Le code comprend différentes parties/étapes : d'abord, on introduit les fonctions mathématiques/graphiques qui vont permettre d'effectuer les calculs et d'afficher les courbes. Puis, on indique le chemin pour lire le fichier comportant les données. Ensuite, on définit chaque action (ex : action du calcul de l'humidex, action d'affichage des valeurs statistiques…). Enfin, on peut écrire le programme en lui-même.

LES DIFFÉRENTES FONCTIONS QUE L'ON VA UTILISER DANS NOTRE PROGRAMME

• ÉTAPE 1

```
from numpy import *
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import csv
import sys
```

-> Importation des fonctions mathématiques et des fonctions graphiques nécessaires. Importation du tableur.

• ÉTAPE 2

```
df=pd.read_csv('EIVP_KM.csv',sep=';')
```

-> Lecture du tableur.

• ÉTAPE 3

```
def display(nom_variable, date_deb, date_fin):
    if date_deb == '':
        date_deb = df['sent_at'].min()
    if date_fin == '':
        date_fin = df['sent_at'].max()
    df_filtre = df[(df['sent_at'] >= date_deb) & (df['sent_at'] <= date_fin)]
    df_filtre.plot(x="sent_at", y=nom_variable, color='blue')
    plt.show()
```

-> Affichage des courbes montrant l'évolution d'une variable (nom_variable) en fonction du temps (sent_at).
Possibilité de préciser un intervalle de temps (date_deb, date_fin) avec la commande filtre.

• ÉTAPE 4

```
def display_stats(nom_variable, date_deb, date_fin):
    if date_deb == '':
        date_deb = df['sent_at'].min()
    if date_fin == '':
        date_fin = df['sent_at'].max()
    df_filtre = df[(df['sent_at'] >= date_deb) & (df['sent_at'] <= date_fin)]

    data = df_filtre[nom_variable]
    moyenne = data.mean()
    ecart_type = data.std()
    variance = data.var()
    mediane = data.median()
    min = data.min()
    max = data.max()

    df_filtre.plot(x="sent_at", y=nom_variable, color='blue')
    plt.axhline(moyenne, color='r', label='moyenne')
    plt.axhline(min, color='g', label='min')
    plt.axhline(max, color='g', label='max')
    plt.axhline(mediane, color='y', label='mediane')
    labels = ["moyenne", "min", "max", "mediane"]
    handles, _ = ax.get_legend_handles_labels()
    plt.text(1,50, 'ecart type %f' % ecart_type)
    plt.text(1,60, 'variance %f' % variance)
    plt.legend(handles=handles[1:], labels=labels)
    titre = "Statistiques %s" % variable
    plt.title(titre)
    plt.show()
```

-> Affichage des valeurs statistiques de la variable indiquée.

- ÉTAPE 5

```
def calcul_humidex(temp, humidity):
    exp = 7.5 * (temp / 237.7 + temp)
    h = temp + 5/9 * 6.112 * pow(10, exp) * humidity
    return temp
```

-> Définition de l'humidex via sa formule mathématique faisant intervenir les deux paramètres de la température et de l'humidité.

```
def display_humidex(date_deb, date_fin):
```

```
    if date_deb == '':
        date_deb = df['sent_at'].min()
    if date_fin == '':
        date_fin = df['sent_at'].max()
    df_filtre = df[(df['sent_at'] >= date_deb) & (df['sent_at'] <= date_fin)]
         df_humidex = df_filtre[['temp', 'humidity']].apply(lambda x:
calcul_humidex(x[0],x[1]))
    df_humidex.plot(x="sent_at", y='humidex', color='blue')
    plt.show()
```

-> Calcul de l'humidex, avec possibilité d'indiquer un intervalle de temps (date_deb, date_fin).

