Institut Universitaire de Technologie,
Aix-Marseille Université

RAPPORT DE STAGE
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Deux aspects de gestion de données interopérables à l'OSU Pythéas

Emilien Bonneton

Institut Méditerranéen d'Océanologie

Responsable entreprise : Maurice Libes

Responsable académique : Delphine Rousseau

2022

Table des matières

1 Introduction

En vue de la validation de mon diplôme au DUT R&T, j'ai effectué un stage de 10 semaines au sein du service d'observation de l'Observatoire des Sciences de l'Univers (OSU) Pythéas, et du Laboratoire Institut Méditerranéen Océanographe (MIO en anglais).

L'Institut Méditerranéen d'Océanologie a été fondé le 1er janvier 2012. C'est un laboratoire public de recherche en Océanologie des Universités d'Aix-Marseille, de Toulon, du CNRS (Centre National de la Recherche Scientifique) et de l'IRD (Institut de recherche pour le développement).

Le MIO est un organisme de recherche qui dans ses projets recueille de nombreuses données sur le milieu marin. Dans le cadre de la Science ouverte ces données doivent être Faciles à trouver, Accessibles, Interopérables et Réutilisables. On parle de données « FAIR ».

Dans le cadre de la gestion FAIR de ces données, l'Institut Océanographique Méditerranéen a décidé de rendre ses données accessibles via différentes plateformes.
Pour que les données soient FAIR il faut qu'elles soient bien documentées et décrites à l'aide de métadonnées, et dans des formats standards ouverts communément utilisés dans une discipline scientifique.

Mon stage s'est déroulé dans le service d'observation de l'OSU Pythéas et a eu pour objectif de travailler dans le cadre de la science ouverte sur 2 projets :

1. La mise à disposition de données météo provenant de stations météo qui appartiennent à l'OSU Pythéas
2. Le reformatage automatique de fichiers CSV (Comma-separated values) en format interopérable NetCDF (Network Common Data Format)

1.1 Le service d'observation de l'OSU

Le Service d'Observation (S.O.) de l'OSU a une mission de service support à la recherche transversale de l'Institut et une réponse à la demande sociétale. L'observation à moyen et à long terme de l'évolution de l'environnement est clairement reconnue comme une nécessité impérative si l'on veut comprendre comment les écosystèmes terrestres ou marins réagissent à la fois aux contraintes naturelles de l'environnement et aux effets anthropiques.

Un Service d'Observation rassemble des méthodes et des opérateurs autour d'un ensemble de paramètres observés dont le maintien sur le très long terme (plus de 10 ans) se justifie scientifiquement par la dynamique des systèmes observés et la nature des questions scientifiques qui justifient les observations. Un SO doit donc à minima correspondre à :

- Une collection d'observables sur un/des objet(s) complexe(s) avec un cahier des charges sur le suivi des mesures (répétabilité, fidélité,…), leur pérennisation (fiabilité, robustesse …) et leur qualité (sensibilité, précision …)
- La mise à disposition des observations et des métadonnées reliées en direction de la communauté scientifique nationale et internationale dans les formes définies par le SO en suivant les standards internationaux le cas échéant.

- La mise à disposition des observations sans aucune restriction à toute entité souhaitant y avoir accès. L'entité est seule responsable des conséquences qu'elle pourrait faire de l'usage de ces observations. Les formes de diffusion seront les mêmes que celles pour la communauté scientifique sauf accord réciproque identifiée au préalable.

1.2 Ressources

Pour mener à bien ces 2 projets, j'ai eu à ma disposition un poste de travail sur Linux Mint ainsi que l'accès aux différentes machines et serveurs du Laboratoire via des connections SSH. Pour la conception des programmes j'ai principalement travaillé sur mon poste de travail en local. Néanmoins les quelques machines auxquelles j'ai eu accès ont été :

- meteo-osu : serveur hébergeant les traitements et l'affichage des données météo : http://meteo-osu.osupytheas.fr
- erddap : serveur généraliste hébergeant l'accès aux données : http://erddap.osupytheas.fr
- istsos : serveur hébergeant les services de gestion des séries temporelles et l'accès aux données : http://istsos.osupytheas.fr

2 Mise à disposition de données météo de l'OSU Pythéas

2.1 Contexte

L'OSU possède un ensemble de stations météos réparties sur le pourtour de la région marseillaise. Elles se situent sur l'île du Frioul, la chaîne de l'Etoile, Toulon, l'observatoire St Michel et enfin la station marine d'Endoume (Marseille littoral). Elles génèrent des séries temporelles relevant divers paramètres météorologiques tels que la température, la pression, l'humidité, la pression, la vitesse du vent, sa direction, les radiations solaires ou encore les précipitations.

Dans le cadre de la gestion FAIR et de la mise à disposition des données météo, l'équipe de mon tuteur de stage (Maurice Libes) a choisi de diffuser les données sur différents logiciels afin de tester leur efficacité, leur rapidité et leurs fonctionnalités.

Les logiciels-plateformes Cooperate, ERDDAP étaient déjà utilisés avant mon arrivée et IstSOS a été sélectionné pour tester son efficacité lors de la visualisation de ces données en séries temporelles avec un protocole particulier appelé SOS (pour Sensor Observation Service)

Le logiciel Cooperate est le service historique et non-interopérable créé par un collaborateur du service d'observation de l'OSU Pythéas. Le logiciel Erddap (Environmental Research Division Data Access Protocol) est un logiciel conçu pour la diffusion de données dans un cadre FAIR. Le projet IstSOS (Istituto Scienze della Terra Sensor Observation Service), un logiciel open-source conçu pour l'exploitation de données en séries temporelles interopérables et respectant le standard SOS (Sensor Observation Service), a constitué la première partie de mon stage.

2.2 Objectifs

En vue du contexte et du projet global de la gestion FAIR des données, les objectifs de mon travail ont été de :

1. Récupérer les fichiers bruts venant directement des stations météos de l'OSU.
2. Les convertir avec des programmes Python dans des formats spécifiques en fonction de leur provenance.
3. Les déposer dans les répertoires adaptés et cela avec une fréquence d'un cycle par heure.

Les stations concernées par ce projet sont celles de l'Etoile, d'Endoume et enfin Toulon. (Cf. Figure 1)

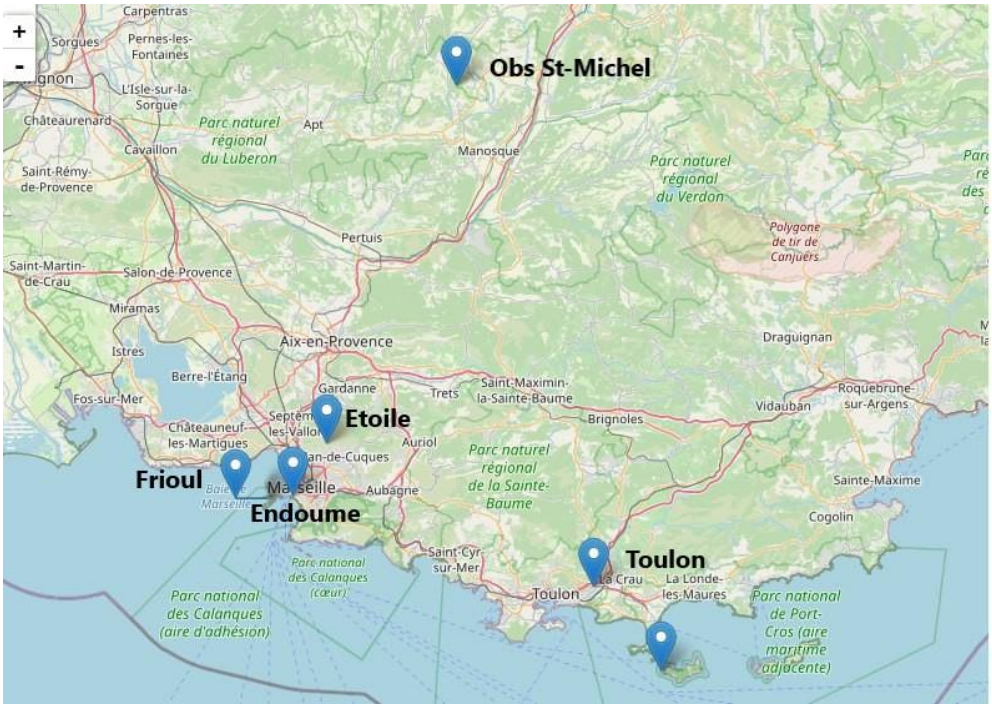


Figure 1 : *Carte des stations de l'OSU*

2.3 Les formats sources

Les 3 stations météos génèrent des fichiers hétérogènes dans le sens où les stations ne mesurent pas toutes les mêmes paramètres, ni à la même heure et ni à la même fréquence.

```
root@meteo-osu:~# more /mnt/meteo-data/la-garde/00006_010_1_COMPLEMENTAIRE_06_2022.TXT
PRIMARY_KEY DATE_RECORD DATE_OBS  INDICE_OBS  TAIR_M  UAIR_M  UAIRTD_M  RR_6M RG_6M DI_6M RDIFF_6M  VDMO
Y10_M VFMOY10_M VDIMX_M VFIMX_M PSTA_M  PMER_M  PREF  HPREF_M PQNH_M  PQFE_M
339389  01/06/2022 22:57:40 01/06/2022 00:00:00 0 16.8000 80.0000 13.3000 0.0000  0.0000    0.0000  190.
0000  0.5000      1007.900  1016.300
339390  01/06/2022 22:57:40 01/06/2022 00:06:00 0 16.6000 81.0000 13.3000 0.0000  0.0000    0.0000  0.00
00  0.0000      1007.800  1016.200
339391  01/06/2022 22:57:40 01/06/2022 00:12:00 0 16.6000 81.0000 13.3000 0.0000  0.0000    0.0000  0.00
00  0.0000      1007.900  1016.300
339392  01/06/2022 22:57:40 01/06/2022 00:18:00 0 16.6000 81.0000 13.3000 0.0000  0.0000    0.0000  0.00
00  0.0000      1007.900  1016.300
```

Exemple de fichier source météo pour Toulon

```
# date, time, temperature[C], humidity[%], dew point[C], sealevel pressure[hPa], avg wind speed[m
/s], gust speed[m/s], winddir, rainfall00
2022-06-16,00:00,22.2,87,19.8,1019.6,0.2,1.4,29,0.0
2022-06-16,01:00,22.0,85,19.3,1019.1,0.0,0.9,107,0.0
2022-06-16,02:00,21.9,82,18.6,1018.9,0.3,2.4,80,0.0
2022-06-16,03:00,21.5,80,17.8,1018.8,0.5,1.9,258,0.0
2022-06-16,04:00,21.3,79,17.5,1018.9,0.1,0.9,77,0.0
2022-06-16,05:00,20.8,76,16.4,1019.2,0.1,0.9,349,0.0
2022-06-16,06:00,20.9,74,16.1,1019.4,0.1,0.9,110,0.0
2022-06-16,07:00,22.1,71,16.4,1019.6,0.3,0.9,15,0.0
2022-06-16,08:00,22.8,67,16.3,1019.9,0.8,1.9,48,0.0
2022-06-16,09:00,23.5,65,16.4,1020.1,0.7,1.4,281,0.0
2022-06-16,10:00,23.4,73,18.1,1020.0,1.5,3.8,264,0.0
```

Exemple de fichier source météo pour Endoume

```
"TOA5","8395","CR3000","8395","CR3000.Std.32.05","CPU:CR3000_final_v4.CR3","4732","Table1"
"TIMESTAMP","RECORD","PTemp","RH","AirTC","pyr_en","WD_deg","WS_ms","PhCount_30m_borne7","PhCount
_30m_borne9","PhCount_20m","PhCount_5m","PhCount_3m","Permitivity_41_2_12","TH2O_41_2_12","Permit
ivity_63_2_13","TH2O_63_2_13","Permitivity_63_1_14","TH2O_63_1_14","Permitivity_30_2_15","TH2O_30
_2_15","Permitivity_EXT2_16","TH2O_EXT2_16","Permitivity_30_1_17","TH2O_30_1_17","Permitivity_34_
1_18","TH2O_34_1_18","Permitivity_40_1_21","TH2O_40_1_21","Permitivity_34_2_22","TH2O_34_2_22","P
ermitivity_41_1_23","TH2O_41_1_23","Permitivity_40_2_24","TH2O_40_2_24","Permitivity_39_1_25","TH
2O_39_1_25","Permitivity_39_2_26","TH2O_39_2_26","Permitivity_62_2_27","TH2O_62_2_27","Permitivit
y_62_1_28","TH2O_62_1_28"
"TS","RN","","","","","","","","","","","","","","","","","","","","","","","","","","","","","
","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","","
","","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","
Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","S
mp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp","Smp"
"2022-06-16 23:00:00",20192,26.67,39.09,27.75,772.2,-360,0.079,7999,-16.45,-16.45,-16.45,-16.45,N
AN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NA
N,NAN,NAN,NAN,NAN,NAN
"2022-06-16 23:15:00",20193,26.68,39.29,27.42,444.8,0,0.056,7999,-16.45,-16.45,-16.45,-16.45,NAN,
NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,NAN,N
```

Exemple de fichier source météo pour Endoume

En outre, certains fichiers ont des entêtes contenant le nom des colonnes et d'autres n'en ont pas. Les données et en particulier la date ne sont pas au bon format. Il a donc été nécessaire de convertir les dates au format ISO qui est un format universel de format de date.

2.4 Les formats CSV cibles

Le but de mon projet a été de convertir en Python les formats météo sources dans des formats CSV cibles pour les 3 logiciels qui vont les « ingérer » : Cooperate, ERDDAP et IstSOS.

2.4.1 Format Cooperate

Le format Cooperate doit posséder un entête qui possède des paramètres qui évolue en fonction de la date et de l'heure. Aussi, les données sont dans des fichiers horaires pour les stations marseillaises et journaliers pour la station de Toulon qui effectuent des enregistrements plus ou moins fréquemment en fonction de la configuration de la station.

Pour la station de l'Etoile, le programme que j'ai fait génère un fichier horaire contenant 4 enregistrements d'un pas de 15 minutes :

```
root@meteo-osu:~# more /data/climed/2022-06-10/SN000001/001_20220610T120000.txt
File name: /data/climed/2022-06-10/SN000001/001_20220610T120000.txt
Job:         JOB_001
Schedule:    B "JOB_001"
Serial num:  000001

Channel map:
0: none  name:"HU_INS_Com"  units:"%"
1: none  name:"T2m_INS_Com"  units:"deg C"
2: none  name:"PYR_INS_Com"  units:"W/m2"
3: none  name:"DDmoy_INS_Com"  units:"degres"
4: none  name:"FFmoy_INS_Com"  units:"m/s"

DATA start:
2022/06/10 12:00:00.0000, 21.03, 28.81, 949, 360, 0.055
2022/06/10 12:15:00.0000, 20.86, 28.76, 612.7, -540.2, 0.052
2022/06/10 12:30:00.0000, 21.34, 28.8, 514.4, -359.5, 0.054
2022/06/10 12:45:00.0000, 22.01, 27.85, 128.5, 360, 0.035
```

Exemple de fichier type Cooperate (ici un fichier de la station de l'Etoile)

Pour la station Endoume, mon programme génère un fichier horaire contenant 1 enregistrement par heure.

Pour la station de Toulon, mon programme génère un fichier journalier contenant tous les enregistrements de la journée d'un pas de 6 minutes.

2.4.2 Format Erddap

Les fichiers destinés à Erddap sont des fichiers CSV journaliers avec un entête simple et explicite (toujours en fonction des paramètres mesurés par la station).

Pour la station de l'Etoile, on a un enregistrement tous les quarts d'heure dans un fichier journalier.

Pour Endoume, on a un enregistrement toutes les heures dans un fichier journalier :

```
root@meteo-osu:~# more /mnt/SO-DATA/meteo-endoume/erddap/meteo-endoume-erddap-2022-06-10.csv
Datetime, temperature[C], humidity[%], dew point[C], sealevel pressure[hPa], avg wind speed[m/s], gust speed[m/s],
 winddir[degree], rainfall00, Longitude, Latitude
2022-06-10T00:00:00, 20.2, 52.0, 10.1, 1016.4, 3.8, 10.4, 336.0, 0.0, 5.3474133, 43.2807698
2022-06-10T01:00:00, 19.7, 55.0, 10.3, 1016.9, 2.2, 8.5, 345.0, 0.0, 5.3474133, 43.2807698
2022-06-10T02:00:00, 19.2, 56.0, 10.3, 1017.2, 2.3, 8.1, 333.0, 0.0, 5.3474133, 43.2807698
2022-06-10T03:00:00, 18.8, 57.0, 10.2, 1017.4, 1.3, 5.7, 341.0, 0.0, 5.3474133, 43.2807698
2022-06-10T04:00:00, 18.4, 58.0, 9.9, 1017.5, 1.4, 6.6, 86.0, 0.0, 5.3474133, 43.2807698
2022-06-10T05:00:00, 18.1, 54.0, 8.6, 1017.6, 1.6, 6.6, 35.0, 0.0, 5.3474133, 43.2807698
2022-06-10T06:00:00, 17.9, 54.0, 8.6, 1018.0, 1.0, 5.2, 337.0, 0.0, 5.3474133, 43.2807698
2022-06-10T07:00:00, 18.6, 55.0, 9.3, 1018.1, 0.9, 3.3, 320.0, 0.0, 5.3474133, 43.2807698
2022-06-10T08:00:00, 19.7, 54.0, 10.1, 1018.4, 0.8, 3.3, 321.0, 0.0, 5.3474133, 43.2807698
2022-06-10T09:00:00, 20.7, 55.0, 11.3, 1018.5, 1.2, 4.7, 322.0, 0.0, 5.3474133, 43.2807698
2022-06-10T10:00:00, 21.8, 52.0, 11.6, 1018.4, 1.3, 5.2, 328.0, 0.0, 5.3474133, 43.2807698
2022-06-10T11:00:00, 23.0, 48.0, 11.5, 1018.7, 1.7, 4.7, 312.0, 0.0, 5.3474133, 43.2807698
2022-06-10T12:00:00, 23.4, 49.0, 12.2, 1018.7, 2.2, 5.2, 302.0, 0.0, 5.3474133, 43.2807698
2022-06-10T13:00:00, 24.5, 43.0, 11.3, 1018.8, 2.7, 4.7, 297.0, 0.0, 5.3474133, 43.2807698
2022-06-10T14:00:00, 25.5, 36.0, 9.1, 1018.9, 1.9, 3.3, 279.0, 0.0, 5.3474133, 43.2807698
2022-06-10T15:00:00, 25.1, 38.0, 9.7, 1018.8, 2.9, 5.2, 249.0, 0.0, 5.3474133, 43.2807698
2022-06-10T16:00:00, 23.8, 47.0, 11.5, 1018.8, 3.1, 5.2, 213.0, 0.0, 5.3474133, 43.2807698
2022-06-10T17:00:00, 24.4, 43.0, 11.0, 1018.6, 3.3, 5.2, 193.0, 0.0, 5.3474133, 43.2807698
2022-06-10T18:00:00, 23.8, 45.0, 11.0, 1018.5, 2.0, 4.3, 144.0, 0.0, 5.3474133, 43.2807698
2022-06-10T19:00:00, 24.1, 45.0, 11.4, 1018.4, 2.7, 5.7, 189.0, 0.0, 5.3474133, 43.2807698
2022-06-10T20:00:00, 22.4, 50.0, 11.3, 1018.7, 3.4, 6.2, 238.0, 0.0, 5.3474133, 43.2807698
2022-06-10T21:00:00, 21.8, 51.0, 11.3, 1019.0, 0.6, 2.8, 131.0, 0.0, 5.3474133, 43.2807698
2022-06-10T22:00:00, 22.1, 48.0, 10.5, 1019.5, 0.5, 2.8, 274.0, 0.0, 5.3474133, 43.2807698
```

Exemple de fichier type Erddap pour la station d'Endoume

Pour Toulon, on a un enregistrement toutes les 6 minutes dans un fichier journalier.

