, la différence s'opérant sur le fait que Gitlab soit totalement gratuit et open-source là où Github propose des abonnements pour avoir plus d'options.

FIGURE 5 : Le logo de Git et de Gitlab

3.1.4 Gnuplot

Gnuplot (voir figure 6) ([7]) est un logiciel en ligne de commande servant à produire des représentations graphiques de fonctions numériques ou de données. Régie sous une **licence libre**, Gnuplot a été crée en 1986 par Thomas Williams et Colin Kelley.

Le programme peut être utilisé interactivement afin de produire un tracé. Il est également possible de réaliser des graphiques à partir de script gnuplot.

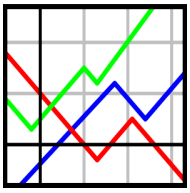

FIGURE 6 : Le logo de Gnuplot

3.2 Solutions technologiques

3.2.1 Polars

Quand il s'agit de manipuler et d'analyser des données significatives en Python, la bibliothèque la plus populaire est Pandas. Pour autant, populaire ne signifie pas forcément optimale. Il existe bien d'autres bibliothèques faisant office d'alternatives à Pandas. De plus, comme Pandas nous a été déconseillé, nous sommes rapidement tombés sur une autre bibliothèque, construite au-dessus de Pandas, mais permettant la parallélisation des calculs : **Dask** ([2]). Le problème de Dask étant que comme mentionné, elle est construite au-dessus de Pandas. Ce qui signifie que peu importe si on peut réaliser plusieurs calculs en parallèle, nous aurons toujours la lenteur des algorithmes de Pandas.

Nous nous sommes mis en quête de chercher une nouvelle bibliothèque, plus performante que Pandas, mais avec les avantages de Dask. Une en particulière a retenu notre attention : **Polars** ([9]) (voir figure 8). Destiné à la base pour le langage de programmation **Rust**, Polars a entamé une transition pour Python. Et le moins qu'on puisse dire, c'est que Polars a beau être une bibliothèque récente et légère, le benchmark issu du site internet db-benchmark ([3]) est sans appel : Polars est un choix bien plus pertinent que la majorité de ses concurrents (voir figure 7)

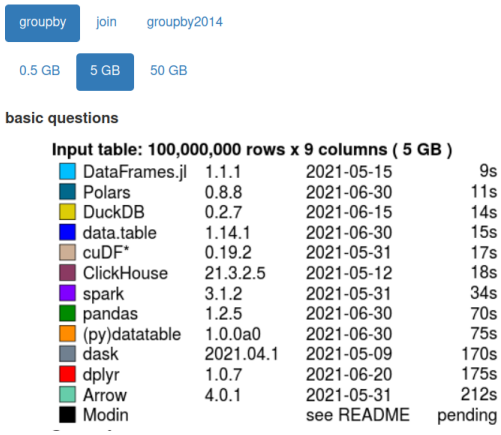

FIGURE 7 : Benchmark de plusieurs bibliothèques pour un traitement groupby sur un fichier CSV de 5 Go

Comment expliquer cette rapidité ? Et bien tout d'abord, le créateur de Polars, sous le pseudonyme « ritchie46 » est entre-autre, un ancien fondateur de Pandas. Il sait ainsi les erreurs à ne pas reproduire sur sa bibliothèque. De plus, Polars a été conçu pour réaliser des calculs en parallèles contrairement à Dask qui se contente de paralléliser ceux de Pandas. Cela est possible grâce à une notion appelée le lazyframe. Fini de se prendre la tête pour savoir si on réalise les calculs parallèles de façon classique (un thread pour le traitement d'une ligne) ou par chunk (un thread pour le traitement d'un bloc de lignes), Polars choisira automatiquement le plus adapté. Par ailleurs, Polars utilise la notion d'expression (comme groupby sur l'illustration précédente). C'est le fait de transformer un calcul donné. Par exemple, si on souhaite filtrer uniquement les MMSI dont l'identifiant est 0, on écrit avec Polars :

```
polars.col("MMSI") == 0
```

