```
# Aide mémoire 2
import matplotlib.pyplot as plt   #pour les graph
import numpy as np  #pour tous les calculs math et plus
import pandas as pd  #pour lire des fichiers

################## opérations mathématiques  ################################
# Division
7 / 2 #donne 3.5
# Division entière
7 // 2 #donne 3
# Modulo
7 % 3
# Puissance
2**3
# sinus
np.sin(x) #idem pour les autres fonctions trigo
# pi
np.pi
# racine carrée
np.sqrt(x)
# e^x
np.exp(x)
# logarithme en base 10
np.log10(x)

################## fonctions  ################################

# Déclaration de fonction (indentation = 1 tabulation ou 4 espaces)
def ma_fonction(parametre1, parametre2):
    # Code de la fonction
    resultat = parametre1 + parametre2
    return resultat

# Appel de fonction
resultat_fonction = ma_fonction(10, 20)
#Pour afficher une variable en console
print(resultat_fonction)

################## structure conditionnelle  ################################

if condition:
    # Code à exécuter si la condition est vraie (obligatoire)
elif autre_condition:
    # Code à exécuter si une autre condition est vraie (optionnel)
else:
    # Code à exécuter si aucune des conditions précédentes n'est vraie (optionnel)

################## Listes  ################################

#Pour créer une liste de début à fin avec le nombre_de_points. Si on ne met rien, c'est
50 par défaut
valeurs_x= np.linspace(debut,fin,nombre_de_points)

# Déclarer une liste
ma_liste = [5,10,15,20]
# Déclarer un array
mon_array = np.array([5,10,15,20])
# Récupérer un élément dans une liste ou un array:
mon_element = ma_liste[2]
# Maximum et minimum des valeurs d'une liste ou un array dans une variable
max(ma_liste)
min(ma_liste)
#Longueur dans une variable
```

```
len(ma_liste)
#somme dans une variable
sum(ma_liste)
#moyenne dans une variable
np.mean(ma_liste)
#écart-type dans une variable
np.std(ma_liste)
#ajouter un élément (une valeur) à une liste à la fin
ma_liste.append(valeur)
#insérer un élément (une valeur) à une position précise dans une liste
ma_liste.insert(position, valeur)
#enlever une valeur a la fin
ma_liste.pop()
#enlever une valeur à un index précis
ma_liste.pop(index)

################## Graphiques  ################################
'''
Paramètre: figsize = (largeur,hauteur) par défaut, c'est 4.8 pouces par 6.4 pouces
'''
plt.figure()  #pour débuter un nouveau graphique

'''
Paramètres importants disponibles dans plot:
1. x : (liste) Une séquence de valeurs pour l'axe des abscisses.
2. y : (liste) Une séquence de valeurs pour l'axe des ordonnées.
3. color : (facultatif) La couleur de la ligne (par exemple, 'b' pour bleu).
4. linewidth : (facultatif) L'épaisseur de la ligne en points.
5. linestyle : (facultatif) Le style de ligne ('-', '--', '-.', ':', etc.).
6. marker : (facultatif) Le marqueur utilisé pour indiquer les points ('o', '^', 's',
etc.).
7. markersize : (facultatif) La taille des marqueurs en points.
8. label : (facultatif) Le nom de la série utilisé dans la légende.
'''
plt.plot(...) # voir les paramètres ci-haut

'''
Paramètres importants disponibles dans scatter:
1. x : Tableau ou séquence des coordonnées x des points.
2. y : Tableau ou séquence des coordonnées y des points.
3. s : Taille des points (chiffre). Vous pouvez fournir un scalaire pour une taille
constante ou un tableau pour des tailles variables.
4. c : Couleur des points. Vous pouvez fournir un seul couleur pour tous les points ou un
tableau pour des couleurs variables.
5. marker : Forme du marqueur utilisé pour représenter chaque point (par exemple, 'o'
pour un cercle, '^' pour un triangle,'D' pour un losange, etc.).
6. cmap : Carte de couleurs utilisée pour les points si 'c' est un tableau de valeurs
numériques (par exemple, 'viridis', 'jet', etc.).
edgecolor = 'black' : contour des points

Les couleurs proposées pour cmap:
cmap='jet' : Une carte de couleurs arc-en-ciel qui va du bleu au rouge.
cmap='hot' : Une carte de couleurs allant du noir au rouge.
cmap='cool' : Une carte de couleurs allant du cyan au violet.
cmap='spring' : Une carte de couleurs allant du magenta au jaune.
cmap='summer' : Une carte de couleurs allant du vert au jaune.
cmap='autumn' : Une carte de couleurs allant du rouge à l'orange.
cmap='winter' : Une carte de couleurs allant du bleu au vert.
'''
plt.scatter(...)
plt.colorbar() # pour afficher la barre de couleur à droite
```

```python
# Les titres d'axes et le titre
plt.xlabel("Écrire ce qui apparaît sur l'axe des x")
plt.ylabel("Écrire ce qui apparaît sur l'axe des y")
plt.xticks([0,10,20], rotation=45)   #liste des valeurs où on veut voir une ligne de graduation en x
plt.yticks([-10,-5,0,5,10])   #liste des valeurs où on veut voir une ligne de graduation en y
plt.xlim(20, 40) # limiter les valeurs à montrer sur l'axe des x
plt.ylim(0, 20) # limiter les valeurs à montrer sur l'axe des y
plt.title("Graphique de la fonction ...$x^2$ entre signes de dollars pour la beauté d'affichage ")

# Ajout des axes plus foncés passant par (0,0)
plt.axhline(0, color='black', linewidth=2)
plt.axvline(0, color='red', linewidth=3)

plt.grid(True)   #la commande pour mettre les quadrillages
plt.tight_layout()   # pour que les noms sur les axes s'affichent sans empiéter
plt.legend(loc = 'upper right')  # pour que le label de la fonction plot s'affiche.
plt.show() #pour que tout ce qui se trouve entre plt.figure et cette commande s'affiche

#Pour les fonctions théoriques:
def f(x):
    return 2 * x

valeurs_x= numpy.linspace(-10,10)
valeurs_y = f(valeurs_x) #passer la liste x à la fonction f pour avoir les y

################## Manipulation fichiers  ################################

# Lire le fichier CSV.
# IMPORTANT Mettre les fichier python et csv dans le même DOSSIER
data = pd.read_csv("eclipse_solaire.csv",encoding = "utf-8")

# Afficher un résumé des 5 premières lignes
print(data.head())

# Afficher les informations sur le fichier
data.info()

# Faire des listes à partir des colonnes avec le nom du titre de la colonne ou bien son numéro
liste = data['nom_de_la_colonne'].tolist()
liste2 = data.iloc[:, 3].tolist()

# Convertir la colonne 'Timestamp' en format datetime
data['Timestamp'] = pd.to_datetime(data['Timestamp'])

# Trier les données en fonction d'une colonne spécifique
data_triees = data.sort_values(by='nom_de_la_colonne')

# Appliquer un filtre sur les données
data_filtrees = data.query('nom_de_la_colonne > valeur')

# Filtrer les données pour ne garder que les lignes qui contiennent 'abc'
data_filtrees = data.query('nom_de_la_colonne.str.contains("abc", na=False)')  #sans compter les cellules vides -> na=False
# Si on a plusieurs valeurs à filtrer:
data_filtrees = data.query('nom_de_la_colonne.str.contains("abc", na=False) or nom_de_la_colonne.str.contains("xyz", na=False) or ...')

# Faire une liste avec les data_filtrées:
liste_noms = filtre['nom_de_la_colonne'].tolist()
```