2.4.3 Format IstSOS

Les fichiers destinés à Erddap sont des fichiers CSV journaliers avec un entête spécifique pour le logiciel IstSOS (toujours en fonction des paramètres mesurés par la station) et au format '.dat'.

Pour la station de l'Etoile, on a un enregistrement tous les quarts d'heure dans un fichier journalier :

```
root@meteo-osu:~# more /mnt/SO-DATA/meteo-climed/istsos/meteorologie_climed_20220615234500.dat
urn:ogc:def:parameter:x-istsos:1.0:time:iso8601,urn:ogc:def:parameter:x-istsos:1.0:longitude,urn:ogc:def:parameter
:x-istsos:1.0:latitude,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:humidity,urn:ogc:def:parameter:x-istsos:1.0:me
teo:air:temperature,urn:ogc:def:parameter:x-istsos:1.0:meteo:solar:radiation,urn:ogc:def:parameter:x-istsos:1.0:ai
r:wind:direction,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:wind:speed
2022-06-15T00:00:00.000000+0000, 5.42, 43.37, 75.33, 19.66, 2050, -999.99, 0.015
2022-06-15T00:15:00.000000+0000, 5.42, 43.37, 71.99, 20.09, 2426, 360, 0.029
2022-06-15T00:30:00.000000+0000, 5.42, 43.37, 67.99, 21.24, 2860, -360, 0.014
2022-06-15T00:45:00.000000+0000, 5.42, 43.37, 66.65, 21.91, -999.99, -999.99, 0.023
2022-06-15T01:00:00.000000+0000, 5.42, 43.37, 67.56, 21.91, -999.99, 360, 0.022
2022-06-15T01:15:00.000000+0000, 5.42, 43.37, 68.54, 21.94, 3131, 720, 0.024
2022-06-15T01:30:00.000000+0000, 5.42, 43.37, 64.2, 23, 2346, -360, 0.014
2022-06-15T01:45:00.000000+0000, 5.42, 43.37, 66.99, 22.22, -999.99, 360, 0.017
2022-06-15T02:00:00.000000+0000, 5.42, 43.37, 64.57, 22.6, -999.99, -999.99, 0.023
2022-06-15T02:15:00.000000+0000, 5.42, 43.37, 60.84, 22.65, -999.99, -999.99, 0.036
2022-06-15T02:30:00.000000+0000, 5.42, 43.37, 53, 23.42, 3738, 360, 0.031
2022-06-15T02:45:00.000000+0000, 5.42, 43.37, 50.32, 22.46, 2130, -999.99, 0.023
2022-06-15T03:00:00.000000+0000, 5.42, 43.37, 56.34, 23.14, -999.99, 360, 0.022
2022-06-15T03:15:00.000000+0000, 5.42, 43.37, 55.76, 23.48, -999.99, 360, 0.031
2022-06-15T03:30:00.000000+0000, 5.42, 43.37, 53.62, 23.69, 3781, -999.99, 0.025
2022-06-15T03:45:00.000000+0000, 5.42, 43.37, 52.41, 23.99, 3568, 0, 0.022
2022-06-15T04:00:00.000000+0000, 5.42, 43.37, 53.41, 23.61, 3053, -999.99, 0.022
2022-06-15T04:15:00.000000+0000, 5.42, 43.37, 55.63, 23.08, 2277, -999.99, 0.034
2022-06-15T04:30:00.000000+0000, 5.42, 43.37, 53.04, 23.57, 3131, 360, 0.026
2022-06-15T04:45:00.000000+0000, 5.42, 43.37, 50.3, 23.85, 3039, 360, 0.032
2022-06-15T05:00:00.000000+0000, 5.42, 43.37, 46.97, 24.61, -999.99, 0, 0.025
2022-06-15T05:15:00.000000+0000, 5.42, 43.37, 50.44, 24.44, 3684, 360, 0.035
2022-06-15T05:30:00.000000+0000, 5.42, 43.37, 59.01, 23.08, 2384, 360, 0.025
2022-06-15T05:45:00.000000+0000, 5.42, 43.37, 72.07, 22.63, 2675, 0, 0.037
2022-06-15T06:00:00.000000+0000, 5.42, 43.37, 76.71, 22.45, 3514, -999.99, 0.032
2022-06-15T06:15:00.000000+0000, 5.42, 43.37, 80.1, 22.49, 3784, 360, 0.03
2022-06-15T06:30:00.000000+0000, 5.42, 43.37, 72.91, 23.25, -999.99, 240.2, 0.024
2022-06-15T06:45:00.000000+0000, 5.42, 43.37, 70.7, 23.62, -999.99, 0, 0.01
```

Exemple de fichier type IstSOS pour la station de l'Etoile

Pour Endoume, on a un enregistrement toutes les heures dans un fichier journalier.
Pour Toulon, on a un enregistrement toutes les 6 minutes dans un fichier journalier.

2.5 La structure des programmes de conversion de fichiers en Python

Dans le cadre de ce projet j'ai dû réaliser 3 programmes en Python, 1 programme par station météo, pour convertir les fichiers météo « sources » dans les formats demandés par les logiciels.

La raison pour laquelle il y a un programme par station est liée aux différents paramètres qui sont générés par les stations. En effet, les programmes ont été pensés de manière très spécifique et sont donc destinés à n'être utilisés que dans le cadre de la station pour laquelle ils ont été conçus. Ce choix a été effectué avant mon arrivé au sein du service.

Néanmoins on peut simplifier la structure du programme en 5 parties :

- *Paramétrage des répertoires d'entrée et de sortie des fichiers*
- *Création des fichiers de sortie pour Erddap et IstSOS*
- *Ecriture de la ligne d'entête pour Erddap et IstSOS*
- *Lecture des données dans les fichiers météo sources avec la librairie Pandas de Python*
- *Acquisition des données dans un « dataframe »* (DF) de Panda*
- *Transformation du DF en liste de liste*
- *Traitement des données ligne par ligne (mise au bon format pour la date par exemple, ou bien l'ajout des coordonnées de la station sur chaque ligne)*
- *Interrogation d'un fichier Base De Donnée* (BDD) pour vérifier si la ligne lue a déjà été traitée ou pas*
- *Création et écriture dans les fichiers de sortie*

Les programmes dans leur totalité sont disponibles en Annexes ainsi que les graphiques résultants du projet.

2.6 Le workflow des données

Les programmes de conversion s'inscrivent dans la première étape du workflow* de données.

Le but de ce workflow étant de rendre accessible ces données sur les différentes plateformes et ce de façon automatique, il a fallu mettre en place la prochaine étape qui est de faire « ingérer » ces données aux logiciels.

Pour ce qui est des logiciels Cooperate et Erddap, l'ingestion se fait par un process automatique inhérent au logiciel.

Pour le logiciel IstSOS l'ingestion de données se fait avec une commande (Cf. Commande IstSOS), lancée chaque heure lorsqu'il y a des nouvelles données. J'ai donc fait un script en Bash, qui lance la commande permettant à IstSOS de récupérer les nouveaux enregistrements et qui va ensuite classer les nouveaux fichiers dans des répertoires par mois et année.

python3 scripts/csv2istsos.py -user admin -password 'monpass' -s meteorologie -p meteorologie_endoume -u http://istsos.osupytheas.fr/istsos -o Meteo_OSU_Pytheas -w /mnt/SO-DATA/meteo-endoume/istsos/

On peut alors résumer le workflow de la station de l'Etoile (que l'on appelle aussi Climed) par ce schéma :

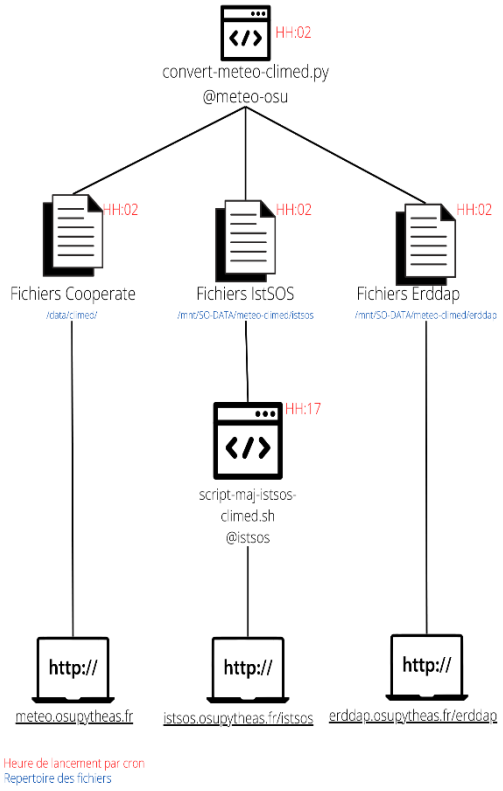


Schéma du workflow de la station de l'Etoile

2.7 Conclusions

2.7.1 La méthodologie, la conception de la structure du programme

La rigueur et l'organisation sont 2 aspects essentiels dans la bonne conception d'un programme. En effet, un programme est une suite logique d'opérations qui peut être plus ou moins complexe. Pour que la conception soit efficace le développeur doit faire preuve de méthode notamment lors de la conception ou lors des phases de tests.

Pour la conception des programmes, j'ai pu prendre connaissance du contexte grâce à mon tuteur de stage qui a su m'expliquer toutes les contraintes de production de fichiers.
Une fois toutes les contraintes établies, je les ai retranscrites à l'écrit de façon claire et ordonnée. J'ai aussi retranscrit la structure de mon premier programme via un schéma afin de compléter sa documentation.

Les premières semaines de mon stage je n'ai pas vraiment eu de méthodologie concernant l'intégration de fonctions.
En effet, je créais mes fonctions en une fois ce qui apportait forcément plus ou moins d'erreurs difficiles à résoudre car je ne pouvais déterminer quelle opération ne se passait pas comme je l'avais prévu.
Par la suite j'ai su faire les choses de manières plus ordonnées. En effet, la méthode la plus efficace est d'étudier ce que l'on a en entrée, bien déterminer clairement ce que l'on souhaite en sortie pour la prochaine étape et enfin faire des tests de quelques opérations afin d'étudier leur fonctionnement et trouver la bonne suite d'opérations à effectuer.

Les phases de tests se sont faites en local dans un premier temps, puis sur une machine dédiée aux phases de tests.

Les tests que j'ai effectués de mon côté n'étaient pas vraiment réfléchis les premières semaines. En effet, je faisais des tests après des modifications (légères ou importantes), cette méthode s'est révélée très peu efficace car les modifications effectuées créaient en réalité d'autres(s) problème(s). J'ai pu apprendre par la suite à mieux structurer mes tests, notamment par un débogage à l'aide de fonctions print() afin de mieux suivre le comportement de mon programme.

2.7.2 Les problèmes rencontrés

Tout au long de mon stage j'ai pu rencontrer différents types de problèmes.
En effet, j'ai pu avoir des erreurs générées par des éventualités que je n'ai pas pu anticiper et lesquelles j'ai dû traiter. C'est à dire corriger le problème ou bien indiquer par un message dans la console ou par mail les conditions qui n'ont pas été respectés et font échouer le programme.
Aussi, le traitement de certaines erreurs a impliqué de faire prendre une direction plus ou moins flexible au programme

Aussi, par défaut le programme traite le fichier météo de la date du jour.
Effectivement, le programme est lancé par le système « cron »* de Linux tous les jours à 23h55 donc il traite les données de la station reçu entre 00h00 et 23h55.
Or, les stations sont au format horaire UTC, cela veut dire que lorsqu'une mesure est effectuée à 23h elle ne sera pas notée '23h' mais '01h' du jour suivant.
C'est pourquoi dans ce cas-là on doit pouvoir traiter une date antérieure, en passant la date sur la ligne de commande avec l'option « -d ».

Exemple : ./convert-meteo-toulon.py -d 2022-06-07

De cette manière le système « cron » peut lancer la date de la veille pour récupérer les 2 derniers fichiers du jour précédent.

3 Reformatage de fichiers CSV en format interopérable NetCDF

Le format CSV est un format très répandu dans la gestion des DATA car c'est un format très souple et simple de lecture. Néanmoins, ce format de données est plus ou moins archaïque dans le sens où il n'impose aucune contrainte sur la description des données qu'il contient. Les libellés des colonnes peuvent être absents ou incompréhensibles, il peut manquer les unités des mesures etc.….

3.1 Les objectifs

C'est pourquoi, dans le cadre de la gestion FAIR des données pour la science ouverte, il est préférable de reformater les données dans le format NetCDF standard et interopérable qui peut fournir des informations supplémentaires notamment sur les prises de mesures et leurs conditions.

L'objectif de ce programme que j'ai réalisé est de convertir un fichier au format CSV au format NetCDF. L'objectif est de faire un programme le plus générique possible afin de pouvoir convertir n'importe quel type de données.

3.2 Le format NetCDF

Le format de fichier NetCDF permet de stocker des données scientifiques multidimensionnelles, c'est à dire des variables exprimées en fonction d'une ou plusieurs dimensions comme le temps, la profondeur ou l'altitude, les coordonnées Latitude/Longitude etc…
Ces variables peuvent être exprimées en fonction d'une ou plusieurs dimensions ce qui rend l'exploitation des données plus précise. Ce format a été créé par la NASA* et est très utilisé dans de nombreux domaines scientifiques telles que l'océanographie, l'atmosphère, la climatologie etc.

Les données au format NetCDF sont auto-documentées, c'est-à-dire qu'un fichier NetCDF contient de nombreuses métadonnées permettant de décrire les données qu'il contient.

NetCDF fournit un set de librairies disponibles dans de nombreux langages dont Python. En effet, la librairie sous Python permet de créer des fichiers NetCDF via des fonctions de créations et d'accès.

Un fichier NetCDF contient 4 grandes parties
- La partie qui exprime les dimensions des données (temps, profondeur, latitude/longitude, etc…) (Cf. Figure 2)
- La partie qui décrit toutes les variables du fichier (Cf . Figure 3)
- La partie des « global attributs » qui sont les métadonnées de tout le fichier
- Les données elles-mêmes (Cf. Figure 4)

BDD_HYDRO_H03_GIPREB_22122021...	SOOT-SEA : Impact of Black Carbon in South ...	Local File
Chloa	Chlorophylle a	1D
COP	Carbone Organique Particulaire	1D
Datetime	Datetime	1D
Latitude	Latitude	—
Longitude	Longitude	—
MES	Matieres en suspension	1D
NH4	Ammonium	1D
NO2	Nitrites	1D
NO3	Nitrates	1D
NOP	Nitrate Organique Particulaire	1D
Ntot	Nitrates Total	1D
Phaeo	Phaeo Pigments	1D
PO4	Phosphates	1D
Profondeur	profondeur	1D
Ptot	Phosphates Total	1D
Station	Station	—
station_name	station name	—

Exemple de fichier NetCDF (ouvert avec Panoply)*

```
netcdf file:/home/libes/Documents/Service-Observation/csv2
  dimensions:
    lenstation = 5;
    Datetime = 296;
    Profondeur = 2;
```

Figure 2 : *Partie qui exprime les dimensions*

```
float Longitude;
  :_FillValue = -999.99f; // float
  :units = "degrees_east";
  :long_name = "Longitude";
  :standard_name = "longitude";
  :coordsys = "geographic";
  :axis = "X";
  :coverage_content_type = "coordinate";
  :valid_range = -180.0f, 180.0f; // float
  :point_spacing = "even";

float Profondeur(Profondeur=2);
  :_FillValue = -999.99f; // float
  :units = "m";
  :long_name = "profondeur";
  :standard_name = "depth";
  :coordsys = "geographic";
  :axis = "Z";

float NO3(Datetime=296, Profondeur=2);
  :_FillValue = -999.99f; // float
  :units = "µmol.l-1";
  :long_name = "Nitrates";
  :standard_name = "nitrates";
  :coordsys = "geographic";
```

Figure 3 : *Partie qui decrit les variables*

```
data:

stationname = "EMSO_Ligure_Ouest_Albatross" ;

latitude = 42.793 ;

longitude = 6.038 ;

time = 1654473613, 1654475413, 1654477213, 1654479013, 1654480813,
    1654482613, 1654484413, 1654486213, 1654488013, 1654489813, 1654491613,
    1654493413, 1654495213, 1654497013, 1654498813, 1654500613, 1654502413,
    1654504213, 1654506013, 1654507813, 1654509613, 1654511413, 1654513213,
    1654515013, 1654516813, 1654520413, 1654522213, 1654524013, 1654525813,
    1654527613, 1654529413, 1654531213, 1654533013, 1654534813, 1654536613,
    1654538413, 1654540213, 1654542013, 1654543813, 1654545613, 1654547413,
    1654549213, 1654551013, 1654552813, 1654554613, 1654556413, 1654558213 ;

sensor_theoric_depth = 2000 ;
```

Figure 4 : *Partie qui montre les données (afficher avec ncdump)*

Les étapes de création de fichiers NetCDF sont donc :

- La création du fichier NetCDF
- La création des dimensions
- La création de la partie des « global attributs »
- La création des variables
- Ecriture des données

3.2.1 La création du fichier NetCDF

En Python, la création d'un fichier NetCDF se fait à l'aide de la fonction :

```
ds = Dataset(file_name, mode="w", format='NETCDF4')
```

Fonction permettant la création du fichier NetCDF

Que l'on assigne à une variable (ici « ds ») afin de pouvoir y ajouter les dimensions, les variables et les données par la suite.

3.2.2 La création des dimensions

La création d'une dimension NetCDF se fait à l'aide de la fonction :

```
ds.createDimension("x", 20)
ds.createDimension("y", 20)
ds.createDimension("time", None)
```

Fonctions permettant la création de dimensions pour un fichier NetCDF

Les dimensions sont caractérisées par un nom et une taille qui doit être un entier naturel. Ici on a les dimensions « x », « y » et « time ». La dimension « time » est dite « Unlimited » grâce à l'option « None » c'est-à-dire que la taille de la dimension sera calculée en fonction du nombre de valeur « time ». Seule la dimension principale peut avoir la taille « Unlimited ».

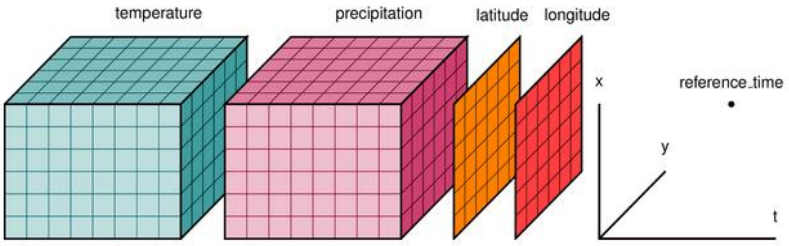
3.2.3 La création des variables

La création d'une variable NetCDF se fait à l'aide de la fonction :

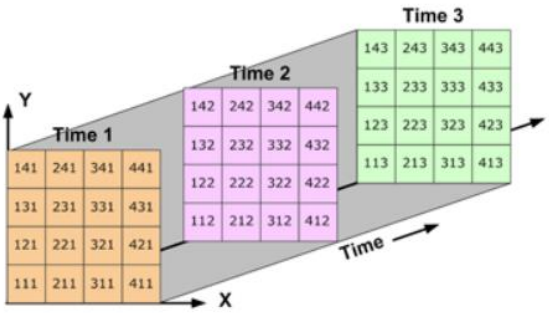
```
var1 = ds.createVariable("field1", "f4", ("time", "x", "y"))
var2 = ds.createVariable("field2", "f4", ("time", "x", "y"))
```

Fonctions permettant la création de variables pour un fichier NetCDF

Une variable est un ensemble de valeurs du même type. Les variables sont caractérisées par un nom, un type de donnée et une forme qui est déterminé par la liste de dimensions spécifiée lors de sa création. Ici, on a « var1 » qui dépend des dimensions « time », « x » et « y ». On a donc une variable en 3 dimensions (3D).