- ÉTAPE 6

```
def display_correlation(variable1, variable2, date_deb, date_fin):
    if date_deb == '':
        date_deb = df['sent_at'].min()
    if date_fin == '':
        date_fin = df['sent_at'].max()
    df_filtre = df[(df['sent_at'] >= date_deb) & (df['sent_at'] <= date_fin)]

    corr = df_filtre[variable1].corr(df_filtre[variable2])
    print("correlation entre %s et %s = %f" % (variable1, variable2, corr))
```

-> Calcul de l'indice de corrélation entre deux variables

```
    ax = plt.gca()
    df.plot(kind='line', x='sent_at', y=variable1, color='blue', ax=ax)
    df.plot(kind='line', x='sent_at', y=variable2, color='red', ax=ax)
    plt.show()
```

-> Affichage de l'indice de corrélation sur les courbes

LE PROGRAMME EN LUI-MÊME

```python
if __name__ == "__main__":
    try:
        print("tableau des arguments:%s" % sys.argv)
        # verification du nombre d'arguments en ligne de commande
        if len(sys.argv) < 3:
            print("Nb arguments insuffisants")
            sys.exit(0)

        action = sys.argv[1]
        if action in ['display', 'displayStats']:
            # récuperation des autres arguments
            variable = sys.argv[2]
            if variable != 'humidex' and variable not in df.columns:
                print('variable non connue')
                sys.exit(0)

            # dates optionnelles
            if len(sys.argv) > 3:
                date_deb = sys.argv[3]
            else:
                date_deb = ''
            if len(sys.argv) > 4:
                date_fin =  sys.argv[4]
            else:
                date_fin = ''

            if action == 'display':
                if variable == 'humidex':
                    display_humidex(date_deb, date_fin)
                else:
                    display(variable, date_deb, date_fin)
            elif action == 'displayStats':
                display_stats(variable, date_deb, date_fin)

        elif action == 'correlation':
            # récupération des autres arguments
            variable1 = sys.argv[2]
            variable2 = sys.argv[3]
            if not variable1 in df.columns:
                print("variable1 non connue")
                sys.exit(0)
            if not variable2 in df.columns:
                print("variable2 non connue")
                sys.exit(0)

            # dates optionnelles
            if len(sys.argv) > 4:
                date_deb = sys.argv[4]
            else:
                date_deb = ''
            if len(sys.argv) > 5:
                date_fin =  sys.argv[5]
            else:
                date_fin = ''

            display_correlation(variable1, variable2, date_deb, date_fin)

        else:
            print("action inconnue")

    except Exception as e:
        print("Erreur rencontrée:" % e)
```

TESTS DES DIFFÉRENTES FONCTIONS DANS LE TERMINAL

- AFFICHAGE DES COURBES MONTRANT L'ÉVOLUTION D'UNE VARIABLE EN FONCTION DU TEMPS (exemple avec la variable lum)