Polars va en premier lieu convertir cette condition en expression puis l'optimiser. L'expression obtenue est semblable à une requête issue du langage **SQL** (Structured Query Langage) :

```
SELECT COLUMN "MMSI" WHERE "MMSI" = 0
```

FIGURE 8 : Le logo de Polars pour Python

3.2.2 Argparse

Un programme nécessite forcément une interaction avec l'utilisateur. Cette interaction peut être soit en CLI, soit en GUI. Le programme ayant vocation à être utilisé sur un serveur, le GUI n'est pas une priorité. Nous optons donc pour Argparse, une bibliothèque incluse dans Python, permettant la réalisation d'interfaces simples en ligne de commandes.

3.2.3 Les bibliothèques secondaires

En plus de Polars et Argparse, nous utilisons des bibliothèques secondaires permettant la simplification de la conception du programme. Une page entière sur ces dernières n'est pas pertinente, voici donc une liste des autres bibliothèques qui nous accompagnerons :

- Glob : pour la gestion des expressions rationnelles ;
- Inspect : permet de lister les composantes d'un fichier Python comme les fonctions et leurs paramètres, le nom de la classe, etc ;
- Os : pour pouvoir changer des paramètres liés à Python comme le répertoire où l'on travaille (pour pouvoir lancer certains fichiers comme le benchmark) ;
- Pathlib : remplaçant de la sous-bibliothèque os.path, permettant la manipulation des chemins d'un système de fichiers afin de choisir uniquement des fichiers avec certaines extensions, savoir si nous sommes dans un répertoire, etc.

3.3 Solutions concrètes

3.3.1 Décomposition du programme

Pour permettre de mener à bien ce projet, de nombreuses questions se sont posées. Tout d'abord, comment organiser le programme ? Python étant un langage multi-paradigme, nous avions d'abord penser à tout réaliser en orienté objet. Or, ce choix s'est vite élargi : par exemple pour la réalisation de l'interface en ligne de commande, le mieux était d'utiliser une orientation classique, c'est-à-dire avec un script. Ainsi, uniquement la classe LazyFrame a été organisée en orienté objet, permettant la manipulation plus simple pour des fichiers. Le reste, sont des simples scripts ou des descriptifs.

3.3.2 La classe LazyFrame et le fichier cli

La classe LazyFrame est une classe permettant l'enregistrement des fichiers à traiter ainsi que leur nom. Cette classe permet entre autre d'enregistrer des fichiers CSV donnés, mais encore des dossiers ou des expressions rationnelles. Cette classe vérifie également que les noms de colonnes soient corrects et refuse le traitement d'un fichier (en le rejetant) si elles ne sont pas bonnes. Cela évite les éventuelles erreurs liées à Polars si une colonne n'existe pas. À noter que la fonction save, est la fonction qui permet le traitement par Polars. Comme nous sommes en lazy, le traitement s'effectue uniquement lors de la sauvegarde des fichiers concernés, pour optimiser et paralléliser le traitement.

Quant au fichier cli, c'est lui qui gère l'interface en ligne de commandes en utilisant Argparse. Chaque chaîne de caractère entré est stocké dans une liste et la classe LazyFrame s'occupe du reste. Ensuite, pour reconnaître si la fonction souhaitée utilise des paramètres, il suffit de vérifier que la fonction contient des parenthèses ou non. Les paramètres seront ajoutées après l'ajout de la chaîne de caractère target, qui est la liste des fichiers, dossiers ou expressions rationnelles à traiter.

3.3.3 Les énumérations et le fichier utils

Pour que le programme soit plus facile à maintenir, des énumérations sont à disposition. Ces énumérations, permettent de modifier une valeur et que ce changement soit répercuté partout dans le programme. Les énumérations concernent :

- Args, une énumération permettant de modifier les paramètres de certaines fonctions de Polars (comme le dossier par défaut où sont sauvegardés les CSV, le séparateur utilisé pour les CSV, etc.) ;
- Column, une énumération avec toutes les colonnes qu'un fichier CSV correct est censé posséder ;
- Extension, une énumération de toutes les extensions de fichiers acceptées ;
- NavigationalStatus, une énumération des messages de statut de navigation.