Representation visuelle de données 3D, 2D et 1D



Autre représentation visuelle de données 3D

Les variables de rang 0 sont dites des scalaires. Celles de rang 1 sont des vecteurs, celles de rang 2 des matrices et enfin celles de rang 3 sont des matrices « empilées » les unes sur les autres. Dans l'exemple, les variables « field1 » et « field2 » sont de rang 3, stockées respectivement dans « var1 » et « var2 » et sont au format « float4 » (c'est-à-dire un nombre relatif de 4 chiffres).

En outre, NetCDF permet de bien décrire les variables avec de nombreux attributs propres à chacune :

```
float lon(y=277, x=349);
  :units = "degrees_east";
  :long_name = "Longitude";
  :CoordinateAxisType = "Lon";

float lat(y=277, x=349);
  :units = "degrees_north";
  :long_name = "Latitude";
  :CoordinateAxisType = "Lat";

double time(time=31);
  :long_name = "Time";
  :delta_t = "";

float tasmax(time=31, y=277, x=349);
  :long_name = "Daily maximum temperature";
  :units = "Celcius";
  :missing_value = -999.0; // double
  :coordinates = "lon lat";
```

Exemple de description de variables

Les attributs servent à décrire les données, ce sont des métadonnées. On peut ajouter des métadonnées en écrivant dans « var1 ». Par exemple, si « var1 » était la Température on indiquerait les unités de cette manière :

```
var1.units = "Celcius"
var1.long_name = "Surface air temperature"
```

Opérations permettant l'ajout d'attributs pour une variable NetCDF

3.2.4 Ecriture des données

Enfin, l'écriture des données dans les variables doit se faire suivant le rang et la taille des dimensions de la variable. C'est-à-dire que si la variable est une constante, alors elle n'écrira qu'une unique valeur dans la variable.
Si elle est de rang 1 alors on écrira un vecteur (une matrice de 1 ligne ou bien une SeriesPandas*) de longueur la taille de la dimension en question.
Si elle est de rang 2 alors ce sera une matrice (un dataframe) de N lignes et M colonnes, avec N la taille de la première dimension et M la taille de la deuxième dimension.
Si elle est de rang 3 alors se sera une matrice 3D (un dataframe en 3D).

Une fois la forme défini, l'écriture de donnée dans une variable NetCDF se fait de cette manière :

```
var1[:] = data
```

Opérations permettant l'écriture des données dans la variable NetCDF

3.3 Les formats de fichiers sources en CSV

Pour pouvoir convertir les fichiers CSV en format NetCDF il va falloir nécessairement fournir au programme des indications qui n'existent pas dans le fichier CSV, notamment les attributs globaux, les dimensions, le nom standard des variables, les dimensions ou bien les variables.

Les fichiers sources doivent être au format CSV et doivent posséder un entête qui indique :
- les attributs standard de chaque variable, qui indiquent également
- si une variable est dimension ou non et enfin
- une ligne qui indiquent de quelle dimension doit dépendre la colonne en question.

Un entête qu'on rajoute au fichier CSV ressemble à celui-ci :

nom_colonnes	Station	Datetime	Latitude	Longitude	Profondeur	NO3	NO2	NH4	PO4
unites	string	time	degree_north	degree_east	m	µmol.l-1	µmol.l-1	µmol.l-1	µmol.l-1
standard_name	platform_name	datetime	latitude	longitude	depth	nitrates	nitrites	ammonium	phosphates
long_name	Station	Datetime	Latitude	Longitude	Profondeur	Nitrates	Nitrites	Ammonium	Phosphates
dimension		D1			D2				
variable	SD	D1	SD	SD	D2	D1,D2	D1,D2	D1,D2	D1,D2

Exemple d'entête précédant des données à convertir

La première colonne donne le nom des attributs standards qui sont :
- nom_colonnes : qui servira à donner le nom de la variable ou dimension
- unites : qui servira à remplir l'attribut « units »
- standard_name : permet de remplir l'attribut « standard_name »
- dimension : permet de déterminer quelle colonne sera une dimension
- variable : permet de déterminer de quoi dépendent les données de la colonne en question

3.4 La structure du programme de conversion CSV vers NetCDF

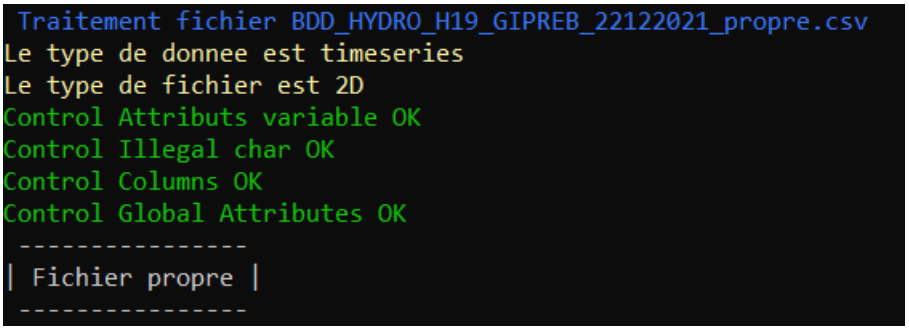
Dans le cadre de ce projet, j'ai dû ajouter la fonctionnalité « 2Dimensions » au programme déjà existant mais qui ne traitait seulement les fichiers en « 1Dimension ».

Les étapes de ce programme peuvent se décomposer ainsi :

- *Paramétrages des répertoires d'entrées et de sorties*
- *Ouverture des fichiers CSV d'entrée (il peut y en avoir plusieurs s'ils sont de la même forme)*
- *Ouverture du fichier « global_attributs » et récupération de ces informations*
- *Contrôle de l'homogénéité ainsi que de la propreté des fichiers d'entrée*
- *Création du fichier de sortie NetCDF*
- *Récupération des informations concernant les dimensions (1D ou 2D) et les variables*
- *Création des dimensions et des métadonnées*
- *Création des variables NetCDF et écriture des données*

Les étapes affichées sur la console permettent aussi de déterminer à quelle étape le programme se situe dans le traitement du fichier ainsi que des indications sur la structure des données. Notamment la taille des tableaux de données, leur noms, etc…

- *Contrôle de la propreté du fichier :*

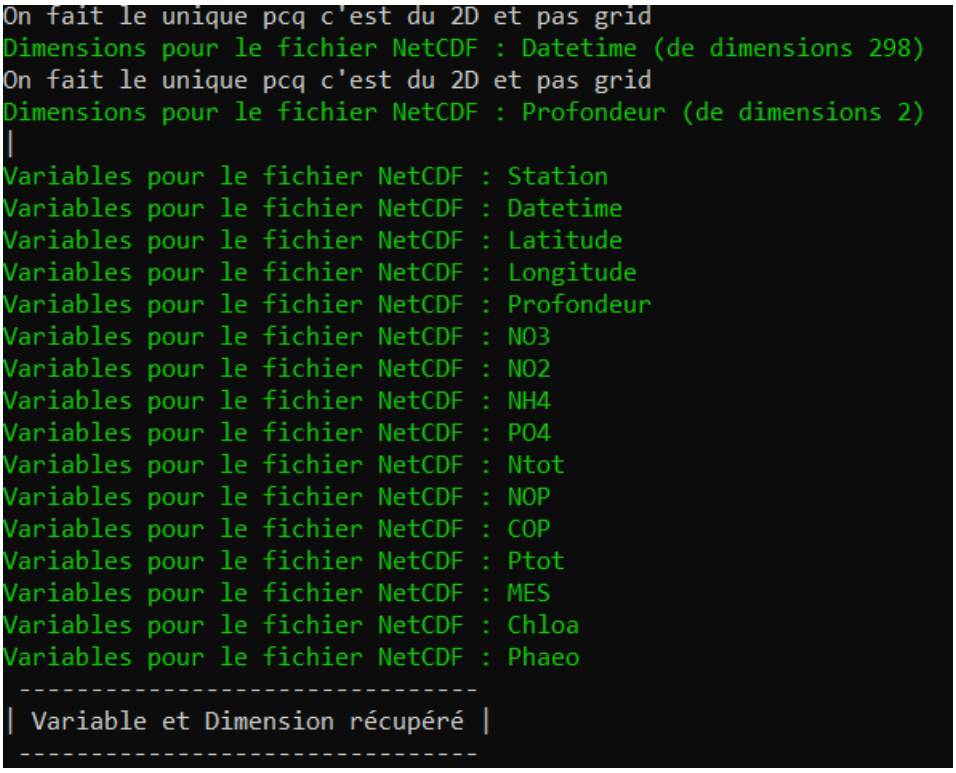


Capture n°1/5 des messages généré par le programme dans la console

- *Création du fichier de sortie :*

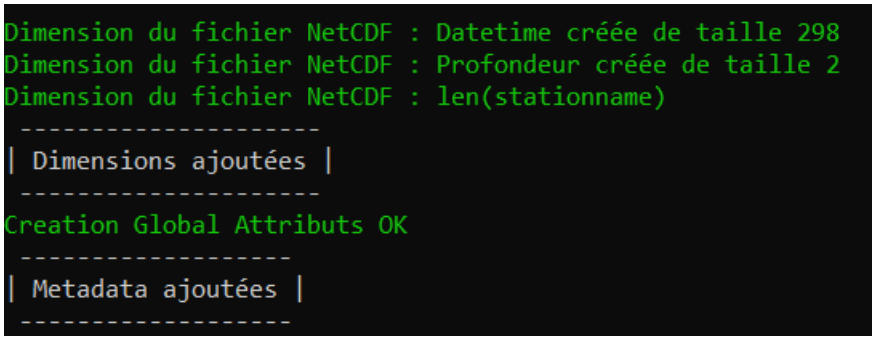
Capture n°2/5 des messages généré par le programme dans la console

- *Récupération des variables et dimensions dans le fichier d'entrée :*



Capture n°3/5 des messages généré par le programme dans la console

- *Ajout des méta données et des dimensions au fichier NetCDF :*



Capture n°4/5 des messages généré par le programme dans la console

- *Création des variables et écriture de leur données dans le fichier NetCDF :*

```
        Traitement col -Station-
Creation de la var Station de dimension SD -> OK
taille du tableau SD a ecrire:  1
Variable Station et ses datas ecrit avec succes
        Traitement col -Datetime-
Creation de la var Datetime de dimension Datetime -> OK
taille du tableau 1D (var dimension) de Datetime a ecrire:  298
Variable Datetime et ses datas ecrit avec succes
        Traitement col -Latitude-
Creation de la var Latitude de dimension SD -> OK
taille du tableau SD a ecrire:  1
Variable Latitude et ses datas ecrit avec succes
        Traitement col -Longitude-
Creation de la var Longitude de dimension SD -> OK
taille du tableau SD a ecrire:  1
Variable Longitude et ses datas ecrit avec succes
        Traitement col -Profondeur-
Creation de la var Profondeur de dimension Profondeur -> OK
taille du tableau 1D (var dimension) a ecrire:  2
Variable Profondeur et ses datas ecrit avec succes
        Traitement col -NO3-
Creation de la var NO3 de dimension ('Datetime', 'Profondeur') -> OK
taille du tableau 2D  a ecrire:  (298, 2)
Variable NO3 et ses datas ecrit avec succes
        Traitement col -NO2-
Creation de la var NO2 de dimension ('Datetime', 'Profondeur') -> OK
taille du tableau 2D  a ecrire:  (298, 2)
Variable NO2 et ses datas ecrit avec succes
```

Capture n°5/5 des messages généré par le programme dans la console

L'étape de la création des variables et surtout l'écriture des données est la plus détaillée car c'est celle qui génère le plus d'erreurs notamment lorsque le fichier n'est pas « propre ».

Le programme est disponible dans son intégralité en Annexes.

3.5 Conclusions

Ce stage m'a permis de réaliser certaines choses dont l'importance de la méthodologie en amont des erreurs et la communication en aval des erreurs.

3.5.1 La méthodologie

Pour réaliser le programme je suis parti d'une première version existante qui traitait les données en 1D, par exemple l'évolution de la température en fonction du temps : T=f(time).

Le but de la deuxième partie de mon stage a été de faire évoluer ce programme pour qu'il traite les données qui évoluent en 2Dimensions : par exemple lorsque la Température évolue à la fois en fonction du temps ET de la profondeur : T=f(time, profondeur).

J'ai donc dû comprendre la structure du programme et trouver comment se forment les jeux de données en tableaux 2D.

Aussi, j'ai été plus structuré dans la conception en faisant des schémas, des notes, en cloisonnant le programme. C'est-à-dire faire des fonctions en expliquant bien avec des cprint() (c'est une fonction qui permet de faire afficher des messages dans la console à la manière de print() mais en

couleur) et des commentaires dans le code indiquant toutes les étapes du programme car plus les étapes sont claires, mieux on peut déterminer d'où vient l'erreur et la repérer facilement.

Au plus c'est clair dans le programme, au moins il y a d'erreurs et au plus c'est facile d'ajouter des fonctionnalités par la suite par l'auteur du programme ou bien par une tout autre personne. En effet la compréhension du programme passe essentiellement par l'explicité des opérations.

3.5.2 Les problèmes rencontrés

- J'ai dû trouver un moyen de déterminer quelle colonne du fichier va être dimension 1 (D1) et dimension 2 (D2), chose qui n'était pas présente dans le programme de conversion en 1D et mettre en les structures de données 2D associés.
- Dans le cadre de la résolution d'erreurs, nous avons dû réfléchir à un moyen de rendre le programme plus flexible car les données sont souvent hétérogènes. Par exemple pour une fonction Température(station, profondeur) il n'y a pas forcément la même profondeur à toutes les stations donc des cases du tableaux devront être vide. Donc des situations où nous avons des stations avec des nombres de profondeurs différentes, tel que :

Profondeur / Station	1m	2m	3m	4m	5m
S 1	16	15	14		
S 2	17	16			
S 3	15	14	14	13	13

Avec cette situation mon programme rempli une matrice creuse avec des « Nan » pour les valeurs manquantes.

- Dans le cadre du traitement d'un type de fichier avec ce programme j'ai pu remarquer que la date des fichiers n'était pas au bon format. J'ai donc créé un programme (à partir d'un programme de Mr Libes) qui :
    - Met la date au bon format
    - Remplace les valeurs erronées par la valeur erronée standard de NetCDF
    - Remplace les « S » de Surface et « F » de Fond par leur valeur numérique

4 Conclusion

Durant ce stage, j'ai pu mettre en pratique mes connaissances et compétences en programmation Python acquises au cours de ma formation ainsi que les développer et les élargir au monde de l'entreprise et plus particulièrement à la gestion des datas.

Aussi j'ai dû faire preuve d'initiative, d'organisation, d'autonomie mais aussi de communication pour m'adapter au mieux au monde de l'entreprise.

J'ai pu fournir des outils qui me semble répondre aux attentes du Service d'Observation et dont je suis fier car ce sont mes premiers programmes à utilisations concrètes, utiles et respectant les objectifs fixés.

Aussi, j'ai pu développer ma méthodologie qui est, selon mon expérience, l'aspect le plus important dans la programmation.

Ce stage m'a fourni une expérience en entreprise que je souhaitais retrouver dès l'an prochain et qui vient confirmer et conforter mon choix de continuer mes études en alternance. Aussi, je souhaite développer mes compétences en programmation car elle requière une certaine rigueur qui me plait beaucoup et dans laquelle j'aime m'investir.

5 Remerciements

Je souhaite remercier le MIO pour m’avoir accueilli dans ses locaux et permis de réaliser mon stage.
Je tiens aussi à remercier le Service d’Observation de l’OSU Pythéas pour m’avoir laisser intégrer le service et en particulier Maurice Libes, mon tuteur qui m’a accordé sa confiance pour réaliser ces missions et qui a été disponible pour m’aider, répondre à mes interrogations et m’aider lorsque je rencontrais des difficultés techniques durant ces 10 semaines.
Aussi je souhaite remercier Didier Mallarino du SO qui s’est aussi investi dans la conception de mes programmes.
Enfin je souhaite remercier Delphine Rousseau pour m’avoir suivi tout au long de ma période de stage.

6 Glossaire

DUT, Diplôme Universitaire de Technologie.
MIO - Institut Méditerranéen d'Océanologie.
OSU - Observatoire des Sciences de l'Univers.
CNRS - Centre National de la Recherche Scientifique.
IRD - Institut de recherche pour le développement.
FAIR (data) - Findable, Accessible, Interoperate, Reusable.
Métadonnées - Caractéristique formelle normalisée et structurée utilisée pour la description et le traitement des contenus des ressources numériques.
CSV - Comma-separated values.
NetCDF - Network Common Data Format.
SSH - Secure Shell.
Erddap - Environmental Research Division Data Access Protocol.
IstSOS - Istituto Scienze della Terra Sensor Observation Service.
ISO - C'est un vecteur de valeurs d'une variable.
DF - Un DataFrame est une structure de données bidimensionnelle, c'est-à-dire que les données sont alignées de façon tabulaire en lignes et en colonnes.
BDD - Une base de données est une collection organisée d'informations structurées, généralement stockées électroniquement dans un système informatique.
Workflow - Un workflow de données est une série d'étapes dans le but d'effectuer une tâche.
Cron - Cron est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiée à l'avance, ou selon un cycle défini à l'avance.
NASA (NetCDF) - La National Aeronautics and Space Administration, plus connue sous son acronyme NASA, est l'agence fédérale responsable de la majeure partie du programme spatial civil des États-Unis.
Panoply - Panoply est un logiciel de visualisation de fichier NetCDF développé par la NASA.
SeriesPandas - C'est un vecteur de valeurs d'une variable.