Ligne de commande : *bash-3.2$ python3 dmalgo.py display lum*

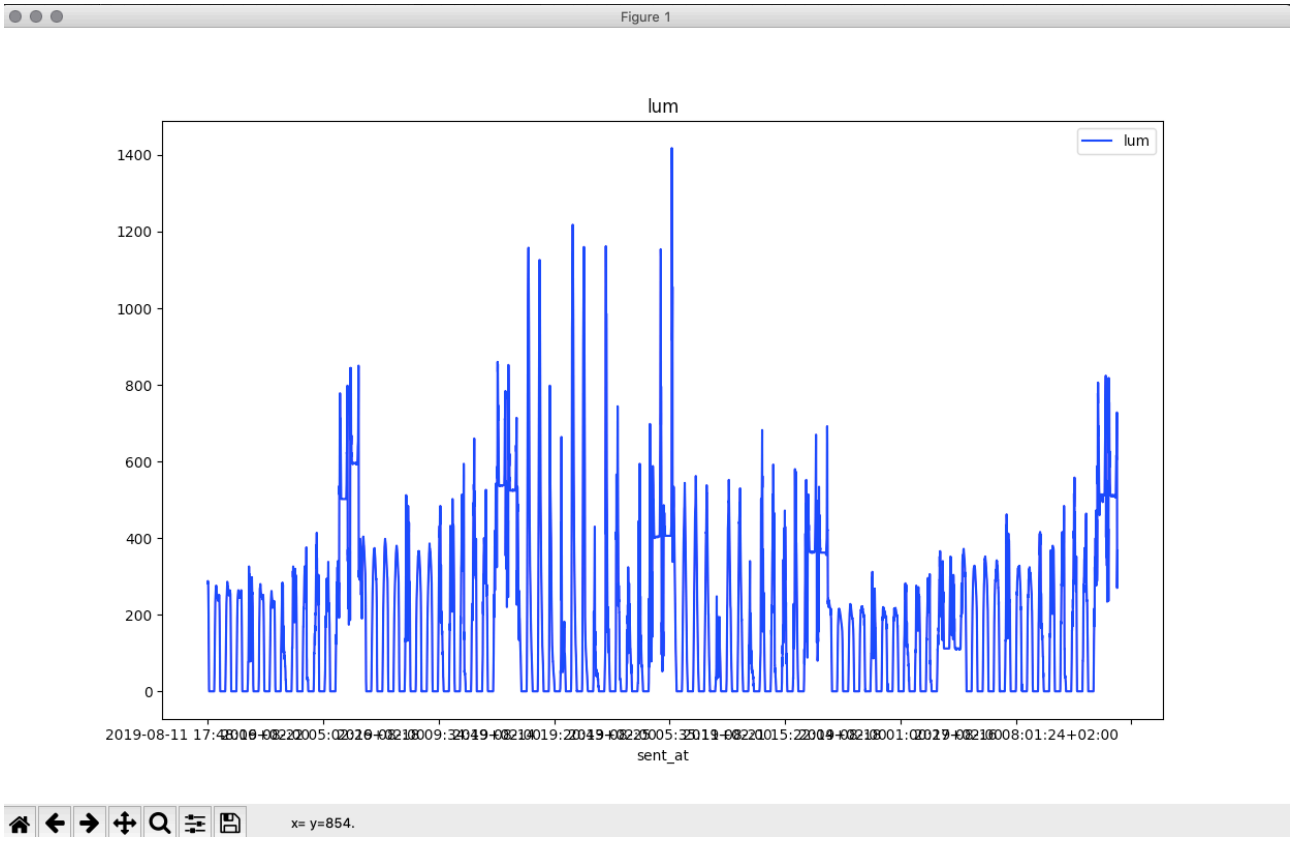
-> « python3 » correspond à la version de python