Pour récupérer la valeur d'une énumération, on importe l'énumération concernée et on appelle « value » dessus. Par exemple, pour avoir la valeur du statut de navigation amarré (moored) :

```
from enums.NavigationalStatus import NavigationalStatus

NavigationalStatus.MOORED.value
```

Le fichier utils quant à lui, est un simple fichier permettant de vérifier certaines choses concernant le programme. Par exemple, le fichier cli doit afficher uniquement une liste des fonctions issues du fichier functions et non les fonctions des modules annexes utilisées. Le fichier utils possède également la fonction pour retourner cette liste.

3.3.4 Le fichier functions

Le fichier functions est l'un des fichiers principaux du programme. Conçu pour permettre plus facilement d'ajouter, supprimer ou de modifier des fonctions, le fichier est écrit de façon descriptif. Par ailleurs, chaque fonction issue du fichier functions, se retrouve automatiquement dans la liste des fonctions que la page aide de l'interface en ligne de commande affiche. En particulier, on y retrouve la fonction route, dont le procédé est comme suit :

- On trie d'abord par MMSI ;
- On compare le statut de navigation actuel avec le statut de navigation suivant, pour permettre d'éliminer les redondances en fonction du MMSI (comme le fait qu'un navire envoie deux fois le même statut de navigation pour dire qu'il est amarré) ;
- On garde uniquement les navires dont le statut de navigation est amarré ou en mouvement (qui signifie ici que l'on sort d'un port) ;
- On retrie en fonction de la date.

Pour utiliser le plein potentiel de ce fichier, le mieux est de décomposer ses fonctions en sous-fonctions, afin qu'elles soient utilisables pour un utilisateur et pas juste restreint à la fonction principale concernée. Attention ! Une nouvelle fonction doit forcément avoir comme premier paramètre le nom « lazyFrame » (sensible à la casse). Par exemple :

```
def x(lazyFrame, param1, param2, ...)
```

3.3.5 Le fichier benchmark

Le fichier benchmark permet comme son nom l'indique, la réalisation d'un benchmark pour une fonction, afin de déterminer le temps d'exécution moyen. Le benchmark est réalisé sur un grand ensemble de données pour avoir des informations plus pertinentes. En particulier, le fichier benchmark a été conçu pour le temps moyen d'exécution de la fonction route (voir figure 9).