7 Bibliographie

http://www.python-simple.com/

https://www.guillaumedueymes.com/

https://www.google.com

https://www.mio.osupytheas.fr/fr

https://pro.arcgis.com/

https://www.odatis-ocean.fr

Institut Universitaire de Technologie,
Aix-Marseille Université

ANNEXES
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Deux aspects de gestion de données interopérables à l'OSU Pythéas

Emilien Bonneton

Institut Méditerranéen d'Océanologie

Responsable entreprise : Maurice Libes

Responsable académique : Delphine Rousseau

2022

Table des matières

1 Programme de conversion Météo de la station de l'Etoile :

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
#
#  traitement des fichiers météo de la station CLIMED
#  mise au format "cooperate" pour l'application http://meteo.osupytheas.fr
#
#   on ouvre les fichiers climed d'une journée, et
#   on assemble les 2 fichiers d'entrée H:00 et H:30
#   dans un seul fichier de sortie cooperate a l'heure pleine H:00
#
#  Copyright 2022 Emilien Bonneton & Maurice Libes <maurice.libes@osupytheas.fr>
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your optiongit clone https://gitlab.osupytheas.fr/meteo-osu/scripts-meteo-osu.git) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
#  MA 02110-1301, USA.
#


import glob
import pandas as pd
import os
import re
from datetime import datetime
import argparse
from termcolor import cprint
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import copy
import math

####
def Get line(df,tableau):

    nl=len(df)
    regex = re.compile('^(((\d){4})(-(\d){2}){2})')
    i = 0
    while i < nl :
        ligne=df[i]
#on determine si c'est une ligne de donnee en cherchant une date en debut de ligne
        trouve = re.match(regex,str(ligne[0]))
        if trouve :
#si oui on traite la ligne pour avoir des donnees propres a mettre dans le tab de donnees
                ligne coop=Traite ligne(ligne)
                tableau.append(ligne coop)
        i = i + 1
    return tableau

####
def Traite ligne(ligne):
#traite le format de la date et les valeurs NAN et +INF
    ligne[0]=ligne[0].replace('-','/')+'.0000'
    #print("avant =>",ligne)
    for i in range(0,len(ligne)) :
        ligne[i] = str(ligne[i])
        if (ligne[i] == 'NAN') or (ligne[i] == '+INF') or (ligne[i] == 'inf'):
            ligne[i] = '9999.999'
    #print('apres ==>',ligne)
    return ligne

####
def CreaFicOut(date):
    global OutputDir,nomficout, fileout

    dateFicOut=date[:4]+'-'+date[4:6]+'-'+date[6:8]
    OutputDir=racineOut+dateFicOut+"/"+Centrale

#creation du dossier de sortie
    if (not os.path.exists(OutputDir)):
        os.makedirs(OutputDir)
        cprint("Creation Rep de sortie "+OutputDir, 'blue', attrs=['bold'])

    fileout=OutputDir+'001 '+date[:8]+'T'+date[8:11]+'00'+date[11:]

# creation fichier de sortie, on ecrase l'ancien fichier si deja existant
    try :
        FileCooperate=open(fileout,'w')
        cprint("Creation fichier "+fileout, 'blue', attrs=['bold'])
    except:
        cprint("Echec ouverture FileCooperate "+date, 'red', attrs=['bold'])
        sys.exit(0)

    return FileCooperate

####
def Write Entete Cooperate(FileCooperate):
# ecriture de l'entete meteo dans fichier cooperate
    FileCooperate.write("File name: "+fileout)
    FileCooperate.write(" \n");
    FileCooperate.write("Job:          JOB 001\n")
    FileCooperate.write("Schedule:     B \"JOB 001\" \n")
    FileCooperate.write("Serial num:  000001 \n")
    FileCooperate.write(" \n")
    FileCooperate.write("Channel map:\n");
    FileCooperate.write("0: none  name:\"HU INS Com\"  units:\"%\" \n")
```

```
    FileCooperate.write("0: none  name:\"HU INS Com\"  units:\"%\" \n")
    FileCooperate.write("1: none  name:\"T2m INS Com\"  units:\"deg C\" \n")
    FileCooperate.write("2: none  name:\"PYR INS Com\"  units:\"W/m2\" \n")
    FileCooperate.write("3: none  name:\"DDmoy INS Com\"  units:\"degres\" \n")
    FileCooperate.write("4: none  name:\"FFmoy INS Com\"  units:\"m/s\" \n")
    FileCooperate.write(" \n");
    FileCooperate.write("DATA start: \n")

####
def Get date atraiter():
# soit on passe une date en argument -d , soit on prend la date du jour
    parser = argparse.ArgumentParser(description='-d <date format YYYY-mm-dd> ')
    parser.add_argument('-d', '--date', help='Entrer une date à traiter(format YYYY-mm-dd)', required=False)
    args = parser.parse_args()

# -d date atraiter , si pas d'argument -d on prend la date du jour
    d=args.date
    if (d == None ):
        today=datetime.now()
        strtoday=today.strftime("%Y-%m-%d")
        date atraiter=strtoday
        #print("date a traiter ", date atraiter)
        cprint("Date a traiter "+date atraiter, 'yellow', attrs=['bold'])
    else:
        #print("date a traiter = ",d)
        cprint("date a traiter = "+d, 'yellow', attrs=['bold'])
        date atraiter = d

    return date atraiter

####
def Write line(tableau,FileCooperate):
#recupere les donnees dans tableau et les ecris dans FileCooperate
    print("tableau horaire",tableau)
    for ligne in range(0,len(tableau)): #ligne sont des listes correspondant aux lignes des fichiers d'entree
        for i donnee in range(0,len(tableau[ligne])): #donnee sont les index des donnees dans la ligne
            if (i donnee < len(tableau[ligne])-1): #pas de virgule a la fin des lignes
                FileCooperate.write(str(tableau[ligne][i donnee])+", ")
            else :
                FileCooperate.write(str(tableau[ligne][i donnee]))
        FileCooperate.write('\n')
    return 1

#### envoie un mail a l'equipe CTD
def Mailto(message):
    body=""

    msg = MIMEMultipart()
    msg['From'] = fromctd
    msg['To'] = dest
    msg['Subject'] = 'Erreur Programme Convert Climed'

    for lig in message:
        body=body+lig+'\n'

    msg.attach(MIMEText(body, 'plain'))

    server = smtplib.SMTP(mailserver, 25)
    server.ehlo()
    server.starttls()

    text=msg.as string()
    server.sendmail(fromctd, dest, text)
    server.quit()

    return 1

 ####
def CreaFicOut erddap istsos(date):

    fileout erddap=racineOutErddap+'meteo-climed-erddap-'+str(date)+'.csv'
    fileout istsos = racineOutIstSOS+'meteorologie climed '+date[:4]+date[5:7]+date[8:10]+'234500.dat'

    #creation du dossier de sortie erddap
    if (not os.path.exists(racineOutErddap)):
        os.makedirs(racineOutErddap)
        cprint("Creation Rep de sortie "+racineOutErddap, 'blue', attrs=['bold'])

    #creation du dossier de sortie istsos
    if (not os.path.exists(racineOutIstSOS)):
        os.makedirs(racineOutIstSOS)
        cprint("Creation Rep de sortie "+racineOutIstSOS, 'blue', attrs=['bold'])

    #creation fichier de sortie erddap
    try :
        file erddap=open(fileout erddap,'w')
        cprint("Creation fichier erddap "+fileout erddap, 'blue', attrs=['bold'])
    except:
        cprint("Echec creation fichier erddap "+fileout erddap, 'red', attrs=['bold'])
        sys.exit(0)

    #creation fichier de sortie erddap
    try :
        file istsos=open(fileout istsos,'w')
        cprint("Creation fichier istsos "+fileout istsos, 'blue', attrs=['bold'])
    except:
        cprint("Echec creation fichier istsos "+fileout istsos, 'red', attrs=['bold'])
        sys.exit(0)

    #entete erddap
    entete erddap="Datetime, Longitude, Latitude, Humidity (%), Temperature Air (degree C), Solar radiation (W.m^-2),
    Wind direction (degree), Wind speed (m.s^-1)"
    file erddap.write(entete erddap+'\n')

    #entete istsos
    entete istsos=
```

```
    entete istsos=
    "urn:ogc:def:parameter:x-istsos:1.0:time:iso8601,urn:ogc:def:parameter:x-istsos:1.0:longitude,urn:ogc:def:parameter
    :x-istsos:1.0:latitude,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:humidity,urn:ogc:def:parameter:x-istsos:1.0:met
    eo:air:temperature,urn:ogc:def:parameter:x-istsos:1.0:meteo:solar:radiation,urn:ogc:def:parameter:x-istsos:1.0:air:
    wind:direction,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:wind:speed"
    file istsos.write(entete istsos+'\n')

    return file erddap, file istsos

####
def Convert datetime(ligne,formficout):
    if formficout == 'istsos':
        ligne = ligne.replace('/','-')
        ligne = ligne.replace(' ','T')
        ligne = ligne[0:-5]
        ligne += '.000000+0000'
    elif formficout == 'erddap':
        ligne = ligne.replace('/','-')
        ligne = ligne.replace(' ','T')
        ligne = ligne[0:-5]
        #ligne += '+0100'
    return ligne

####
def Write Erddap Istsos(taball,file erddap,file ist):

    taball erddap = copy.deepcopy(taball)
    taball istsos = copy.deepcopy(taball)

    #boucle sur les heures pleines
    for HP in range(0,len(taball)):
        #boucle sur les quarts dheure (ou lignes)
        for QH in range(0,len(taball[HP])):
            #boucle sur les colonnes
                date erddap = Convert datetime(taball erddap[HP][QH][0],'erddap')
                date istsos = Convert datetime(taball istsos[HP][QH][0],'istsos')

                ligne = taball istsos[HP][QH][1:]

                #on ajoute latitude et longitude de la station climed
                ligne.insert(0, '5.42')
                ligne.insert(1, '43.37')

                ligne =  [str(i) for i in ligne] # conversion en string nécessaire pour les +INF

                ligne erddap= date erddap + ', ' + ', '.join(ligne)
                ligne istsos= date istsos + ', ' + ', '.join(ligne)

                #convention des valeur erronnees n'est pas la meme sur istsos
                ligne istsos = ligne istsos.replace('9999.999','-999.99')

                file erddap.write(ligne erddap)
                file ist.write(ligne istsos)
                file ist.write(ligne istsos)

                file erddap.write('\n')
                file ist.write('\n')
    return 0

####
def main(args):
    global data, racineOut, Centrale, numcol, fromctd, dest, mailserver, racineOutErddap, racineOutIstSOS

# racine en Dev Linux
    fromctd = "emilien.bonneton@mio.osupytheas.fr"
    dest = "emilien.bonneton@mio.osupytheas.fr"
    #mailserver='smtp.osupytheas.fr'
    #racineIn = "/home/bonneton/METEO/climedIN/"
    #racineOut = "/home/bonneton/METEO/climedOut/"
    #racineOutErddap = racineOut+'erddap/'
    #racineOutIstSOS = racineOut+'istsos/'

# racine en Prod
    #fromctd = "maurice.libes@osupytheas.fr"
    #dest = "maurice.libes@osupytheas.fr"
    racineIn = "/mnt/meteo-data/climed/"
    racineOut = "/data/climed/"
    mailserver='smtp.osupytheas.fr'
    racineOutErddap = '/mnt/SO-DATA/meteo-climed/erddap/'
    racineOutIstSOS = '/mnt/SO-DATA/meteo-climed/istsos/'

#nom de la centrale pour cooperate
    Centrale = "SN000001/"

#fonction prenant en compte la date en argument, ou bien la date du jour si pas d'argument
    date atraiter = Get date atraiter()

#motif du chemin absolu des fichiers a traites + sorted
    motif = racineIn+str(date atraiter)+"/CR3000*.txt"
    filelist = glob.glob(motif)
    filelist = sorted(filelist)

#body du mail erreur
    msg mail = []

#si la liste des fichiers a traites est vide alors on arrete le programme
    if (len(filelist) == 0):
        cprint(" pas de fichier a traiter pour la date "+date atraiter,'red', attrs=['bold'])
        msg mail.append('pas de fichier pour la date : ' +str(date atraiter))
        Mailto(msg mail)
        sys.exit(0)

#titre et numero des colonnes a recuperer dans les fichiers
    #headr = ["TIMESTAMP","RH","AirTC","pyr en","WD deg","WS ms"]
    numcol = [0,3,4,5,6,7]  # numero des colonnes a prendre dans le fichier d'entree

#init tableau qui stocke les donnees des fichiers H:00 et H:30
```

```
#init tableau qui stocke les donnees des fichiers H:00 et H:30
    tableau=[]

#init tableau qui sert a stocker les donnees de facon journalieres
    tab all=[]
#variable qui sert au erreur exceptionnelle
    one error=False #compteur pour envoyer un mail
    erreur fic prec=False #sert a creer un fichier a H:30 (si valide) si erreur au fichier H:00

#COEUR DU PROGRAMME
    for nomfic in filelist:
        reset tab=False #variable qui sert a reinitialiser le tableau

        cprint('Fichier traité : '+nomfic, 'green', attrs=['bold'])

        regex = re.compile('CR3000 (\d{8}) (\d{2}[03]0).txt$') ## on prend les fichiers a heure 00:00 ou 00:30
        match=regex.search(nomfic)
        if not (match is None):
            date,heure=match.groups()
        else :
            cprint('Erreur format nom de fichier :'+nomfic, 'red', attrs=['bold'])
            msg='Erreur nom du fichier : '+str(nomfic)+' heure non pleine'
            msg mail.append(msg)
            erreur fic prec=True
            one error=True
            continue


        fic out existe=racineOut+date[:4]+'-'+date[4:6]+'-'+date[6:8]+'/'+Centrale+'001 '+date[:8]+'T'+heure[:2]+
        '0000.txt'

        #if (os.path.exists(fic out existe)):
        #    cprint('Fichier deja traite : '+fic out existe,'green',attrs=['bold'])
        #    continue

#ouverture des fichiers dans un dataframe
        try :
            data = pd.read csv(nomfic, sep=",",usecols=numcol, header=None)  ##attention apres il y a des headers
        except :
            print("Pb ouverture Fichier : "+nomfic)
            sys.exit(0)

#transformation du dataframe en liste
        liste ligne=data.values.tolist()

#on recupere et stocke les donnees propres grace a getline
        tabligne=Get line(liste ligne,tableau)
        print("lon tab ",len(tabligne))
        #print(tabligne)
        if (len(tabligne)>1):
#on cree un nouveau fichier à l'heure 30 avec l'entete
            if (heure[2:] == '30' or  erreur fic prec):
                nomficout=date+heure[:2]+"00"+'.txt'
                FileCooperate = CreaFicOut(nomficout)
                Write Entete Cooperate(FileCooperate)
                Write Entete Cooperate(FileCooperate)
                reset tab=True #alors on peut vider le tableau de donnees des 2 fichiers H:00 et H:30
        else :
            cprint('Erreur fichier vide : '+nomfic,'red',attrs=['bold'])
            msg='Erreur fichier vide : '+str(nomfic)
            msg mail.append(msg)
            one error=True
#si H:30 alors on peut ecrire dans le fichier de sortie et reset le tableau de donnees
#s erreur fic preceden alors on ecrit
        if reset tab or erreur fic prec :
            #print("ecriture du tableau de ligneé")
            Write line(tabligne,FileCooperate)
            tab all.append(tabligne)
            tableau=[] #tableau qui stocke les donnees sur 1h
            erreur fic prec=False

    FicOut erddap, FicOut istsos = CreaFicOut erddap istsos(date atraiter)

    Write Erddap Istsos(tab all,FicOut erddap,FicOut istsos)