-> « dmalgo.py » correspond au nom du fichier

-> « display » correspond à l'action à exécuter

-> « lum » correspond à la variable

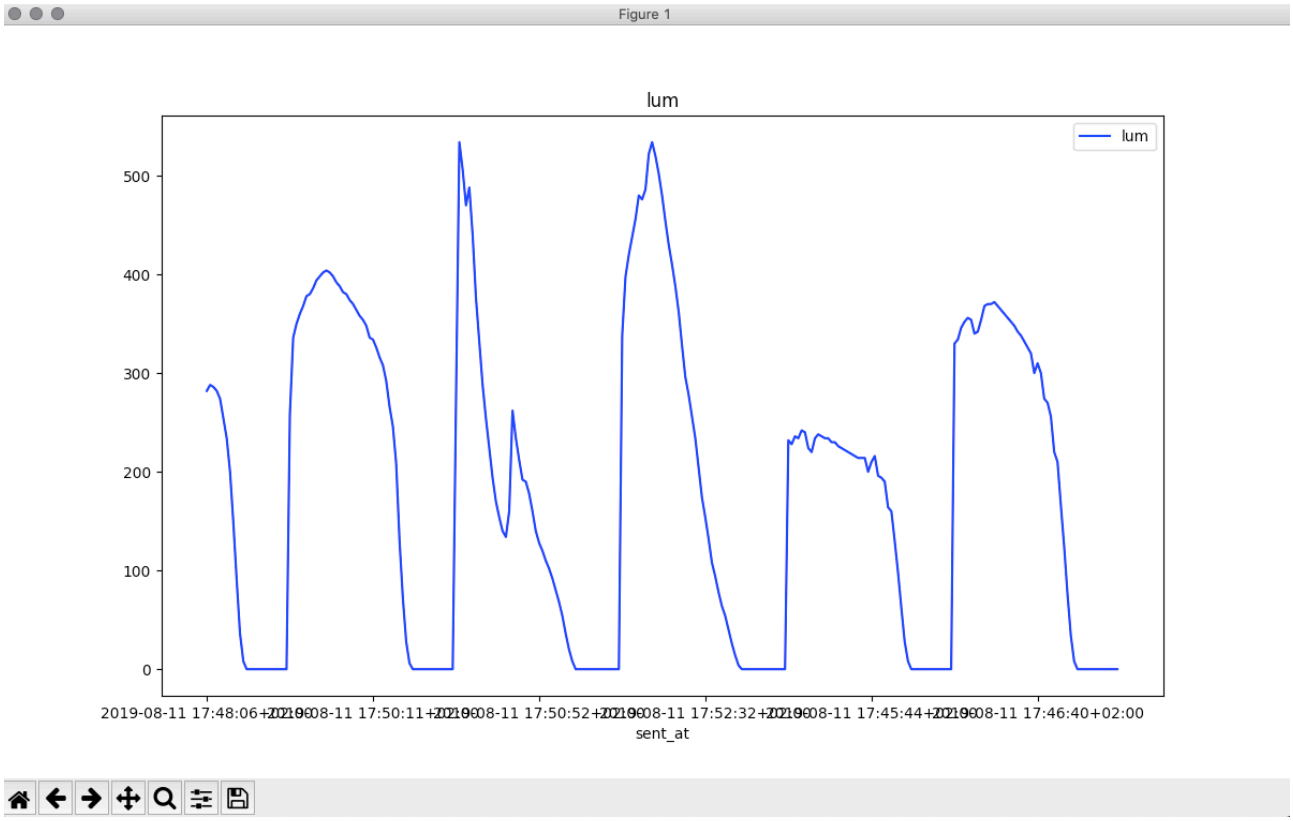
On fait « entrée », un graphique s'affiche montrant l'évolution de la variable « lum » en fonction du temps « sent_at ».



On peut aussi afficher ce même graphique, mais en précisant un intervalle de temps précis (avec date_deb et date_fin):

Ici date_deb : 2019-08-11 et date_fin : 2019-08-12

Ligne de commande : *bash-3.2$ python3 dmalgo.py display lum 2019-08-11 2019-08-12*



- AFFICHAGE DES VALEURS STATISTIQUES (exemple avec la variable lum)

