5 Lecture et sauvegarde des données

Dans un premier temps vous pouvez continuer votre lecture en passant directement au paragraphe 6.1. Il est possible de revenir un peu plus tard à cette partie plus technique sur le stockage des données.

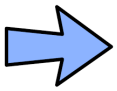
5.1 Utilisation d'un fichier CSV

Pour échanger, partager et analyser les données collectées lors d'une expérience, elles doivent être enregistrées dans un fichier au format informatique ouvert. Le plus utilisé est le format de fichier CSV (**C**omma-**S**eparated **V**alues). Ce type de format est en général proposé par tous les logiciels d'acquisition de données (Régressi, Latis Pro, Synchronie...) Lors de la sauvegarde les informations contenues dans votre tableur sont transformées en un fichier texte, par opposition aux formats dits binaires. Chaque ligne de texte de votre fichier correspond alors à une ligne de votre tableur et un **délimiteur** comme la virgule ou le point virgule correspondent aux séparations entre les colonnes. Les portions de texte séparées par une virgule correspondent ainsi aux contenus des cellules du tableau. La première ligne de ce fichier contient en général les titres de colonnes (grandeur mesurée en physique-chimie).

Exemple de fichier CSV

- Séparateur de colonnes : le point virgule.
- Le # indique un commentaire contenant du texte non convertible en valeur numérique.
- La virgule est utilisée comme séparateur de la partie entière et décimale.

	A	B	C
1	L	1/L	f0
2	20	0,05	161,4
3	25	0,04	127
4	30	0,033333333	105
5	35	0,028571429	89,7



Fichier CSV

```
# L; 1/L; f0
20; 0,05; 161,4
25; 0,04; 127
30; 0,033333333; 105
35; 0,028571429; 89,7
```

5.2 Lire les données contenues dans un fichier CSV

Pour lire un fichier, ici pas d'explorateur de fichiers, il faut indiquer à Python le répertoire dans lequel le fichier à lire se trouve. Il faut donc obtenir le répertoire par défaut dans lequel Python effectue la lecture ou l'enregistrement d'un fichier. Pour cela on utilise le package **os** qui permet de gérer le système d'exploitation. Puis on demande le répertoire courant de travail.

```
1 import os             #operating system
2 print(os.getcwd())    #c=current w=working d=directory
```

On peut ensuite modifier le répertoire courant de travail

```
1 os.chdir("C:\chemin\_absolu\_repertoire")  # Depuis de la racine
```

La modification est permanente, il n'est plus nécessaire d'indiquer le chemin du fichier à lire si celui-ci se trouve dans le répertoire de travail. Pour lire un fichier CSV avec Python on utilise le package **csv**. Il propose un objet `reader` permettant de décoder un fichier CSV. L'exemple le plus simple que l'on puisse écrire est le suivant :

```
1 import csv                                  # le module pour les fichiers csv
2 file = open("mon_fichier.csv", "r")         # ouvrir le fichier
3 reader = csv.reader(file, delimiter = ";") # initialisation d'un lecteur de fichier
4 for row in reader:                          # parcours du lecteur avec une boucle
5     print row                               # affichage ligne à ligne
6 file.close()                                # fermeture du fichier
```

Quelques précisions :

- ligne 2 : Le paramètre "r" de la fonction `open` impose l'ouverture du fichier en lecture seule. Dans ce cas il n'est pas possible de modifier son contenu.
- ligne 3 : Le délimiteur de colonnes est un point-virgule. Même s'il n'existe pas de spécification formelle pour l'écriture d'un fichier CSV, le séparateur par défaut est la virgule. Le point-virgule est surtout utilisé dans les pays, comme la France, où la partie décimale d'un nombre est précédée d'une virgule.

26. À l'aide d'un éditeur de texte ou d'un logiciel de gestion d'acquisition de données (LatisPro, Regressi, ...) élaborer un fichier CSV.
    Remarques :
    - Word ou Libre Office ne sont pas des éditeurs de texte, il faut utiliser Notepad sous Windows, gedit sous Linux ou TextEdit sous macOS. Attention de bien indiquer **l'extension csv** lors de la sauvegarde : mon_fichier.csv
    - Si vous utiliser un logiciel du type LatisPro, il y a normalement un menu permettant d'**exporter** vos données au format CSV.
27. Placer ensuite ce fichier dans le répertoire de travail définit avec Python, puis effectuer la lecture de ce fichier. Attention à bien indiquer le bon symbole pour le délimiteur de colonnes.

La lecture d'un fichier CSV réalisée sur LatisPro permet d'obtenir l'affichage suivant :

```
['Longueur onde', 'Absorbance']
['4E-7'; '0,3287']
['4,05E-7'; '0,3546']
['4,1E-7'; '0,3731']
```

L'objectif est maintenant l'exploitation des données récupérées du fichier CSV.

- On observe que chaque ligne du tableur de LatisPro est placée dans une liste Python. Or si ces données sont destinées à tracer des grahiques avec Python, nous venons de voir que les valeurs d'une même grandeur doivent appartenir à même liste. Ce qui n'est manifestement pas le cas.
- On remarque également que toutes les valeurs sont considérées comme chaîne de caractères. Une conversion automatique des nombres est possible, mais elle ne fonctionne pas si la séparation partie entière, partie décimale est une virgule.