    if one error:
        Mailto(msg mail)
    return 0

if __name__ == '  main  ':
    import sys
    sys.exit(main(sys.argv))
```

2 Programme de conversion Météo de la station d'Endoume :

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-
#
#  convert-meteo-endoume.py
#  traitement des fichiers météo de la station Endoume
#  mise au format "cooperate" pour l'application http://meteo.osupytheas.fr
#
#  rajout production des fichiers meteo en CSV pour ERDDAP et ISTSOS (emilien bonneton)
#
#  Copyright 2022 Emilien Bonneton & libes <maurice.libes@osupytheas.fr>
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
#  MA 02110-1301, USA.
#
#
import os
import os.path
import re
import sys
import glob
import datetime
import time
import argparse
from datetime import date, time, datetime
from termcolor import colored, cprint
import pandas as pd
import dbm
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart


#
def Get date atraiter():
# soi ton passe une date en argument -d , soit on prend la date du jour
    parser = argparse.ArgumentParser(description='-d <date format YYYY-mm-dd> ')
    parser.add argument('-d', '--date', help='Entrer une date à traiter(format YYYY-mm-dd)', required=False)
    args = parser.parse args()

# -d date atraiter , si pas d'argument -d on prend la date du jour
    d=args.date
    if (d == None ):
        today=datetime.now()
        strtoday=today.strftime("%Y-%m-%d")
        date atraiter=strtoday
        print("date a traiter ", date atraiter)
    else:
        print("date a traiter = ",d)
        date atraiter = d

    return date atraiter
##
def Cree Fic Out Cooperate(pathfic, date atraiter):
    #cree le fchier de sortie cooperate dans le bon repertoire du jour
    global OutputDir,nomficout

    OutputDir=racineOut+"/"+date atraiter+"/"+Centrale+"/"
    if (not os.path.exists(OutputDir)):
        os.makedirs(OutputDir)
        cprint("Creation Rep de sortie "+OutputDir,'yellow', attrs=['bold'])


    f=os.path.basename(pathfic)
    pattern=re.compile('SME (\d{8})(\d{4}).txt')  #on decortique la date yyyymmdd et hhmm
    match=pattern.search(f)
    date,heure=match.groups()
    #print('date trouve ',date,heure)
    #nom du fchier de sortie
    nomficout="001 "+date+'T'+heure+"00.txt"  # exemple 001 20170104T101700.txt
    cprint('Creation fic cooperate : '+nomficout, 'blue', attrs=['bold'])
    PathFicout=OutputDir+"/"+nomficout
    FileCooperate=open(PathFicout,'w')

    return FileCooperate
##
##
def Write Entete Cooperate(FileCooperate):
# ecriture de l'entete meteo dans fichier cooperate
    FileCooperate.write("File name: "+OutputDir+nomficout)  ## PathFicout=OutputDir+"/"+nomficout
    FileCooperate.write(" \n");
    FileCooperate.write("Job:          JOB 001\n")
    FileCooperate.write("Schedule:    B \"JOB 001\" \n")
    FileCooperate.write("Serial num:  000001 \n")
    FileCooperate.write(" \n")
    FileCooperate.write("Channel map:\n");
    FileCooperate.write("0: none  name:\"T2m INS Com\"  units:\"deg C\" \n")
    FileCooperate.write("1: none  name:\"HU INS Com\"  units:\"%\" \n")
    FileCooperate.write("2: none  name:\"Td INS Com\"  units:\"deg C\" \n")
    FileCooperate.write("3: none  name:\"Pmer INS Com\"  units:\"hPa\" \n")
    FileCooperate.write("4: none  name:\"FFmoy INS Com\"  units:\"m/s\" \n")
    FileCooperate.write("5: none  name:\"FFX INS Com\"  units:\"m/s\" \n")
    FileCooperate.write("6: none  name:\"DDmoy INS Com\"  units:\"degres\" \n")
    FileCooperate.write("7: none  name:\"RR INS Com\"  units:\"mm\" \n")
    FileCooperate.write(" \n");
    FileCooperate.write("DATA start: \n")
```

```
    FileCooperate.write("DATA start: \n")

    return 1
##
def Detect Nan(ligne):
    #s'il y a un seul NaN on retourne True
    #print("ligne ",ligne,len(ligne))
    nan=False
    for v in ligne:
        if (pd.isna(v)):
            nan=True
            #print("nan = ",nan)
            return nan

    #print("nan = ",nan)
    return nan
##
def Get New Data(df):
    global newligne

    nl,nc=df.shape  # les dimensions du dataframe
    i=new=0
    newligne={}
    while i < nl:   #parcours des lignes du dataframe
        ligne=df.iloc[i].values
        #print("ligne ",i,ligne)
        date=ligne[0]; time=ligne[1]
        date = date.replace('-','/')   # cooperate veut un format date avec des "/"
        time=time+":00.0000"  #cooperate veut la chaine avec secondes et microsecondes
        #print("date, time", date, time)
        chaine=keydbm=date+" "+time
        ligne chaine=[str(i) for i in ligne[2:]] # conversion des donnees en chaine string apres la date et l'heure
        val=", ".join(ligne chaine)   #on separe les valeurs par ", "
        chaine=chaine+", "+val
        valuedbm=val
        #print("ligne chaine",ligne chaine)
        existe=fdbm.get(keydbm)  #on cherche dans la BD DBM si une clé date:heure existe deja
        nan=Detect Nan(ligne)
        if (not nan):  # on ecrit la ligne s'il n'ya aucun NaN dans la ligne
            if (existe):
                #cprint("Cet enregistrement existe deja : "+keydbm, 'red', attrs=[])
                deja=True
            else:
                new+=1
                #FileCooperate.write(chaine+"\n");
                newligne[i]=chaine
                #print("keydbm ",keydbm); print("valuedbm ",valuedbm)
                fdbm[keydbm]=valuedbm
                cprint("Nouvel enregistrement : "+keydbm+" : "+valuedbm, 'green', attrs=[])
        i+=1

    if (new==0):
            cprint("Aucun enregistrement nouveau ", 'yellow', attrs=[])
            cprint("Aucun enregistrement nouveau ", 'yellow', attrs=[])
            cprint("fichier deja traité ", 'yellow', attrs=['bold'])

    return new

def CreaFicOut erddap istsos(date):

    #init variable pour tester l'existence des fichiers pour ecrire l'entete qu'une fois
    existe pas erddap=False
    existe pas istsos=False

    fileout erddap=racineOutErddap+'meteo-endoume-erddap-'+date+'.csv'
    fileout istsos = racineOutIstSOS+'meteorologie endoume '+date[:4]+date[5:7]+date[8:10]+'234500.dat'

    #creation du dossier de sortie erddap
    if (not os.path.exists(racineOutErddap)):
        os.makedirs(racineOutErddap)
        cprint("Creation SME 202205062005.txtRep de sortie "+racineOutErddap, 'blue', attrs=['bold'])

    #creation du dossier de sortie istsos
    if (not os.path.exists(racineOutIstSOS)):
        os.makedirs(racineOutIstSOS)
        cprint("Creation Rep de sortie "+racineOutIstSOS, 'blue', attrs=['bold'])

    #test pour determiner si le fichier a deja ete cree ou non
    if (not os.path.exists(fileout erddap)):
        existe pas erddap=True

    #creation fichier de sortie erddap
    try :
        file erddap=open(fileout erddap,'a')
        cprint("Creation fichier erddap "+fileout erddap, 'blue', attrs=['bold'])
    except:
        cprint("Echec creation fichier erddap "+fileout erddap, 'red', attrs=['bold'])
        sys.exit(0)

    #test pour determiner si le fichier a deja ete cree ou non
    if (not os.path.exists(fileout istsos)):
        existe pas istsos=True

    #creation fichier de sortie istsos
    try :
        file istsos=open(fileout istsos,'a')
        cprint("Creation fichier istsos "+fileout istsos, 'blue', attrs=['bold'])
    except:
        cprint("Echec creation fichier istsos "+fileout istsos, 'red', attrs=['bold'])
        sys.exit(0)

    if existe pas erddap :
        #entete erddap
        entete erddap="Datetime, temperature[C], humidity[%], dew point[C], sealevel pressure[hPa], avg wind
        speed[m/s], gust speed[m/s], winddir[degree], rainfall00, latitude, longitude"
        file erddap.write(entete erddap+'\n')

    if existe pas istsos :
```

```
    if existe pas istsos :
        #entete istsos
        entete istsos=
        "urn:ogc:def:parameter:x-istsos:1.0:time:iso8601,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:temperature,urn:o
        gc:def:parameter:x-istsos:1.0:meteo:air:humidity,urn:ogc:def:parameter:x-istsos:1.0:meteo:dew:point,urn:ogc:def
        :parameter:x-istsos:1.0:meteo:air:pressure,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:wind:speed,urn:ogc:def:
        parameter:x-istsos:1.0:air:gust:speed,urn:ogc:def:parameter:x-istsos:1.0:air:wind:direction,urn:ogc:def:paramet
        er:x-istsos:1.0:meteo:precipitation cumul 1h,urn:ogc:def:parameter:x-istsos:1.0:longitude,urn:ogc:def:parameter
        :x-istsos:1.0:latitude"
        file istsos.write(entete istsos+'\n')

    return file erddap, file istsos

def Convert datetime(ligne,formficout):
    if formficout == 'erddap':
        ligne = ligne.replace('/','-')
        ligne = ligne.replace('.0000','')
        ligne = ligne[:10]+'T'+ligne[11:]
    elif formficout == 'istsos':
        ligne = ligne.replace('/','-')
        ligne = ligne.replace('.0000','.000000+0000')
        ligne = ligne[:10]+'T'+ligne[11:]
    return ligne

##
def Open DBM Endoume():
    global fdbm
    try:
        fdbm=dbm.open(Dir Appli+"dbm endoume.dat", flag='c', mode=438)
    except:
        cprint('erreur ouverture fichier DBM','red',attrs=['bold'])
        sys.exit(0)

    return fdbm

#### envoie un mail a l'equipe CTD
def Mailto(message):
    body=""

    msg = MIMEMultipart()
    msg['From'] = fromctd
    msg['To'] = dest
    msg['Subject'] = 'Programme Convert Meteo Endoume'

    for lig in message:
        body=body+lig+'\n'

    msg.attach(MIMEText(body, 'plain'))

    server = smtplib.SMTP(mailserver, 25)
    server.ehlo()
    server.starttls()

    text=msg.as_string()
    server.sendmail(fromctd, dest, text)
    server.quit()

    return 1

##
def main(args):
    global date atraiter
    global racineIn, racineOut, Dir Appli
    global Centrale
    global fdbm
    global racineOutErddap, racineOutIstSOS
    global fromctd, dest, mailserver

    #Mail
    fromctd = "maurice.libes@osupytheas.fr"
    dest = "maurice.libes@osupytheas.fr"
    mailserver = 'smtp.osupytheas.fr'

    Centrale="SN000001"
    #PROD
    racineIn="/mnt/meteo-data/endoume"
    racineOut="/data/endoume/"
    Dir Appli="/appli/MeteoFiles 2 Cooperate/"

    # DEV
    #racineIn="../endoumeIN"
    #racineOut="../endoumeOUT/"
    #Dir Appli="./"
    racineOutErddap = '/mnt/SO-DATA/meteo-endoume/erddap/'
    racineOutIstSOS = '/mnt/SO-DATA/meteo-endoume/istsos/'


# ces 2 tableaux ne servent a rien : je les ai mis pour la comprhension
#conversion des libelles des fichiers dans les libelles pour la BD cooperate
    Tab Col Conversion={" temperature[C]":"T2m INS Com", \
    " humidity[%]":"HU INS Com", \
    " dew point[C]":"Td INS Com", \
    " sealevel pressure[hPa]":"Pmer INS Com",\
    " avg wind speed[m/s]":"FFmoy INS Com", \
    " gust speed[m/s]":"FFX INS Com", \
    " winddir":"DDmoy INS Com",\
    " rainfall00":"RR INS Com"}
#association des libellés et unités
    Tab Param Units={"T2m INS Com":"deg C",\
    "HU INS Com":"%", \
    "Td INS Com":"deg C", \
    "FFmoy INS Com":"m/s", \
    "DDmoy INS Com":"degres", \
    "FFX INS Com":"m/s", \
    "Pmer INS Com":"hPa", \
    "RR INS Com":"mm"}

    # ce tableau ne sert a rien : c'est pour bien visualiser le nom des colonnes
```

```
    # ce tableau ne sert a rien : c'est pour bien visualiser le nom des colonnes
    colonnes=["# date"," time"," temperature[C]"," humidity[%]"," dew point[C]"," sealevel pressure[hPa]", \
    " avg wind speed[m/s]"," gust speed[m/s]"," winddir"," rainfall00"]
    print("libelles des colonnes ",colonnes)

    date atraiter=Get date atraiter()
    InputFiles=racineIn+"/"+date atraiter+"/"

    #FicOut erddap, FicOut istsos = CreaFicOut erddap istsos(date atraiter)

    fileNames=glob.glob(InputFiles+"SME *.txt")   ##SME 202203031103.txt
    fileNames=sorted(fileNames)
    cprint("InputFiles "+InputFiles, 'green', attrs=['bold'])

    #body du mail erreur
    msg mail = []

    if (len(fileNames)==0):
        cprint("==> pas de fichiers a traiter pour la date "+date atraiter, 'red', attrs=['bold'])
        msg mail.append('Pas de fichier pour la date : ' +str(InputFiles))
        Mailto(msg mail)
    #print("fichiers a traiter : ",fileNames)

#boucle sur fichiers
    fdbm=Open DBM Endoume()
    for nomfic in fileNames:
        cprint("\nTraitement : "+nomfic, 'green', attrs=['bold'])
        try:
            df = pd.read csv(nomfic, sep = ',', header = 0, index col=None)
            #print(df.head(3))
        except:
            cprint("Pb ouverture Fichier : "+nomfic, 'red', attrs=['bold'])
            sys.exit(0)


        new=Get New Data(df)  # recherche les nouveaux enregistrmeents a inserer dans cooperate
        if (new >=1):
            FileCooperate=Cree Fic Out Cooperate(nomfic,date atraiter)
            Write Entete Cooperate(FileCooperate)
            FicOut erddap, FicOut istsos = CreaFicOut erddap istsos(date atraiter)
            print("len ",len(newligne),newligne)
            for cle in newligne.keys():
                ligne=newligne[cle]
                cprint("ecriture de : "+ligne,'green',attrs=['bold'])
                FileCooperate.write(ligne+"\n");


                #traitement pour erddap et istsos
                #on met le datetime au bon format
                ligne erddap = Convert datetime(ligne,'erddap')
                ligne istsos = Convert datetime(ligne,'istsos')
                #on ecrit dans les fichiers erddap et istsos en ajoutant les colonnes liés à latitude et longitude
                FicOut erddap.write(ligne erddap+', 5.3474133, 43.2807698'+"\n")
                FicOut istsos.write(ligne istsos+', 5.3474133, 43.2807698'+"\n")
                 FicOut istsos.write(ligne istsos+', 5.3474133, 43.2807698'+"\n")


    return 0


if __name__ == '__main__':
    import sys
    sys.exit(main(sys.argv))
```

3 Programme de conversion Météo de la station de Toulon :

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
#
#  convert-meteo-toulon.py
#  traitement des fichiers météo de la station de Toulon/UTLN
#  mise au format "cooperate" pour l'application http://meteo.osupytheas.fr
#
#  Copyright 2022 mallarino <didier.mallarino@osupytheas.fr>
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
#  MA 02110-1301, USA.
#
#
import os
import os.path
import re
import sys
import glob
import datetime
import time
import argparse
from datetime import date, time, datetime
from termcolor import colored, cprint
import pandas as pd
import dbm


# ------------------------------------------------------------------------------------------------
#           Creation fichiers de sortie istSOS et Erddap
# ------------------------------------------------------------------------------------------------
def CreaFicOut erddap istsos(date):

    #init variable pour tester l'existence des fichiers pour ecrire l'entete qu'une fois
    existe pas erddap=False
    existe pas istsos=False

    fileout erddap=racineOutErddap+'meteo-toulon-erddap-'+date+'.csv'
    fileout istsos = racineOutIstSOS+'meteorologie toulon '+date[:4]+date[5:7]+date[8:10]+'235400.dat'

    #creation du dossier de sortie erddap
    if (not os.path.exists(racineOutErddap)):
        os.makedirs(racineOutErddap)
        cprint("Creation Rep de sortie "+racineOutErddap, 'blue', attrs=['bold'])

    #creation du dossier de sortie istsos
    if (not os.path.exists(racineOutIstSOS)):
        os.makedirs(racineOutIstSOS)
        cprint("Creation Rep de sortie "+racineOutIstSOS, 'blue', attrs=['bold'])

    #test pour determiner si le fichier a deja ete cree ou non
    if (not os.path.exists(fileout erddap)):
        existe pas erddap=True

     #test pour determiner si le fichier a deja ete cree ou non
    if (not os.path.exists(fileout istsos)):
        existe pas istsos=True

    #creation fichier de sortie erddap
    try :
        file erddap=open(fileout erddap,'a')
        cprint("Creation/ouverture fichier erddap "+fileout erddap, 'blue', attrs=['bold'])
    except:
        cprint("Echec creation fichier erddap "+fileout erddap, 'red', attrs=['bold'])
        sys.exit(0)

    #creation fichier de sortie erddap
    try :
        file istsos=open(fileout istsos,'a')
        cprint("Creation/ouverture fichier istsos "+fileout istsos, 'blue', attrs=['bold'])
    except:
        cprint("Echec creation fichier istsos "+fileout istsos, 'red', attrs=['bold'])
        sys.exit(0)

    if existe pas erddap :
        #entete erddap
        entete erddap="Datetime, Temperature (degree C), Humidity (%), Rainfall (6mn), Solar radiation (W.m^-2), Wind
        Direction (degree), Wind speed (m.s^-1), Gust Speed (m.s^-1), Gust Direction (degree), Air-pressure at
        station level (hPa), Air-pressure at sea levele (hPa), Longitude, Latitude"
        file erddap.write(entete erddap+'\n')

    if existe pas istsos :
        #entete istsos
        entete istsos=
        "urn:ogc:def:parameter:x-istsos:1.0:time:iso8601,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:temperature,urn:o
        gc:def:parameter:x-istsos:1.0:meteo:air:humidity,urn:ogc:def:parameter:x-istsos:1.0:meteo:precipitation cumul 6
        mn,urn:ogc:def:parameter:x-istsos:1.0:meteo:solar:radiation,urn:ogc:def:parameter:x-istsos:1.0:air:wind:directi
        on,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:wind:speed,urn:ogc:def:parameter:x-istsos:1.0:air:gust:speed,ur
        n:ogc:def:parameter:x-istsos:1.0:air:gust:direction,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:pressure stati
        on level,urn:ogc:def:parameter:x-istsos:1.0:meteo:air:pressure,urn:ogc:def:parameter:x-istsos:1.0:longitude,urn
        :ogc:def:parameter:x-istsos:1.0:latitude"
        file istsos.write(entete istsos+'\n')

    return file erddap, file istsos


# ------------------------------------------------------------------------------------------------
#           Conversion datetime en iso8601
```

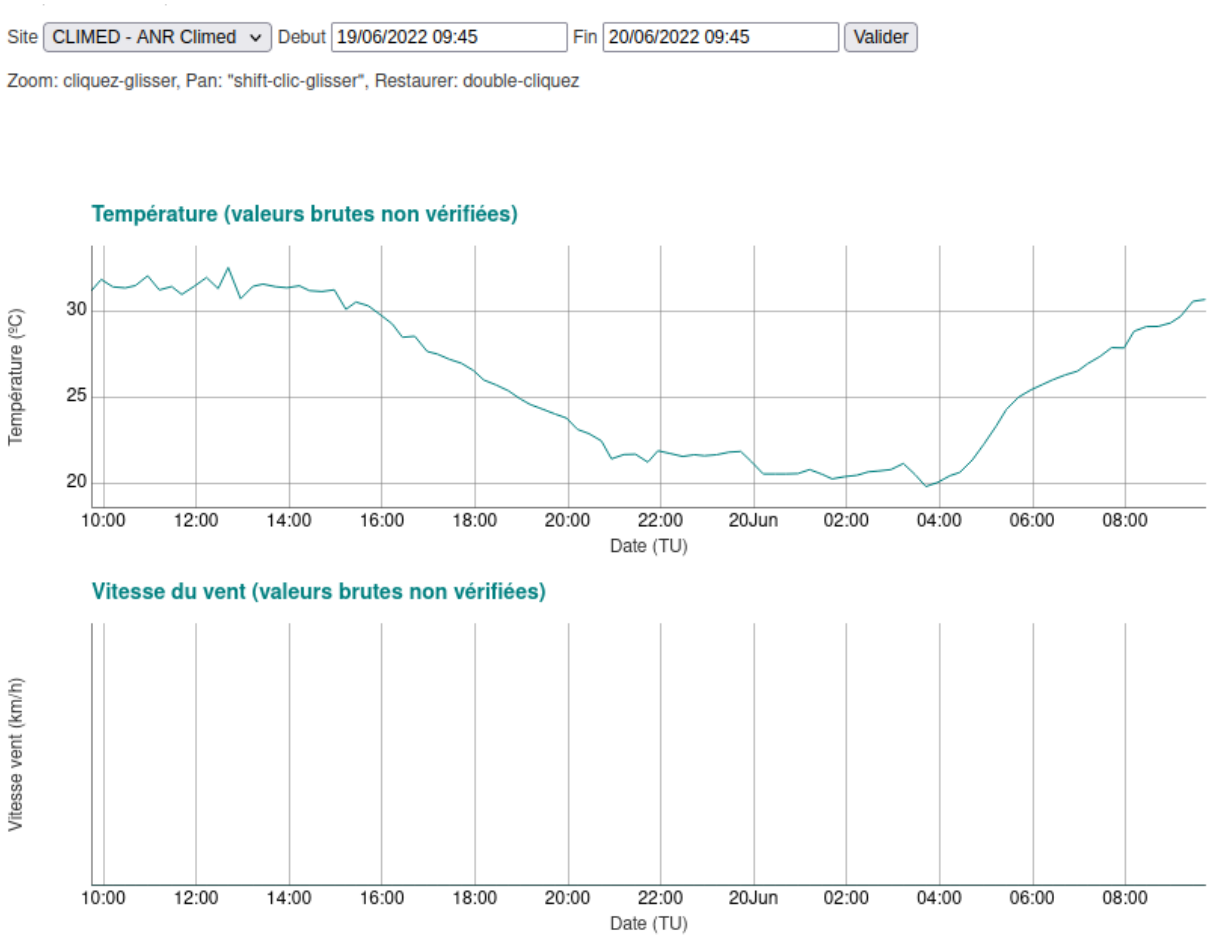
```
# ------------------------------------------------------------------------------------------------
#          Conversion datetime en iso8601
# ------------------------------------------------------------------------------------------------
def Convert datetime(ligne,formficout):
    if formficout == 'istsos':
        ligne = ligne.replace('/','-')
        ligne = ligne.replace('.0000','')
        ligne = ligne[:10]+'T'+ligne[11:]
        ligne = ligne.replace('9999.999','-999.99')
    elif formficout == 'erddap':
        ligne = ligne.replace('/','-')
        ligne = ligne.replace('.0000','')
        ligne = ligne[:10]+'T'+ligne[11:]
        #ligne += '+0100'
    return ligne
# ------------------------------------------------------------------------------------------------
#       Nom du fichier de sortie
# ------------------------------------------------------------------------------------------------
def Cree Fic Out Cooperate(pathfic):
    # Cree le fchier de sortie cooperate dans le bon repertoire du jour
    global OutputDir,nomficout

    OutputDir=racineOut+"/"
    if (not os.path.exists(OutputDir)):
        os.makedirs(OutputDir)
        cprint("Creation Rep de sortie "+OutputDir,'yellow', attrs=['bold'])

    # f=os.path.basename(pathfic)
    # pattern=re.compile(r'(\d{2}) (\d{4})')  #on cherche la date du fichier mmyyyy // Exemple :
    00006 010 1 COMPLEMENTAIRE mm yyyy.TXT
    # result=pattern.search(f)
    # mois,annee=result.groups(f)
    # print("File=",f,"Pattern=",result,'Date trouvee ',mois,annee)
    #Nom du fchier de sortie = date de l'execution
    today=datetime.now()
    strtoday=today.strftime("%Y%m%dT%H%M%S")
    # nomficout="001 "+annee+mois+'ddT'+"hhmmss.txt"  ### exemple 001 202112ddThhmmss.txt
    nomficout="001 "+strtoday+".txt"
    cprint('Creation fichier de sortie Cooperate : '+nomficout, 'blue', attrs=['bold'])
    PathFicout=OutputDir+"/"+nomficout
    print("Out=",PathFicout)
    # sys.exit(0)
    FileCooperate=open(PathFicout,'w')
    return FileCooperate

# ------------------------------------------------------------------------------------------------
#       Entete du fichier Cooperate
# ------------------------------------------------------------------------------------------------
def Write Entete Cooperate(FileCooperate):
    # ecriture de l'entete meteo dans fichier cooperate
    # Definition du header
    FileCooperate.write("File name: "+racineOut+"/"+nomficout+" \n");
    FileCooperate.write("Job:         JOB 001\n")
    FileCooperate.write("Schedule:    B \"JOB 001\" \n")
    FileCooperate.write("Serial num:  000001 \n")
    FileCooperate.write("Serial num:  000001 \n")
    FileCooperate.write(" \n")
    FileCooperate.write("Channel map:\n");
    FileCooperate.write("0: none  name:\"T2m INS Com\"  units:\"deg C\"\n")
    FileCooperate.write("1: none  name:\"HU INS Com\"  units:\"%\" \n")
    FileCooperate.write("2: none  name:\"RR INS Com\"  units:\"mm\" \n")
    FileCooperate.write("3: none  name:\"PYR INS Com\"  units:\"J/cm2\" \n")
    FileCooperate.write("4: none  name:\"DDmoy INS Com\"  units:\"degre\" \n")
    FileCooperate.write("5: none  name:\"FFmoy INS Com\"  units:\"m/s\" \n")
    FileCooperate.write("6: none  name:\"FFX INS Com\"  units:\"m/s\" \n")
    FileCooperate.write("7: none  name:\"DDX INS Com\"  units:\"degre\" \n")
    FileCooperate.write("8: none  name:\"Psfc INS Com\"  units:\"hpa\" \n")
    FileCooperate.write("9: none  name:\"Pmer INS Com\"  units:\"hpa\" \n")
    FileCooperate.write(" \n");
    FileCooperate.write("DATA start: \n")
    return 1