Ligne de commande : *bash-3.2$ python3 dmalgo.py displayStats lum*

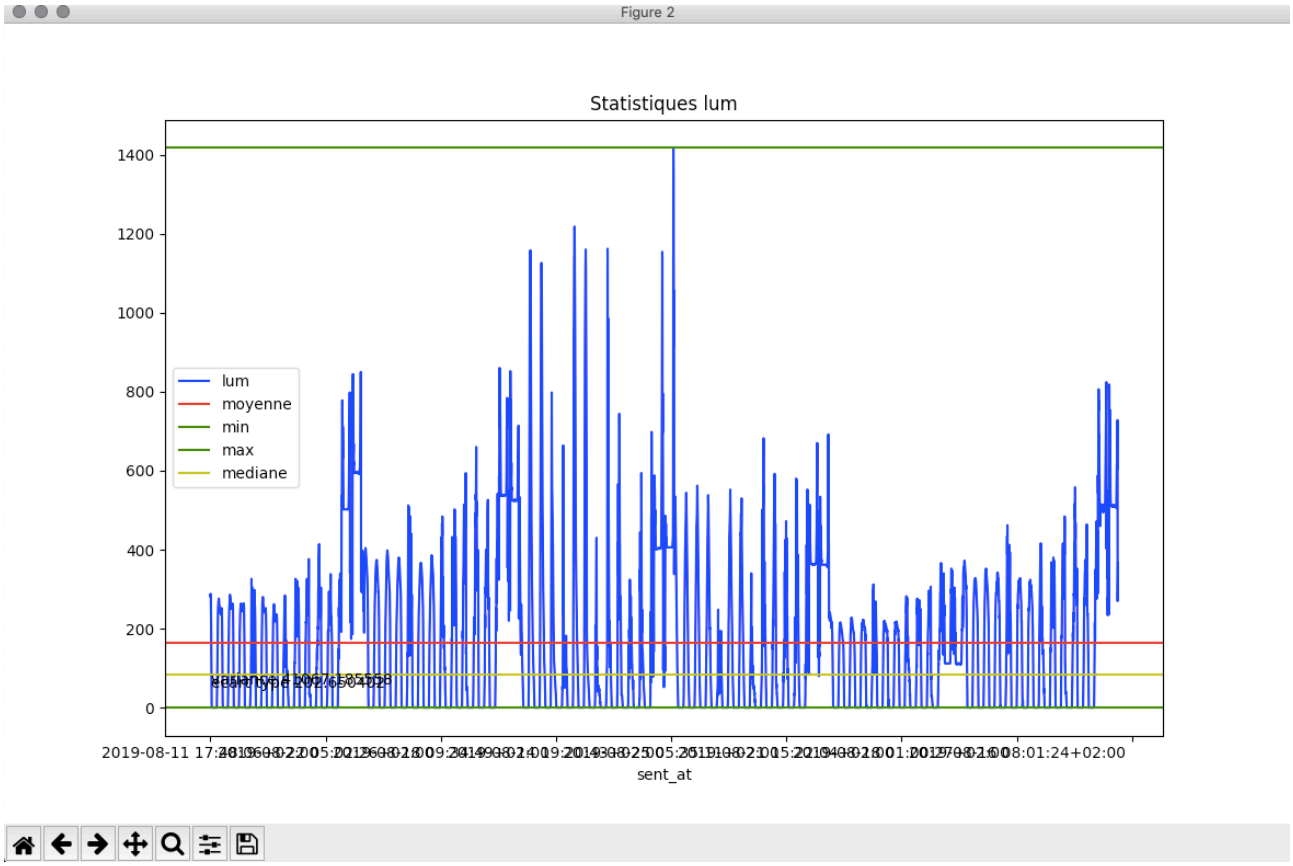
-> « python3 » correspond à la version de python