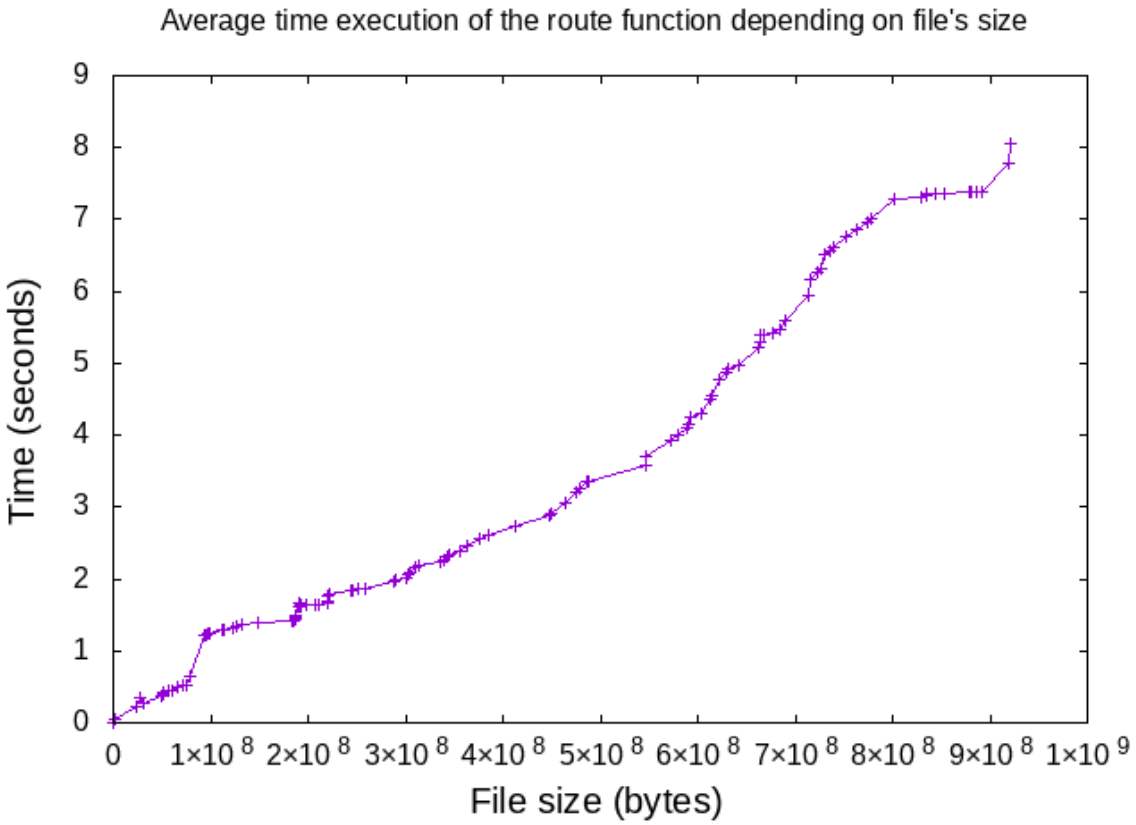

FIGURE 9 : Benchmark de la fonction route

Un constat évident : plus le fichier est lourd, plus l'exécution de la fonction prendra du temps. Un autre paramètre rentre aussi en jeu, il s'agit du nombre de lignes similaires. Effectivement, il est plus simple pour Polars de supprimer des lignes similaires que de comparer une à une les lignes.

4 Résultat

Il est important de noter que le programme requiert au minimum la version 3.7 de Python. Le programme a été également testé pour la version 0.13.39 de Polars.

On utilise le programme via un terminal de commandes. Pour l'exécuter, il suffit de se placer dans le dossier courant du projet et de faire :

```
mkel@HOME:~ python3 cli.py
----- Command-line interface for AIS-Operations -----

** Loading functions list
** Done

usage: cli.py [-h] --target TARGET [TARGET ...] --function FUNCTION
cli.py: error: the following arguments are required: --target, --function
```

Un message d'erreur s'affiche alors. Rien de grave : le programme a besoin de fichiers et d'une fonction pour fonctionner. En rajoutant « -h » après la commande, on obtient une page d'aide du programme :

```
mkel@HOME:~ python3 cli.py -h
----- Command-line interface for AIS-Operations -----

** Loading functions list
** Done

usage: cli.py [-h] --target TARGET [TARGET ...] --function FUNCTION

optional arguments:
  -h, --help          show this help message and exit
  --target TARGET [TARGET ...]
                      string(s) of path(s), file(s) or regexp(s) where to
                      execute the function
  --function FUNCTION sort(column), route, filter(condition)
```

Dans la partie aide du paramètre function, on peut y voir la liste des fonctions disponibles ainsi que les paramètres qui doivent être fournis à la fonction. Ainsi, si on veut par exemple exécuter la fonction sort(column), il suffit d'exécuter la commande suivante :

```
python3 cli.py --target "cible.csv" --function "sort('MMSI')"
```

Pour exécuter la fonction sur plusieurs cibles, il suffit d'ajouter les chemins des cibles les uns à la suite des autres :

```
python3 cli.py --target "cible.csv" "cible2.csv" "/dossier/" --function
    "sort('MMSI')"
```

Il est également possible de donner des expressions rationnelles :

```
python3 cli.py --target "cible.csv" "cible2.csv" "/dossier/*.csv"
    --function "sort('MMSI')"
```

Tous les résultats sont par défaut enregistrés dans le dossier resources/csv.