# ------------------------------------------------------------------------------------------------
#               Fonction de recuperation des nouvelles donnees
#                 Conserve une trace dans le dbm
# ------------------------------------------------------------------------------------------------
def Get New Data(df):
    global newligne
    nl,nc=df.shape  # les dimensions du dataframe
    i=new=0
    newligne={}
    while i < nl:   #parcours des lignes du dataframe
        ligne=df.iloc[i].values
        # On ignore les lignes qui commencent par un nombre negatif / Pas de donnees dedans
        if (ligne[0]>0):
            keydbm=ligne[2]
            existe=fdbm.get(keydbm)  #on cherche dans la BD DBM si une clé date:heure existe deja
            if (forceLeRetraitement==1):
                if (existe):
                    cprint("enregistrement "+keydbm+" existe ",'blue',attrs=['bold'])
                    del fdbm[keydbm]
                    cprint("destruction de "+keydbm,'blue',attrs=['bold'])
                    existe=fdbm.get(keydbm)  #on cherche dans la BD DBM si une clé date:heure existe deja
            if (existe):
                #cprint("Cet enregistrement existe deja : "+keydbm, 'red', attrs=[])
                deja=True
            else:
                dte,heure=keydbm.split(" ")
                j,m,a=dte.split("/")
                dateTime=a+"/"+m+"/"+j+" "+heure+".0000"        #print("date, time", date, time)
                new+=1
                val=str(ligne[4])+", "+str(ligne[5])+", "+str(ligne[7])+", "+str(ligne[8])+", "+str(ligne[11])+", "+
                str(ligne[12])+", "
                val=val+str(ligne[14])+", "+str(ligne[13])+", "+str(ligne[15])+", "+str(ligne[16])
                # On remplace les nan par 9999.999 + concatenation avec la date
                chaine=dateTime+", "+val.replace("nan","9999.999")
                newligne[i]=chaine
                fdbm[keydbm]="1"
                # cprint("Nouvel enregistrement : "+keydbm+" : 1", 'green', attrs=[])
```

```
                # cprint("Nouvel enregistrement : "+keydbm+" : 1", 'green', attrs=[])
            i+=1
    if (new==0):
        cprint("Aucun enregistrement nouveau ", 'yellow', attrs=[])
        cprint("fichier deja traité ", 'yellow', attrs=['bold'])
    else:
        cprint(str(new)+" lignes retenues / "+str(i)+" lignes lues", 'green', attrs=['bold'])
    return new

# ------------------------------------------------------------------------------------------------
#                Gestion en BdD des donnees converties
# ------------------------------------------------------------------------------------------------
def Open DBM Toulon():
    global fdbm
    try:
        fdbm=dbm.open(Dir Appli+"dbm toulon.dat", flag='c', mode=438)
    except:
        cprint('erreur ouverture fichier DBM','red',attrs=['bold'])
        sys.exit(0)

    return fdbm


# ------------------------------------------------------------------------------------------------
#                Traitement d'un fichier
# ------------------------------------------------------------------------------------------------
def traitementToulon(nomDuFichier):
    fdbm=Open DBM Toulon()
    cprint("\nTraitement : "+nomDuFichier, 'green', attrs=['bold'])
    # print ("Fichier à traiter : ",nomDuFichier)
    try:
        # On ignore la premiere ligne du fichier (header)
        df = pd.read csv(nomDuFichier, sep = '\t', header = 0, index col=None)
    except:
        cprint("Pb ouverture Fichier : "+nomfic, 'red', attrs=['bold'])
        fdbm.close()
        sys.exit(5) # Fichier illisible
    # On a pu lire le fichier ; On recupere les donnes qui ne sont pas deja dans le fichier dbm
    new=Get New Data(df)
    if (new >=1):  # Il y a de nouvelles donnees
        FileCooperate=Cree Fic Out Cooperate(nomDuFichier)
        print ("Fichier à traiter : ",nomDuFichier)
        Write Entete Cooperate(FileCooperate)
        cprint("writing "+str(new)+" lignes",'green',attrs=['bold'])


        for cle in newligne.keys():
            # cprint("ecriture de : "+newligne[cle],'green',attrs=['bold'])
            FileCooperate.write(newligne[cle]+"\n");
            #print(type(newligne[cle]),newligne[cle])

            date traitee = newligne[cle].split(' ')[0].replace('/','-')
            fileout erddap=racineOutErddap+'meteo-endoume-erddap-'+date traitee+'.csv'
            fileout istsos = racineOutIstSOS+'meteorologie endoume '+date traitee[:4]+date traitee[5:7]+date traitee[8
            :10]+'234500.dat'
            fileout istsos = racineOutIstSOS+'meteorologie endoume '+date traitee[:4]+date traitee[5:7]+date traitee[8
            :10]+'234500.dat'

            #si le fichier n'existe pas deja, on le cree
            if not (os.path.exists(fileout erddap) and os.path.exists(fileout erddap)):
                FicOut erddap, FicOut istsos = CreaFicOut erddap istsos(date traitee)

            #convention valeurs erronees differentes
            ligne istsos = Convert datetime(newligne[cle],'istsos')
            ligne erddap = Convert datetime(newligne[cle],'erddap')


            #convention format datetime different pour istsos
            ligne istsos=ligne istsos[:19]+'.000000+0000'+ligne istsos[19:]

            #on ecrit dans les fichiers erddap et istsos en ajoutant les colonnes liés à latitude et longitude
            try :
                FicOut erddap.write(ligne erddap+', 6.011558, 43.135978'+"\n")
                FicOut istsos.write(ligne istsos+', 6.011558, 43.135978'+"\n")
            except UnboundLocalError :
                cprint('fichiers de sortie erddap et/ou istsos existent deja','red',attrs=['bold'])


        FileCooperate.close()
    else:
        cprint("==> pas de nouvelles data", 'red', attrs=['bold'])
    fdbm.close()
    return 0


# ------------------------------------------------------------------------------------------------
#                Partie principale du code
# ------------------------------------------------------------------------------------------------

def main(args):
    global file atraiter
    global date atraiter
    global racineIn, racineOut, Dir Appli
    global Centrale
    global fdbm
    global forceLeRetraitement
    global racineOutErddap, racineOutIstSOS


    # Inistialisation des variables
    forceLeRetraitement=0
    Centrale="SN000001"
    # ------------- Repertoires de PROD
    racineIn="/mnt/meteo-data/la-garde/"
    racineOut="/data/toulon/"
    Dir Appli="/appli/MeteoFiles 2 Cooperate/"
    racineOutErddap='/mnt/SO-DATA/meteo-toulon/erddap/'
    racineOutIstSOS='/mnt/SO-DATA/meteo-toulon/istsos/'
    # ------------- Repertoires de DEV
    #racineIn="../toulonIN/"
```

```
            fileout istsos = racineOutIstSOS+'meteorologie endoume '+date traitee[:4]+date traitee[5:7]+date traitee[8
            :10]+'234500.dat'

            #si le fichier n'existe pas deja, on le cree
            if not (os.path.exists(fileout erddap) and os.path.exists(fileout erddap)):
                FicOut erddap, FicOut istsos = CreaFicOut erddap istsos(date traitee)

            #convention valeurs erronees differentes
            ligne istsos = Convert datetime(newligne[cle],'istsos')
            ligne erddap = Convert datetime(newligne[cle],'erddap')


            #convention format datetime different pour istsos
            ligne istsos=ligne istsos[:19]+'.000000+0000'+ligne istsos[19:]

            #on ecrit dans les fichiers erddap et istsos en ajoutant les colonnes liés à latitude et longitude
            try :
                FicOut erddap.write(ligne erddap+', 6.011558, 43.135978'+"\n")
                FicOut istsos.write(ligne istsos+', 6.011558, 43.135978'+"\n")
            except UnboundLocalError :
                cprint('fichiers de sortie erddap et/ou istsos existent deja','red',attrs=['bold'])


        FileCooperate.close()
    else:
        cprint("==> pas de nouvelles data", 'red', attrs=['bold'])
    fdbm.close()
    return 0


# ------------------------------------------------------------------------------------------------
#               Partie principale du code
# ------------------------------------------------------------------------------------------------

def main(args):
    global file atraiter
    global date atraiter
    global racineIn, racineOut, Dir Appli
    global Centrale
    global fdbm
    global forceLeRetraitement
    global racineOutErddap, racineOutIstSOS


    # Inistialisation des variables
    forceLeRetraitement=0
    Centrale="SN000001"
    # ------------- Repertoires de PROD
    racineIn="/mnt/meteo-data/la-garde/"
    racineOut="/data/toulon/"
    Dir Appli="/appli/MeteoFiles 2 Cooperate/"
    racineOutErddap='/mnt/SO-DATA/meteo-toulon/erddap/'
    racineOutIstSOS='/mnt/SO-DATA/meteo-toulon/istsos/'
    # ------------- Repertoires de DEV
    #racineIn="../toulonIN/"
    #racineIn="../toulonIN/"
    #racineOut="../toulonOUT/"
    #Dir Appli="./"
    #racineOutErddap=racineOut+'erddap/'
    #racineOutIstSOS=racineOut+'istsos/'

    # Analyse des arguments de la ligne de commande
    parser = argparse.ArgumentParser(description='Traitement des arguments')
    parser.add argument('--file', help='Entrer le nom du fichier a traiter', required=False)
    parser.add argument('--force', help='Force le retraitement des données', required=False)
    args = parser.parse args()
    if (args.force != None ):
        cprint("==> On va forcer le retraitement des données mêmes si elles sont déjà  ", 'red', attrs=['bold'])
        forceLeRetraitement=1
    f=args.file
    if (f != None ):
        if (not os.path.isfile(f)):
            print("File does not exist. Exiting with error 1")
            sys.exit(1)
        else:
            file atraiter=f
    else:
        file atraiter=""
    # ------------------------------------------------------------------------------------------------
    # ------ Cas ou on passe un seul fichier en option avec -f
    # ------ Dans le cas de Toulon, on peut traiter le fichier
    # ------ 00006 010 1 COMPLEMENTAIRE 00 0000.TXT
    # ------ qui ne change jamais de nom
    # ------------------------------------------------------------------------------------------------
    # file atraiter=Get file atraiter()
    if file atraiter !="" :
        traitementToulon(file atraiter)
        sys.exit(0) # On a traiter le fichier, on sort / C'est une option oneshot
    else:
        print("Aucun fichier en ligne de commande, je regarde dans "+racineIn)

    # ------------------------------------------------------------------------------------------------
    # ------ Pas de fichier, on regarde dans racineIn
    # ------------------------------------------------------------------------------------------------
    # ---- Traitement automatique sur une arborescence complete de data
    # On cherche les fichiers "6 minutes"
    fileNames=glob.glob(racineIn+"00006 010 1 COMPLEMENTAIRE *.TXT")   ## Exemple :
    00006 010 1 COMPLEMENTAIRE 12 2021.TXT
    #print(fileNames)
    inputFilesNames=sorted(fileNames)
    cprint("Racine de recherche "+racineIn, 'green', attrs=['bold'])
    if (len(inputFilesNames)==0):
        cprint("==> pas de fichiers a traiter ", 'red', attrs=['bold'])
        sys.exit(0)  # Pas de fichiers, on sort
    # Il y a des fichiers a traiter, on boucle sur la liste
    for nomFic in fileNames:
        traitementToulon(nomFic)
    sys.exit(0) # On a traiter le fichier, on sort