-> « dmalgo.py » correspond au nom du fichier

-> « displayStats » correspond à l'action à exécuter

-> « lum » correspond à la variable

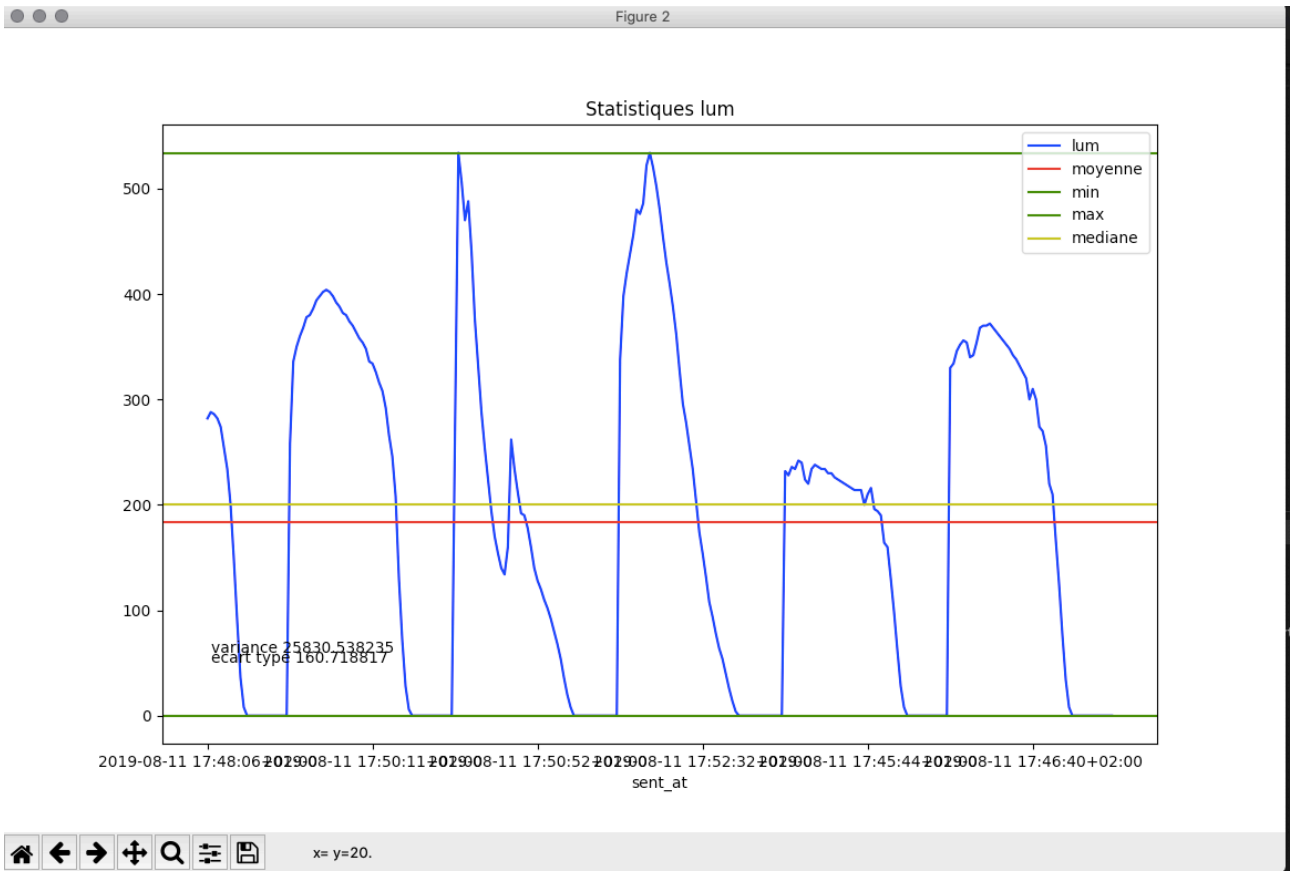
On fait « entrée », un graphique s'affiche montrant les différentes valeurs statistiques de la variable « lum ».



On peut aussi afficher ce même graphique, mais en précisant un intervalle de temps précis (avec date_deb et date_fin):

Ici date_deb : 2019-08-11 et date_fin : 2019-08-12

Ligne de commande : *bash-3.2$ python3 dmalgo.py displayStats lum 2019-08-11 2019-08-12*



- CALCUL DE L'INDICE HUMIDEX (exemple avec la variable lum)

Ligne de commande : *bash-3.2$ python3 dmalgo.py display humidex 2019-08-11 2019-08-12*