Conclusion

Le programme ayant été réfléchi pour pouvoir être utilisé et compléter dans l'avenir, il ne faut pas s'arrêter uniquement sur trois fonctions. Maintenant que la partie la plus difficile a été réalisée (la base du programme), il va falloir proposer des foncti

Cependant, l'utilisation d'une bibliothèque récente (bien que plus performante que ses concurrents) nous a quand même donné quelques problèmes. Effectivement, même si aujourd'hui le site internet de Polars est plus complet que lors du début de ce TPE, ce n'était pas le cas il y a quelques mois. Ainsi lâché dans la nature sans exemples concrets, nous avions dû faire face à un ennemi redouté des développeurs, mais aussi son meilleur ami : la documentation. Comprendre comment Polars fonctionnait globalement pour pouvoir ensuite en tirer profit.

Un autre point qui peut être intéressant, est l'extension du programme pour une autre instance d'AIS qui est ADS-B. ADS-B est l'équivalent d'AIS mais pour le trafic aérien. Il s'agit d'un site développé par les mêmes personnes qu'AISHub et est de ce fait, gratuit, comparé à son populaire concurrent (mais payant), Flightradar24.

Lexique

• Bibliothèque : collection de fonctions pouvant être utilisées directement dans un programme.

• C : langage de programmation.

• C# : langage de programmation.

• CLI (Command-Line Interface) : interface en ligne de commande permettant d'interagir avec un programme.

• CSV (Comma-Separated Values) : type de fichier qu'il est possible de créer ou de modifier dans Excel.

• Dask : bibliothèque Python open source flexible pour le calcul parallèle.

• Débogueur : logiciel qui aide un développeur à analyser les bugs d'un programme.

• DevOps : mouvement en ingénierie informatique et pratique technique visant à l'unification du développement logiciel et de l'administration des infrastructures informatiques.

• Electron : environnement permettant de développer des applications avec des technologies web.

• Github : service web d'hébergement et de gestion de développement de logiciels détenu par Microsoft.

• GUI (Graphical User Interface) : interface graphique permettant d'interagir avec un programme.

• IDE (Integrated Development Environment) : ensemble d'outils qui permet d'augmenter la productivité.

• IntelliSense : aide à la saisie semi-automatique de code.

- Java : langage de programmation.

- Licence GPL (General Public Licence) : licence permettant le droit d'utiliser le programme à n'importe quelle fin, le droit de modifier le programme et accéder à son code source, le droit de copier et distribuer le programme, le droit d'améliorer le programme et publier vos propres versions.

- Licence libre : licence permettant à toute personne recevant le logiciel de l'utiliser, le copier, le modifier, le fusionner, le publier, le distribuer et de le vendre.

- Licence MIT (Massachusetts Institute of Technology) : licence permettant à toute personne recevant le logiciel le droit illimité de l'utiliser, le copier, le modifier, le fusionner, le publier, le distribuer, le vendre et de changer sa licence. La seule obligation est de mettre le nom des auteurs avec la notice de copyright.

- Licence PSFL (Python Software Foundation License) : licence unique de Python similaire à la licence GPL.

- Logiciel libre : un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification et la duplication par autrui en vue de sa diffusion sont permises.

- Maven : outil de gestion et d'automatisation de production des projets logiciels Java.

- Open-source : un logiciel open-source est un logiciel dont le code peut être accessible par tout le monde.

- Pandas : bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données.

- Parallélisation : permet de traiter des informations de façon simultanée.

- Perl : langage de programmation pour traiter des informations textuelles.

- Python : langage de programmation.

- RADAR (RAdio Detection And Ranging) : système utilisant les ondes électromagnétiques pour détecter la présence et déterminer la position ainsi que la vitesse

d'un objet.

- Ruby : langage de programmation.
- Rust : langage de programmation développé par Mozilla.
- Science des données : extraction de connaissance d'ensembles de données.
- Snippet : petite portion réutilisable de code source ou de texte.

Références

[1] AISHub. « Free AIS vessel tracking | AIS data exchange | JSON/XML ship positions ». *Free AIS vessel tracking | AIS data exchange | JSON/XML ship positions*. Création du site le 9 février 2009, mise à jour le 16 mai 2022. [En ligne]. Disponible sur : `https://www.aishub.net/`. (Consulté le 19 novembre 2021).