if __name__ == '__main__':
    import sys
    sys.exit(main(sys.argv))
```

4 Graphique disponible grâce aux programmes de conversion Météo

4.1 Graphiques Cooperate



4.2 Graphique Erddap

Dataset Title: **Meteorologie locale site "climed" chaine de l'Etoile**
Institution: OSU Pytheas UAR 3470 CNRS (Dataset ID: meteo_climed_8bcc)
Range: longitude = 5.42 to 5.42°E, latitude = 43.37 to 43.37°N, time = 2021-11-15T16:00:00Z to 2022-06-20T05:45:00Z
Information: Summary | License | FGDC | ISO 19115 | Metadata | Background | Subset | Data Access Form | Files

Graph Type: linesAndMarkers
X Axis: time
Y Axis: Temperature_Air
Color:

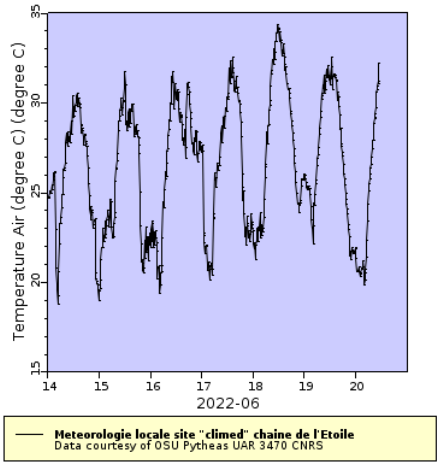
Time range: 7 day(s)

Constraints	Optional Constraint #1	Optional Constraint #2
time	>= 2022-06-14T00:00:00Z	<= 2022-06-21T00:00:00Z

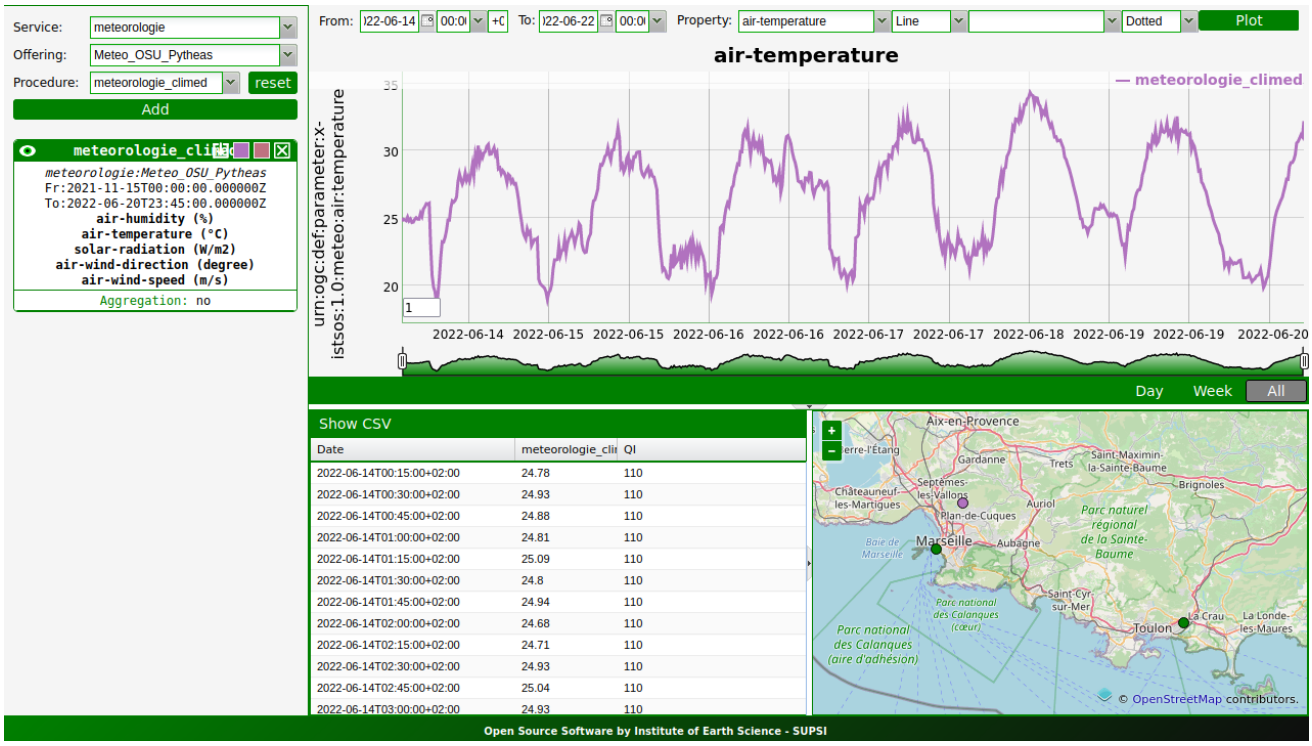
Server-side Functions
distinct()

Graph Settings
Marker Type: Dot Size: 5
Color:
Color Bar: Continuity: Scale:
Minimum: Maximum: N Sections:
Y Axis Minimum: Maximum: Ascending

Redraw the Graph (Please be patient. It may take a while to get the data.)



4.3 Graphique IstSOS



5 Programme de conversion CSV vers NetCDF :

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-    ## important d'avoir ca en 2eme ligne pour les accents é et apostrphophe
#
#  csv2NetCDF.py
#
#  Copyright Aout 2021 Maurice Libes <maurice.libes@osupytheas.fr> & Emilien Bonneton
#
#  Objet: convertit tout type de fichiers CSV en fichier NetCDF
#
#
#  This program is free software; you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation; either version 2 of the License, or
#  (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
#  MA 02110-1301, USA.
#
#
import os
import sys
import glob
import re
try:
    import netCDF4 as nc
    import pandas as pd
    import numpy as np
    import datetime
    import argparse
    from termcolor import  cprint
except:
    print("=> Erreur : il manque un de ces module python : netCDF4 | pandas | argparse | numpy | termcolor")
    print("$ pip3 install netCDF4 pandas argparse numpy termcolor")
    sys.exit(0)

from datetime import datetime
```

```
####### Verifie si tous les attributs de l'entete du fichier sont presents
def Control_Attributes_Var(indexdrop):
    #print("index du fichier ",indexdrop)

    for att in liste_att_obligatoires:
        if att not in indexdrop:
            cprint("Control failed!\nIl manque l\'attribut \'"+att+"\' dans le ficher : "+fic, color='red', attrs=['bold'])
            sys.exit(0)

    cprint('Control Attributs variable OK', color='green', attrs=['bold'])
    return 1


####### Verifie qu'il n'y a pas de caracteres interdits qui vont poser problemes dans le noms des colonnes
def Control_Illegal_char(cols):
    Illegal_char=['#','|',',',"'","''","/"," "]
    for elements in Illegal_char:
        for columns in cols:
            if(elements in columns):
                cprint('** Caractère interdit trouvé dans la colonne du nom:'+columns, color='red', attrs=['bold'])
                sys.exit(0)

    cprint("Control Illegal char OK", color='green', attrs=['bold'])
    return 0

####### Controle si il y a au moins les columns
#######     - timeseries/profile : date|latitude|longitude
#######     - grid : latitude|longitude
####### existe bien dans le fichier d'entree csv
def Control_Header_Columns(cols):
    #print("COLS ",cols)
    for i in range(0, len(cols)):  #on enleve les espace en debut et fin de chaine
        cols[i]=cols[i].lstrip()
        cols[i]=cols[i].rstrip()

        #cols[i]=cols[i].capitalize()
        #print("cols",i,"."+cols[i]+".")

    if (data_type =='grid'):
        colonnes_obligatoires = colonnes_obligatoires_grid

        for col in colonnes_obligatoires:
            if (col not in cols):
                cprint(cols,color="yellow",attrs=['bold'])
                cprint("Control failed!\nIl manque une des colonnes "+str(colonnes_obligatoires)+" dans le fichier :"+fic, color='red', attrs=['bold'])
                sys.exit(0)
```

```
    cprint("Control Columns OK", color='green', attrs=['bold'])
    return 1

#######
def Control_Global_Attributes():
    #Test s'il existe bien dans notre ficher csv metadata les 4 colonnes importante featuretype|conventions...
    indexlist = glob_att.index
    #print("feature type ",featuretype)
    if ( ('featuretype' in indexlist) and ('cdm_data_type' in indexlist)  and ('Conventions' in indexlist) ):
        cprint("Control Global Attributes OK", color='green', attrs=['bold'])
        if (featuretype=="timeseries"):
            if ('cdm_timeseries_variables' not in indexlist):
                cprint("il manque cdm_timeseries_variables", color='red', attrs=['bold'])
                sys.exit(0)
        elif (featuretype=="profile"):
            if ('cdm_profile_variables' not in indexlist):
                cprint("il manque cdm_profile_variables", color='red', attrs=['bold'])
                sys.exit(0)
        elif (featuretype=="grid"):
            if ('cdm_grid_variables' not in indexlist):
                cprint("il manque cdm_grid_variables", color='red', attrs=['bold'])
                sys.exit(0)
    else:
        cprint("Control failed! \nIl manque : featuretype | cdm_data_type | cdm_timeserie_variables | Conventions dans le fichier:"+fileMetaData, color='red', attrs=['bold'])
        cprint(" Conventions avec un C majuscule dans le fichier:"+fileMetaData, color='red', attrs=['bold'])
        sys.exit(0)
    return 0

####### Test si le data_type est Grid (une photo satellite) ou timeserie/profile
def Control_Datatype(fileMetaData):
    global data_type
    with open(fileMetaData,'r') as f:
        data=f.read()
        #print(data)
        if 'featuretype=grid' in data:
            data_type='grid'
        elif 'featuretype=timeseries' in data:
            data_type='timeseries'
        elif 'featuretype=profile' in data:
            data_type='profile'
        else :
            cprint('Veuillez renseigner le type de donnee (cdm_data_type=grid|timeseries|profile) en minuscule et sans espace',color='red',attrs=['bold'])
            sys.exit(0)
    cprint('Le type de donnee est '+data_type,color='yellow',attrs=['bold'])
    return 0
```

```
####### On recupere le nombre de dimension(s) dans le fichier (dans une var global)
def Control_Dimensions():
    global dim_fic

    ligne_dim = Index_selector('dimension')
    #print(dim_index,type(dim_index))
    #print(ligne_dim.values)
    if 'D1' in ligne_dim.values :
        dim_fic='1D'
    else :
        cprint('Veuillez renseigner la dimension D1 dans le fichier source',color='red',attrs=['bold'])
        sys.exit(0)

    if ('D2' in ligne_dim.values):
        dim_fic='2D'

    if ('D3' in ligne_dim.values) :
        cprint('Le programme ne supporte pas le 3D',color='red',attrs=['bold'])
        sys.exit(0)
    cprint('Le type de fichier est '+dim_fic,color='yellow',attrs=['bold'])

    if (data_type == 'grid'):
        if not dim_fic=='2D':
            cprint('Le format grid est uniquement possible en 2D, verifiez les dimensions !',color='red',attrs=['bold'])
            sys.exit(0)

        #print(dim_index.values)
        index_dim1 = list(ligne_dim.values).index('D1')
        #print('index = ',index_dim1)

        index_dim2 = list(ligne_dim.values).index('D2')
        #print('index = ',index_dim2)

        name_index = Index_selector('long_name')
        #print(name_index.values[index_dim1])
        name_dim1 = str(name_index.values[index_dim1])
        name_dim2 = str(name_index.values[index_dim2])

        if not ((name_dim1 == 'latitude') and (name_dim2 == 'longitude')) :
            cprint("Dans le cas d'un type Grid il faut que D1=Lat et D2=Lon",color='red',attrs=['bold'])
            sys.exit(0)
    return dim_fic
```

```
####### On controle si les valeurs de la colonne des dimensions sont monotones = strictement croissante uo decroissante
def Control_Monotonic():
    dim_index = Index_selector('dimension')
    for key,value in dim_index.items():   #pandas series can be treated like dictionnaries
        if (value =='D1') or (value == 'D2'):
            #print(csvdata[key].is_monotonic_decreasing)
            #print(csvdata[key].is_monotonic_increasing)
            #print('type val dim = ',type(csvdata[key]))
            #print('mono ? ',csvdata[key].is_monotonic)
            if (key == libelle_Datetime):
                col_value = Convert_Datenum()
                col_value = pd.Series(col_value).unique()
            else :
                col_value = pd.Series(csvdata[key]).unique()
            #print(col_value,type(pd.Series(col_value)))
            col_value=pd.Series(col_value)
            if not ((col_value.is_monotonic_increasing) or (col_value.is_monotonic_decreasing)):
                cprint('Les valeurs de la dimension '+key+' ne sont pas monotones',color='red',attrs=['bold'])
                sys.exit(0)
    return 1

####### On genere un fichier de sortie NetCDF vierge
def Generate_NetCDF(csvfile):

    netCDFFileName = csvfile.replace('.csv', '.nc')

    cprint("Generating NetCDF File : " +netCDFFileName, color='blue', attrs=['bold'])
    ncFile = nc.Dataset(netCDFFileName, 'w', format='NETCDF4')
    ncFile.set_fill_on()

    return ncFile

####### Permet de selectionner une ligne du fichier csv par son nom d'index
def Index_selector(index):
    #Selectionne une ligne du fichier csv par son nom d'index
    if(index == 'unites'):
        ligne = csvheader.loc[index].fillna("1") #mets 1 à la place du vide
        #print('ligne',ligne)
    elif(index == 'standard_name'):
        ligne = csvheader.loc[index]
    elif(index == 'long_name'):
        ligne = csvheader.loc[index]
    elif(index == 'dimension'):
        ligne = csvheader.loc[index]
    elif(index == 'variable'):
        ligne = csvheader.loc[index]
    elif(index == 'dimension'):
        ligne = csvheader.loc[index]
    elif (index == 'variable'):
        ligne = csvheader.loc[index]
    else:
        cprint("la 1ere colonne doit contenir les mots cles "+str(liste_att_obligatoires),'red', attrs=['bold'])
        sys.exit(0)

        print("ligne ",ligne)
    return ligne

####### On recupere les dimensions, leurs valeurs, leurns noms dans le fichier source pour les fic 1D
####### Et on stocke ces associations dans un tableau assiociatif
def Get_Dimension1D(fic):
    global tabasso_D1
    global list_dimension

    tab_dim={}
    list_dimension=[]

    tab_dim['SD']=("","1") #je rajoute les 'dimensions' de SD pour que la declaration de variable se passe bien

    dim_index = Index_selector('dimension')
    for key,value in dim_index.items():   #pandas series can be treated like dictionnaries
        if (value =='D1'):
            finddim=True
            val_dim=len(csvdata[key])
            tab_dim[value]=key,val_dim
            cprint("Dimensions pour le fichier NetCDF : "+key+' (de dimensions '+str(val_dim)+')',color='green',attrs=['bold'] )
            list_dimension.append(key)
        ####### Partie qui remplie le tableau associatifs avec les caracteristiques des dimensions
        if (value == 'D1'):
            tabasso_D1={}
            tab_uniqueD1=csvdata[key]      #tableau des valeurs uniques d'une dimension D1
            ind=0
            for v in tab_uniqueD1:
                tabasso_D1[v] = ind      #tableau pour associer les indices aux valeurs d'une dimension
                ind+=1
            #print('tab associatif D1 : ',tabasso_D1)

    return tab_dim

####### On recupere les dimensions, leurs valeurs, leurns noms dans le fichier source pour les fic 2D
####### Et on stocke ces associations dans un tableau assiociatif
def Get_Dimension2D(fic):
    global tabasso_D1, tabasso_D2
```

```
def Get_Dimension2D(fic):
    global  tabasso_D1, tabasso_D2
    global list_dimension
    # cherche les colonnes dans le fichier CSV qui contiennent D1,D2 à la ligne dimension, pour savoir quelle est la dimension
    # des variables du fichier NetCDF

    tab_dim={}
    list_dimension=[]

    if (data_type == 'grid'):
        unique = False
    else :
        unique=True


    tab_dim['SD']=("","1") #je rajoute les 'dimensions' de SD pour que la declaration de variable se passe bien

    dim_index = Index_selector('dimension')
    for key,value in dim_index.items():   #pandas series can be treated like dictionnaries
        if ( (value =='D1') or  (value == 'D2') ):
            finddim=True
            list_dimension.append(key)
            if not unique : #cas ou c'est un type grid
                val_dim=len(csvdata[key])
                print("On fait pas le unique pcq c'est grid")
            else : #cas ou c'est un autre type que grid
                val_dim=len(csvdata[key].unique())
                print("On fait le unique pcq c'est du 2D et pas grid")
            #print(csvdata[key].unique(),val_dim)
            tab_dim[value]=key,val_dim
            cprint("Dimensions pour le fichier NetCDF : "+key+' (de dimensions '+str(val_dim)+')',color='green',attrs=['bold'] )

        ####### Partie qui remplie le tableau associatifs avec les caracteristiques des dimensions
        if (value == 'D1'):
            tabasso_D1={}
            if unique :
                tab_uniqueD1=np.sort(csvdata[key].unique())
                #tab_uniqueD1=np.sort(tab_uniqueD1)
            else :
                tab_uniqueD1=csvdata[key]
            ind=0
            for v in tab_uniqueD1:
                tabasso_D1[v] = ind      #tableau pour associer les indices aux valeurs d'une dimension
                ind+=1
            #print('tab associatif D1 : ',tabasso_D1)
        if (value == 'D2'):
            tabasso_D2={}
            if unique :
                tab_uniqueD2=np.sort(csvdata[key].unique())
                #tab_uniqueD2=np.sort(tab_uniqueD2)
            else :
                tab_uniqueD2=csvdata[key]
            ind=0
            for v in tab_uniqueD2:
                tabasso_D2[v] = ind
                ind+=1
            #print('tab associatif D2 : ',tabasso_D2)
        ########

    if (not finddim):
        cprint("Dimensions absente dans le fichier CSV "+fic,color='red',attrs=['bold'] )
        sys.exit(0)

    #cprint("Dimensions pour le fichier NetCDF :",color='green',attrs=['bold'] )
    #print(tab_dim)
    return tab_dim

####### On recupere le nom des colonnes et on associe a quoi ces variables vont dependre dans un tab asso
def Get_variable(fic):
    dict_variable={}
    var_index = Index_selector('variable')  #on prend la ligne qui commence par "variable"
    for key,value in var_index.items():   #pandas series can be treated like dictionnaries
        if (value != ''):
            try :
                value_list=value.split(',')  #pour separer D1,D2
            except AttributeError :
                cprint('Variables absentes',color='red',attrs=['bold'])
                sys.exit(0)
            #print('DIC_FONCTION : key : ',key,' value : ',value, value_list)
            dict_variable[str(key)]=value_list
            cprint("Variables pour le fichier NetCDF : "+key,color='green',attrs=['bold'] )
        else :
            cprint("Variable fonction dans le fichier CSV non renseignee"+fic,color='red',attrs=['bold'] )
            sys.exit(0)
    #print(dict_variable)
    return dict_variable

####### On cree les dimensions du fichier NetCDF (ainsi que la dimension len(stationname))
def Create_Dimensions(dimension):
    for D in dimension:
        if (D != 'SD') :
            dim,val = dimension[D]
```

```
            dim,val = dimension[D]
            #print('dimension: ',D, 'dim ', dim, "val: " , val)
            ncFile.createDimension(dim, val)
            cprint('Dimension du fichier NetCDF : '+dim+' créée de taille '+str(val), color='green', attrs=['bold'])
        #print(stationname, len(stationname))


    ncFile.createDimension('lenstation', len(stationname))
    cprint('Dimension du fichier NetCDF : len(stationname)', color='green', attrs=['bold'])

    #print('Dimension : ',ncFile.dimensions)
    return 0

####### Creation des global attributs/metadata du fichier NC lues dans un fichier CSV externe
def Create_Global_Attributes(ncFile):
    indexlist = glob_att.index
    for index in indexlist:
        value = glob_att.loc[index].values[0]
        if index == 'Conventions':
            ncFile.setncattr(index, value)
        else:
            ncFile.setncattr(index.casefold(), value)
    cprint('Creation Global Attributs OK', color='green', attrs=['bold'])
    return 0

####### Creation de la variable station name (cas particulier|fausse dimension)
def Create_Variable_Station():
    station_name = ncFile.createVariable('station_name', 'S1', 'lenstation')
    station_name.long_name = "station_name"
    station_name.cf_role = featuretype+"_id"
    station_name[:] = nc.stringtoarr(stationname,len(stationname))

    return 0

####### Creation d'un tab a partir de deux tableau pour en creer un contenant tout les infos des variables
####### (ex : tabcreavar['Temperature'] = ['Datetime', 94, 'Profondeur', 10, 2] = [dim1, val_dim1, dim2, val_dim2, nb_var])
def Format_creavar(tab_dim,tab_var):
    tab_creavar={}
    #tab_creavar['Exemple']=['Variable','Dimension de la variable','Derniere case : Format pour la creation de la variable']  exemple 'Temperature': ['Datetime', 9, 'Profondeur', 8, 2],
    for col in tab_var:
        #print("la variable ",col)
        #print(tab_var[col])
        #print(tab_dim[str(tab_var[col])])
        tab_creavar[col]=[]
        liste=tab_creavar[col]
        #print('long liste var ==',len(tab_var[col]))
```

```
        if tab_var[col][0] == 'SD':
            nb_var=0
        else :
            nb_var=len(tab_var[col])

        for i in range (0,len(tab_var[col])):
            #print("a pour variable ",tab_var[col][i])
            #print(tab_dim[tab_var[col][i]])
            dim,val=tab_dim[tab_var[col][i]]
            #print("ce qui corrspond a ",dim, "qui a pour dim ",val)

            liste.append(dim)
            liste.append(val)

            if i == (len(tab_var[col])-1) :
                liste.append(nb_var)    #on stocke le nb de variable dont depend la colonne pcq la commande ncFile.createvariable ne peut pas prendre de str

    #print('tab creavar == ',tab_creavar)
    return tab_creavar

####### Retourne le format (type) de chaque valeur des colonnes du fichier CSV
####### Dans une liste du type ['f4', 'f4', 'i4' ...]
def Get_Formats():
    liste_formats = []  #on recoit un tableau numpy a ce niveau, les types numpy sont np.float, np.int

    ligne=csvdata.iloc[1]# on prend une ligne de données sous forme de série pandas
    l=ligne.tolist()            # on la converti en tableau numpy avec la fonction tolist()
    i=0
    for value in ligne:
        if isinstance(value,str):
            lmax=max(csvdata.iloc[:,i].apply(len))  ## on calcule la longueur de chaine maximale dans la colonne "i"
            #print("indice de colonne String ",i," longueur de chaine max = ",lmax)
            frmt = 'S'+str(lmax)  # format S de la longueur xx max de la colonne 'Sxx'
            #print("value = ",value,type(value))
        elif isinstance(value,np.floating) or isinstance(value, float):   # np.floating renvoie True si on est en float64 ou float32
            frmt = 'f4'
            #print("value = ",value,type(value))
        elif isinstance(value,np.int64):
            frmt = 'i4'
        i+=1
        liste_formats.append(frmt)
        #print("value = ",value,type(value))

    #print('sortie liste_formats ', liste_formats)
    return liste_formats
```

```
451  ####### Permet de convertir une date ISO en date numerique
452  def Convert_Datenum():
453      dateHeure = []
454
455      for dateiso in csvdata[libelle_Datetime]:
456          #print("dateiso = ",dateiso)   # on cherche ce format 2018-07-07T10:08:59Z
457          pattern=re.compile('(\d{4})-(\d{2})-(\d{2})T(\d{2}):(\d{2}):(\d{2})Z?')
458          try:
459              match=pattern.search(dateiso)
460              chainedate=match.groups()
461              annee,mois,jour,heure,minut,sec=chainedate
462          except:
463              cprint("Erreur date ISO "+dateiso,'red',attrs=['bold'])
464              sys.exit(0)
465
466          #print("annee,mois,jour,heure,minut,sec ", int(annee), int(mois), int(jour), int(heure), int(minut), int(sec))
467
468          dateobs = datetime(int(annee), int(mois), int(jour), int(heure), int(minut), int(sec))
469          datenum = nc.date2num(dateobs, units=NC_TIME_FMT)
470          dateHeure.append(datenum)
471
472      #print("tableau dateHeure ",dateHeure)
473
474      return dateHeure
475
476  ####### Compare les fichiers .csv qui sont dans le repértoire CsvFiles : ils doivent etre de meme nature
477  def Compare_Files(cols,firstcols):
478
479      if (len(cols) != len(firstcols)):
480          print("\n==> nombre de colonnes différents ",len(cols),len(firstcols) )
481          cprint("Les fichiers CSV dans le repertoire CsvFiles sont differents !!", color='red',attrs=['bold'])
482          cprint("Le programme ne traite que un ensemble de fichiers identiques !!", color='red',attrs=['bold'])
483          sys.exit(0)
484
485      if (not np.array_equal(cols,firstcols) ):
486          for i in range(0,len(firstcols) ):
487              if (cols[i] != firstcols[i]):
488                  print("\n==> ",cols[i],firstcols[i])
489                  cprint("Les fichiers CSV dans le repertoire CsvFiles sont differents !!", color='red',attrs=['bold'])
490                  cprint("Le programme ne traite que un ensemble de fichiers identiques !!", color='red',attrs=['bold'])
491          sys.exit(0)
492
493  ####### Creation des variables et ecriture des datas 1D dans le fichier NetCDF
496  def Create_Write_Variables1D(tab_creavar):
497      #Creation des variables NetCDF et remplissage de leurs tableaux automatiquement sauf pour le cas particulier 'Date'
498
499      units = Index_selector('unites')
500      standard_name = Index_selector('standard_name')
501      long_name = Index_selector('long_name')
502
503      # déterminer le type des variables (int, float, str...)
504      frmt=Get_Formats()
505      #print("Formats des colonnes : ", frmt)
506
507      # convertit les dates ISO 2011-01-16T09:38:00 en format numérique pour NetCDF
508      try :
509          dateHeure=Convert_Datenum()
510      except :
511          cprint("Il n'y a pas de colonne DateTime dans le fichier",color='red',attrs=['bold'])
512      #print("sortie dateHeure ", dateHeure)
513
514      i = 0 # index de colonnes
515      for col in cols:
516          cprint("\t Traitement col -"+col+'-', 'yellow', attrs=['bold'])
517
518          if (col == libelle_Datetime):
519              frmt[i]='i4'   #on force la date en Int car on l'a convertie en Int avec la fonction Convert_Datenum()
520              units[i] = NC_TIME_FMT
521
522          ## creation des variables NetCDf avec le nom des colonnes du fichier CSV
523          colnc=col.strip()
524          #print("dim ",dim)
525
526          ## creation des variables NetCDf avec le nom des colonnes du fichier CSV
527          colnc=col.strip()
528          nb_variable=tab_creavar[colnc][-1]
529          #print('nb var dim == ',nb_variable)
530
531          #on cree la variable NetCDF et on l'associe a un tableau tabcol
532          if nb_variable == 0:
533              tabcol = ncFile.createVariable(colnc, frmt[i], fill_value=fill_value)
534          elif nb_variable == 1:
535              var = tab_creavar[colnc][0]
536              #print('var de dimension1 == ',var)
537              tabcol = ncFile.createVariable(colnc, frmt[i], var, fill_value=fill_value)
538
539          ## rajout de certains attributs Netcdf en fonction des variables
540          ## Attention : si on rajoute des atributs "toto" dans le header du fichier CSV, il  FAUDRA modifier le code ici
541          ## tabcol.toto = toto[i]
```

```
        ## Attention : si on rajoute des atributs "toto" dans le header du fichier CSV, il  FAUDRA modifier le code ici
        ## tabcol.toto = toto[i]

        # attributs standards , présents dans le header du fichier CSV, pour toutes les variables
        tabcol.units = units[i]
        tabcol.long_name = long_name[i]
        tabcol.standard_name = standard_name[i]


        tabcol = Adapte_Attributs(col,tabcol)
        #cas particuliers si on veut ajouter des attributs specifiques a certaines variables

    ## cas des valeurs en chaines de caracteres : elles sont traitées différement pour Netcdf
        if ( frmt[i][0] =='S'):
            #value=csvdata[col]
            #print("valeur ",value.values[0])
            #print("** longueur  tableau: ",len(value),"long chaine ",l)
            tabcol._Encoding = 'ascii'
            #print("** S1 col = ",col)


  ## ecriture des colonnes de valeurs CSV dans les tableaux tabcol[] des variables de Netcdf
        if (col==libelle_Datetime):
            if nb_variable == 0 :
                tabcol[:] = np.unique(dateHeure)    # cas particulier ecriture des valeurs de temps dans la variable NetCDF
            elif nb_variable == 1 :
                tabcol[:] = dateHeure
            tabcol.origin = NC_TIME_ORIGIN
        else:
            #print("*** ",csvdata[col].values)
            if ( frmt[i][0] =='S'): #cas ou c'est des str
                value = np.array(csvdata[col].values,dtype=frmt[i])
                if nb_variable == 0 :
                    tabcol[:] = np.unique(value)
                elif nb_variable == 1 :
                    tabcol[:] = value
            else: #cas ou c'est des int ou float
                value = csvdata[col].tolist()
                if nb_variable == 0 :
                    tabcol[:] = np.unique(value)    # on ecrit les valeurs des colonnes du fichier CSV dans le tableau de la  variable NetCDF
                elif nb_variable == 1 :
                    tabcol[:] = value


        cprint('Variable '+col+' ecrit avec succes',color='blue',attrs=['bold'])
        i += 1
        i += 1

    cprint('Fin: Ecriture NetCDF 1D OK', color='green', attrs=['bold'])

    return 1

####### Initialisation d'une matrice remplie de Nan avec les dimensions de la variable
def Init_Matrice(dim2D,tab_creavar):
    #print('dim2D = ',dim2D)
    D1,D2 = dim2D
    val_D1 = tab_creavar[D1][1] #on recupere les valeurs des dimensions D1 et D2
    val_D2 = tab_creavar[D2][1]
    #print(val_D1,val_D2)
    array = np.empty((val_D1,val_D2))
    array.fill(fill_value)
    #print(array)
    return array,D1,D2

####### Placement des datas au bon endroit dans la matrice en fct des valeurs des dimensions sur sa ligne dans le fichier source
def Rangement_Data(matriceNan,col,D1,D2):
    matrice = matriceNan
    (nb_ligne,nb_col) = csvdata.shape
    #print(nb_ligne,nb_col,D1,D2)
    #print(type(csvdata))
    #print(csvdata)
    #print(csvdata.loc[1,'Datetime'])
    for index,row in csvdata.iterrows():                                ###explication cas de l'etang de berre
        val_ligne_D1 = row[str(D1)]                  #pour chaque ligne on recupere la date avec sa profondeur associé
        val_ligne_D2 = row[str(D2)]
        val_ligne_var = row[str(col)]
        #print(val_ligne_D1, val_ligne_D2, val_ligne_var)

        index_val_D1 = tabasso_D1[val_ligne_D1]     #on recupere l'index associe a la valeur de D1
        index_val_D2 = tabasso_D2[val_ligne_D2]     #pareil pour D2
        #print(index_val_D1,val_ligne_D1 )

        try :
            matrice[index_val_D1,index_val_D2] = val_ligne_var
        except ValueError :
            cprint('Il y a une valeur erronnée dans la colonne '+col+ '\n Valeur = '+str(val_ligne_var),color='red',attrs=['bold'])
            sys.exit(0)
    #print('matrice de ',col,matrice)
    return matrice