-> « python3 » correspond à la version de python

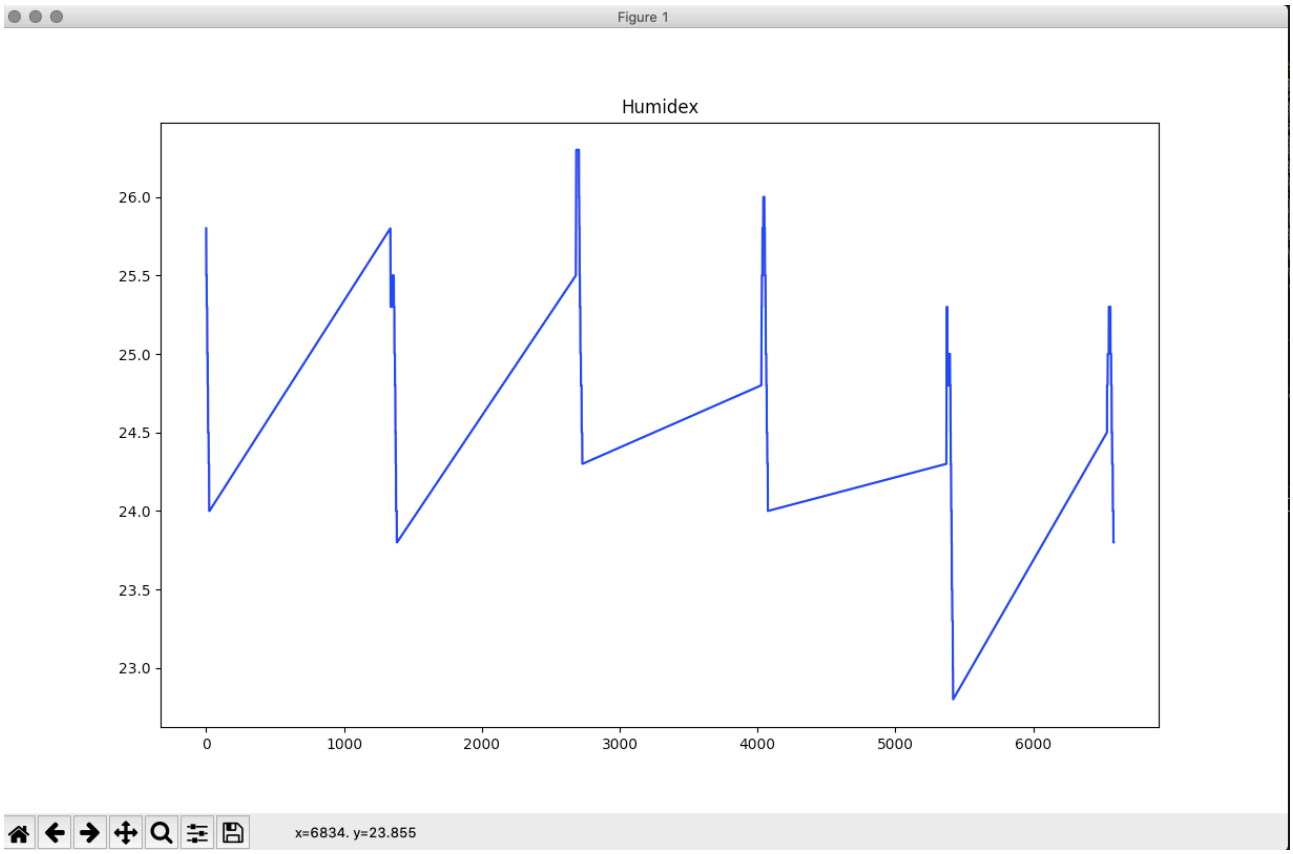
-> « dmalgo.py » correspond au nom du fichier

-> « display humide » correspond à l'action à exécuter

-> « 2019-08-11 2019-08-12 » correspondent à l'intervalle de temps

On fait « entrée » et un tableau apparaît dans le terminal, indiquant l'indice humide pour chaque ligne du fichier de donnée.

```
0       25.8
1       25.5
2       25.5
3       25.5
4       25.3
        ...
6580    24.0
6581    24.0
6582    23.8
6583    23.8
6584    23.8
Length: 275, dtype: float64
```



- CALCUL DE L'INDICE DE CORRÉLATION ENTRE DEUX VARIABLES (exemple avec la variable lum et la variable co2)