[2] M. Rocklin. « Dask : Scalable analytics in Python ». *Dask : Scalable analytics in Python*. Création du site le 7 novembre 2009, mise à jour le 25 novembre 2019. [En ligne]. Disponible sur : `https://dask.org/`. (Consulté le 19 novembre 2021).

[3] H2O.ai. « Database-like ops benchmark ». *Database-like ops benchmark*. Création du site le 21 mai 2019, mise à jour le 16 mai 2022. [En ligne]. Disponible sur : `https://h2oai.github.io/db-benchmark/`. (Consulté le 25 novembre 2021).

[4] Software Freedom Conservancy. « Git ». *Git*. Création du site le 23 juillet 2008, mise à jour le 24 mai 2022. [En ligne]. Disponible sur : `https://git-scm.com/`. (Consulté le 19 novembre 2021).

[5] F. Beuke. « Github Language Stats ». *Github Language Stats*. Création du site le 6 septembre 2017, mise à jour le 24 mai 2022. [En ligne]. Disponible sur : `https://madnight.github.io/githut/#/pull_requests/2021/4`. (Consulté le 26 mai 2022).

[6] Gitlab. « Iterate faster, innovate together | in GitLab ». *Iterate faster, innovate together | GitLab*. Création du site le 15 janvier 2004, mise à jour le 16 août 2021. [En ligne]. Disponible sur : `https://about.gitlab.com/`. (Consulté le 19 novembre 2021).

[7] T. Williams et C. Kelley. « gnuplot homepage ». *gnuplot homepage*. Création du site le 3 janvier 2002, mise à jour le 7 janvier 2022. [En ligne]. Disponible sur : `http://www.gnuplot.info/`. (Consulté le 4 mars 2022).

[8] W. McKinney. « pandas - Python Data Analysis Library ». *pandas - Python Data Analysis Library*. Création du site le 16 décembre 2011, mise à jour le 25 novembre 2019. [En ligne]. Disponible sur : `https://pandas.pydata.org/`. (Consulté le 19 novembre 2021).

[9] R. Vink. « Polars ». *Polars*. Création du site le 15 septembre 2021, mise à jour le 15 septembre 2021. [En ligne]. Disponible sur : `https://www.pola.rs/`. (Consulté le 20 novembre 2021).

[10] Python Software Foundation . « Welcome to Python.org ». *Welcome to Python.org*. Création du site le 26 mars 1995, mise à jour le 24 février 2022. [En ligne]. Disponible sur : `https://www.python.org/`. (Consulté le 19 novembre 2021).

[11] Python Software Foundation. « 3.10.4 Documentation ». *3.10.4 Documentation*. Création du site le 26 mars 1995, mise à jour le 24 février 2022. [En ligne]. Disponible sur : `https://docs.python.org/3/`. (Consulté le 19 novembre 2021).

[12] U SHIP. « Uship : Accastillage et accessoires bateau de plaisance et voilier - Équipement nautique ». *Comment fonctionne la technologie AIS ? | Uship*. Création du site le 9 mars 1999, mise à jour le 17 juillet 2021. [En ligne]. Disponible sur : `https://www.uship.fr/default/blog/post/explication-technologie-ais`. (Consulté le 21 mai 2022).

[13] Stack Exchange. « Stack Overflow Developer Survey 2019 ». *Stack Overflow Developer Survey 2019*. Création du site le 26 décembre 2003, mise à jour le 19 mai 2021. [En ligne]. Disponible sur : `https://insights.stackoverflow.com/survey/2021`. (Consulté le 15 mai 2022).

[14] Microsoft. « Visual Studio Code - Code Editing. Redefined ». *Visual Studio Code - Code Editing. Redefined*. Création du site le 17 mars 1997, mise à jour le 15 février 2022. [En ligne]. Disponible sur : `https://code.visualstudio.com/`. (Consulté le 19 novembre 2021).