```

```
629 ###### Placement des dates dans la matrice (pas en fct de sa ligne comme Rangement_Data car on pioche dans la liste dateHeure)
630 def Rangement_Dateheure(dateHeure,matriceNan,D1,D2):
631     matrice = matriceNan
632     (nb_ligne,nb_col) = csvdata.shape
633     #print(nb_ligne,nb_col,D1,D2)                    #on recupere la taille de la matrice
634     i=0                                             #explication cas viper
635     for ligne in range(0,nb_ligne-1):                    #on fait une double boucle qui parcourt toutes les cases de la matrice
636         for colonne in range(0,nb_col-1):
637             if (i < len(dateHeure)):                #tant qu'on a des dates on reste dans la boucle
638                 #print(' i = ',i)
639                 val_ligne_var = dateHeure[i]
640                 #print('date = ', val_ligne_var)
641                 matrice[ligne,colonne] = val_ligne_var
642                 i+=1
643     return matrice
644  ##
645 def Adapte_Attributs(col,tabcol):
646     #a faire
647 
648     #cas particuliers si on veut ajouter des attributs specifiques a certaines variables
649     if (col=="Latitude"):
650         tabcol.axis="Y"
651         lat_min=csvdata[col].min() # on prend les valeurs min/max de la colonne Latitude
652         lat_max=csvdata[col].max()
653         tabcol.coverage_content_type = "coordinate"
654         tabcol.units = "degrees_north"
655         tabcol.coordsys = "geographic";
656         tabcol.valid_range = (-90.0, 90.0)
657         tabcol.point_spacing = "even";
658         tabcol.coordsys = "geographic";
659         ncFile.setncattr("geospatial_lat_min" , lat_min)
660         ncFile.setncattr("geospatial_lat_max" , lat_max)
661     if (col=="Longitude"):
662         tabcol.axis="X"
663         lon_min=csvdata[col].min()  # on prend les valeurs min/max de la colonne Longitude
664         lon_max=csvdata[col].max()
665         tabcol.coverage_content_type = "coordinate"
666         tabcol.units = "degrees_east"
667         #tabcol.valid_range = (0.0, 360.0)
668         tabcol.valid_range = (-180.0, 180.0)
669         tabcol.point_spacing = "even";
670         tabcol.coordsys = "geographic";
671         ncFile.setncattr("geospatial_lon_min" , lon_min)
672         ncFile.setncattr("geospatial_lon_max" , lon_max)
673     if (col == 'Altitude'):
674         tabcol.axis="Z"
675         tabcol.coverage_content_type = "coordinate"
676         tabcol.units = "altitude"
677         #tabcol.valid_range = (0.0, 360.0)
678         tabcol.valid_range = (0,)
679         tabcol.point_spacing = "even";
680         tabcol.coordsys = "geographic";
681         alt_min=csvdata[col].min()  # on prend les valeurs min/max de la colonne Longitude
682         alt_max=csvdata[col].max()
683         ncFile.setncattr("geospatial_alt_min" , alt_min)
684         ncFile.setncattr("geospatial_alt_max" , alt_max)
685     if (col=="Datetime"):
686         tabcol.axis="T"
687         tabcol.calendar = "Gregorian"
688         tabcol.coverage_content_type = "coordinate"
689     if (col=="Depth" or col=="Profondeur"):
690         tabcol.axis="Z"
691 
692     return tabcol
693 
694  ###### Creation des variables et ecriture des datas 2D dans le fichier NetCDF
695 def Create_Write_Variables2D(tab_creavar):
696     #Creation des variables NetCDF et remplissage de leurs tableaux automatiquement sauf pour le cas particulier 'Date'
697 
698     units = Index_selector('unites')
699     standard_name = Index_selector('standard_name')
700     long_name = Index_selector('long_name')
701 
702     # déterminer le type des variables (int, float, str...)
703     frmt=Get_Formats()
704     #print("Formats des colonnes : ", frmt)
705 
706     # convertit les dates ISO 2011-01-16T09:38:00 en format numérique pour NetCDF
707     #if (not data_type=='grid'):
708     try :
709         dateHeure=Convert_Datenum()
710     except :
711         cprint("Il n'y a pas de colonne DateTime dans le fichier",color='red',attrs=['bold'])
712     #print("sortie dateHeure ", dateHeure)
713 
714     i = 0 # index de colonnes
715     for col in cols:
716 
717         cprint("\t Traitement col -"+col+'-', 'yellow', attrs=['bold'])
718 
719         if (col == libelle_Datetime):
```

```
        if (col == libelle_Datetime):
            frmt[i]='i4'   #on force la date en Int car on l'a convertie en Int avec la fonction Convert_Datenum()
            units[i] = NC_TIME_FMT

        ## creation des variables NetCDf avec le nom des colonnes du fichier CSV
        colnc=col.strip()
        nb_variable=tab_creavar[colnc][-1]  #recupere le nb de variable de dimension dont dépend la colonne
        #print('nb var dim == ',nb_variable)

        #on cree la variable NetCDF et on l'associe a un tableau tabcol
        if nb_variable == 0:
            tabcol = ncFile.createVariable(colnc, frmt[i], fill_value=fill_value)
            cprint('Creation de la var '+col+' de dimension SD -> OK', 'blue', attrs=['bold'])
        elif nb_variable == 1:
            var = tab_creavar[colnc][0]
            #print('var de dimension1 == ',var)
            tabcol = ncFile.createVariable(colnc, frmt[i], var, fill_value=fill_value)
            cprint('Creation de la var '+col+' de dimension '+var+' -> OK', 'blue', attrs=['bold'])
        elif nb_variable == 2:
            var1 = tab_creavar[colnc][0]
            var2 = tab_creavar[colnc][2]
            #print('var de dimension2 == ',var1, var2)
            dim2D=(var1,var2)  #en 2D il faut un tuple
            tabcol = ncFile.createVariable(colnc, frmt[i], dim2D, fill_value=fill_value)
            cprint('Creation de la var '+col+' de dimension '+str(dim2D)+' -> OK', 'blue', attrs=['bold'])

        '''elif nb_variable == 3:
            var1 = tab_creavar[colnc][0]
            var2 = tab_creavar[colnc][2]
            var3 = tab_creavar[colnc][4]
            dim3D=(var1,var2,var3)  #en 3D il faut un tuple
            #print('var de dimension3 == ',var1,var2,var3)
            tabcol = ncFile.createVariable(colnc, frmt[i], dim3D)'''#pour le 3D


        ## rajout de certains attributs Netcdf en fonction des variables
        ## Attention : si on rajoute des atributs "toto" dans le header du fichier CSV, il  FAUDRA modifier le code ici
        ## tabcol.toto = toto[i]

        # attributs standards , présents dans le header du fichier CSV, pour toutes les variables
        tabcol.units = units[i]
        tabcol.long_name = long_name[i]
        tabcol.standard_name = standard_name[i]

        tabcol = Adapte_Attributs(col,tabcol)
        #cas particuliers si on veut ajouter des attributs specifiques a certaines variables
        #cas particuliers si on veut ajouter des attributs specifiques a certaines variables

    ## cas des valeurs en chaines de caracteres : elles sont traitées différement pour Netcdf
        if ( frmt[i][0] =='S'):
            value=csvdata[col]
            #print("valeur ",value.values[0])
            #print("** longueur  tableau: ",len(value),"long chaine ",len(value.values[0]))
            tabcol._Encoding = 'ascii'
            #print("** S1 col = ",col)

## ecriture des colonnes de valeurs CSV dans les tableaux tabcol[] des variables de Netcdf

        if (col==libelle_Datetime) :
            tabcol.origin = NC_TIME_ORIGIN
            arrayDH = np.array(dateHeure)
            if nb_variable == 0 : ## je pourrais mettre le 1D profile ici mais on va separer pour que ca soit bien clair
                print("taille du tableau SD de Datetime a ecrire: ",len(np.unique(arrayDH)))
                tabcol[:] = np.unique(arrayDH)
            elif nb_variable == 1 :
                if col in list_dimension : ## test : col in dimensions
                    print("taille du tableau 1D (var dimension) de Datetime a ecrire: ",len(np.unique(arrayDH)))
                    tabcol[:] = np.unique(arrayDH)
                else : ## si col not in dimensions
                    print("taille du tableau 1D (var non dimension) Datetime a ecrire: ",len(dateHeure))
                    tabcol[:] = dateHeure
            elif nb_variable == 2 :
                matriceNan,D1,D2 = Init_Matrice(dim2D,tab_creavar)          #je cree une matrice avec les bonnes dim et la remplie de Nan + je recupere le nom des colonnes dimensions
                matrice_range = Rangement_Dateheure(dateHeure,matriceNan,D1,D2)          #je range les datas a leur place qui est determine en fct des valeurs de D1 et D2 sur leur ligne
                print("Taille du tableau datetime 2D  a ecrire: ",matrice_range.shape)
                tabcol[:] = matrice_range
        else :
            if nb_variable == 0:  #SD
                value=csvdata[col].unique()
                print("taille du tableau SD a ecrire: ",len(value))
                tabcol[:] = value

            elif nb_variable == 1 :
                if col in list_dimension: ## si col in dimensions
                    value=list(csvdata[col].unique())
                    print("taille du tableau 1D (var dimension) a ecrire: ",len(value))#," value=", value)
                    #print("type ",type(value))
                    #print("tabcol ",tabcol)
                    tabcol[:] = np.array(value)
                else : ##si col not in dimensions
                    value=np.array(csvdata[col])
                    print("taille du tableau 1D (var non dimension) a ecrire: ",len(value))
                    tabcol[:] = value
                    value=np.array(csvdata[col])
                    print("taille du tableau 1D (var non dimension) a ecrire: ",len(value))
                    tabcol[:] = value

            elif nb_variable == 2 :
                #print("*** ",csvdata[col].values)
                #print("** S2 col = ",string_values)
                #tabcol[:] = string_values
                matriceNan,D1,D2 = Init_Matrice(dim2D,tab_creavar)          #je cree une matrice avec les bonnes dim et la remplie de Nan + je recupere le nom des colonnes dimensions
                matrice_range = Rangement_Data(matriceNan,col,D1,D2)        #je range les datas a leur place qui est determine en fct des valeurs de D1 et D2 sur leur ligne
                print("Taille du tableau 2D  a ecrire: ",matrice_range.shape)
                tabcol[:] = matrice_range

        cprint('Variable '+col+' et ses datas ecrit avec succes',color='blue',attrs=['bold'])
        i += 1

    print(' --------------------------------- ');print('| Variables créées + data ajoutées |');print(' --------------------------------- ')

    cprint('Fin: Ecriture NetCDF 2D OK', color='green', attrs=['bold'])

    return 1

#######------ Coeur du programme -----########
def main():
    global NC_TIME_FMT, NC_TIME_ORIGIN
    global csvheader, csvdata
    global stationname, lenstation, OutputPath
    global glob_att,featuretype
    global ncFile,fic
    global cols,indexlist,indexdrop,fill_value
    global libelle_Datetime, colonnes_obligatoires, liste_att_obligatoires, colonnes_obligatoires_grid


    NC_TIME_FMT = 'seconds since 1970-01-01 00:00:00 UTC'
    NC_TIME_ORIGIN = "01-JAN-1970 00:00:00"
    fill_value=-999.99

    # ========== Read CSV file =============

    InputPath="./CsvFiles/"
    OutputPath="./NcFiles/"
    fileMetaData = "global_attributes.csv"
    libelle_Datetime="Datetime"
    colonnes_obligatoires=['Datetime','Latitude','Longitude']
    colonnes_obligatoires_grid=['Latitude','Longitude']
    liste_att_obligatoires=['unites','dimension','standard_name','long_name','variable']
```

```
    parser = argparse.ArgumentParser(description='-s delimiter of the csv file ')
    parser.add_argument('-s', '--separator', help='Please enter a separator for you csvFile', required=True)
    parser.add_argument('-n', '--name', help='Please enter the name of the station', required=False)
    args = parser.parse_args()

    motif=InputPath+'*.csv'  ## pour tous les fichier seabird finissant par .csv
    ficcsv=glob.glob(motif)  ## pour lister fichiers d'un certain motif
    ficcsv=sorted(ficcsv)
    if (len(ficcsv) ==0):
        cprint("Aucun fichier CSV *.csv a traiter dans "+InputPath,color='red',attrs=['bold'])
        cprint("=> Placer les fichiers CSV *.csv a traiter dans le répertoire "+InputPath,color='yellow',attrs=['bold'])
        sys.exit()

    firstcols=[]; nbfic=0
    for fic in ficcsv:
        fic=os.path.basename(fic)
        if fic.endswith(".csv"):
            cprint("\n Traitement fichier "+fic, color='blue', attrs=['bold'])

            n=args.name
            if (n == None ):
                cprint("Donnez le nom de la station pour le fichier "+fic, color='yellow', attrs=['bold'])
                stationname=input()
            else:
                stationname = args.name
                lenstation = len(stationname)

            csvheader = pd.read_csv(InputPath+fic,  low_memory=False, sep=args.separator,index_col=0)
            #print(csvheader)

            nbfic+=1
            #print('test',csvheader)
            indexdrop=csvheader.index.dropna() #selectionne la 1eree colonne du fichier CSV en supprimant toute les valeurs vides (NAN)
            #print("Attributs des variables : ",indexdrop)

            ####### On lit les data en sautant les lignes de header
            csvdata=pd.read_csv(InputPath+fic, sep=args.separator,skiprows=range(1,len(indexdrop)+1),index_col=0)

            ######## on extrait le nom des colonnes du fichier CSV
            cols = csvdata.columns.values
            #print('cols',cols); print(type(cols))

            if (nbfic > 1): ## verifie que les entetes des fichiers CSV dans le repertoire CsvFiles soient les memes
                Compare_Files(cols,firstcols)
            else:
                firstcols = cols

            ####### Ouverture du fichier global_attributs
            glob_att = pd.read_csv(fileMetaData, delimiter='=', header=None, index_col=0)
            indexlist = glob_att.index
            featuretype = glob_att.loc['featuretype'].values[0]  # voir si on calcule le featuretype automatiquement?

            ####### On recupere le type de donnee (en var global) dans le fichier global_attributes.csv
            Control_Datatype(fileMetaData)

            ####### On recupere le nombre de dimensions dans le fichier (en var global)
            dim_fic=Control_Dimensions()

            ####### Controle de la proprete du fichier
            Control_Attributes_Var(indexdrop)
            Control_Illegal_char(cols)
            Control_Header_Columns(cols)
            Control_Global_Attributes()
            Control_Monotonic()
            print(' ---------------- ');print('| Fichier propre |');print(' ---------------- ')

            ####### On genere le fichier de sortie NetCDF
            ncFile = Generate_NetCDF(OutputPath+fic)
            ncFile.setncattr("date_created" , str(datetime.now()))


            ####### On recupere les dimensions et les variables pour les stocker dans un tableau associatif
            if (dim_fic == '1D') :      #cas 1D
                dimension = Get_Dimension1D(fic) #fct qui traite uniquement le 1D
            elif (dim_fic == '2D') :    #cas 2D
                dimension = Get_Dimension2D(fic) #fct qui traite uniquement le 2D
            print('|')
            variable = Get_variable(fic)         # fct qui determine quelle variable evolue en fonction de laquelle
            print(' ------------------------------- ');print('| Variable et Dimension récupéré |');print(' ------------------------------- ')

            ####### On cree les dimensions avec les bonnes valeurs selon le type de donnee
            Create_Dimensions(dimension)
            print(' --------------------- ');print('| Dimensions ajoutées |');print(' --------------------- ')

            ####### On cree les metadata et on les integre au fichier NetCDF
            Create_Global_Attributes(ncFile)
            print(' ------------------- ');print('| Metadata ajoutées |');print(' ------------------- ')

            ####### Creation de la variable station (d'apres le nom passé en argument -n)
            Create_Variable_Station()

            ####### Creation d'un tab contenant tout les infos des variables
            ####### Creation d'un tab contenant tout les infos des variables
            tab_creavar = Format_creavar(dimension,variable)
            #tab_dim_var = Create_Tab_Dim_Var()

            #print('tab_creavar : ',tab_creavar)

            ####### Creation des variables et ecriture des variables
            if (dim_fic == '1D'):
                Create_Write_Variables1D(tab_creavar)
            elif (dim_fic == '2D'):
                Create_Write_Variables2D(tab_creavar)

if __name__=='__main__':
    sys.exit(main())
```