Ligne de commande : *bash-3.2$ python3 dmalgo.py correlation lum co2*

-> « python3 » correspond à la version de python

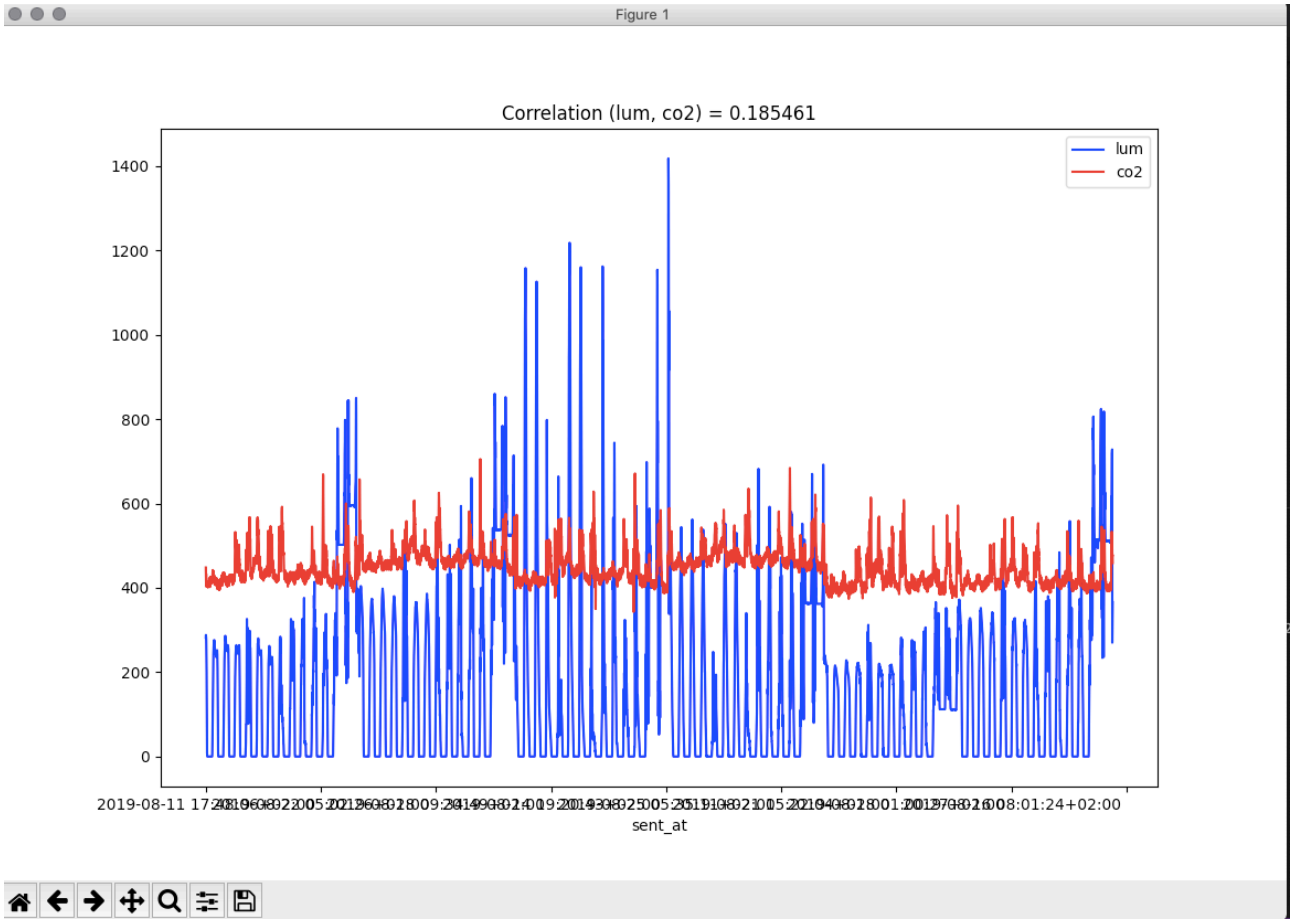
-> « dmalgo.py » correspond au nom du fichier

-> « correlation » correspond à l'action à exécuter

-> « lum co2 » correspondent aux deux variables à comparer

On fait « entrée » et un graphique apparaît.

L'indice de corrélation apparaît également à côté du titre du graphique.



Il est aussi possible de préciser un intervalle de temps.