Il s'avère donc nécessaire d'écrire une fonction de lecture des fichiers CSV un peu moins naïve afin de tenir compte des remarques précédentes. Dans ce but, je propose ci-dessous la fonction `readColCSV` permettant d'extraire une colonne correspondant à la liste des valeurs de la grandeur désirée dans le fichier CSV. Cette fonction ne permettra pas de gérer tous cas de figure que vous pourriez rencontrer lors de l'utilisation de fichier CSV mais c'est un bon point de départ que l'on peut ensuite enrichir en fonction de ses besoins.

Taper la fonction `readColCSV` dans une cellule du Notebook.

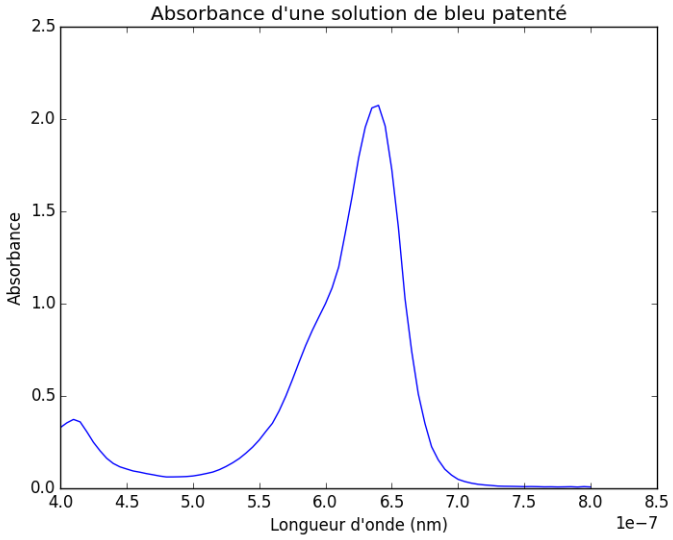
```
def readColCSV(fichier, sep, n):
    '''
    Pour les deux premiers paramètres attention à bien utiliser les guillements
    car la fonction attend des chaines de caractères.
    fichier <str> : Le nom du fichier -> "mon_fichier.csv"
    sep     <str> : Le séparateur des colonnes par exemple -> ";"
    n       <int> : Le numéro de la colonne à lire
    '''
    file = open(fichier , "r")
    reader = csv.reader(file, delimiter = sep)
    col = []
    for row in reader:
        try:
            notation_point = row[n].replace(",", ".")
            col.append(float(notation_point))
        except:
            pass
    file.close()
    return col
```

Pour utiliser cette fonction rien de plus simple, dans une cellule du Notebook, on tape

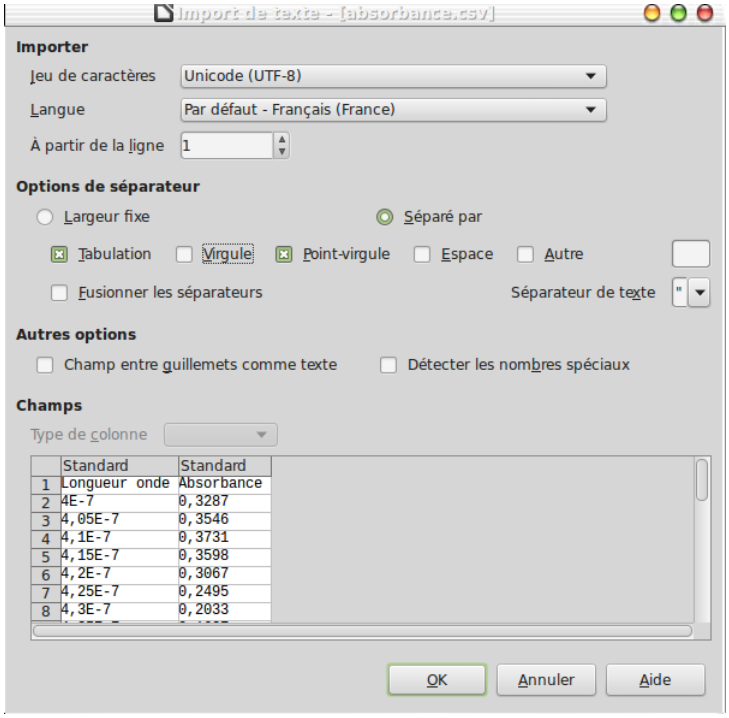
```
1  # On récupère les deux premières colonnes du fichier
2  x = readColCSV("absorbance.csv", ";", 0)
3  y = readColCSV("absorbance.csv", ";", 1)
```

Nous pouvons ensuite tracer le graphique correspondant à $y = f(x)$

```
1  # Affichage graphique
2  import matplotlib.pyplot as plt # Si vous ne l'avez pas déjà chargé
3  %matplotlib inline              # dans le même notebook
4
5  plt.plot(x,y)
6  plt.xlabel("Longueur d'onde (nm)")
7  plt.ylabel("Absorbance")
8  plt.title("Absorbance d'une solution de bleu patenté")
9  plt.savefig("absorbance.png")  # permet de sauvegarder votre graphique
10 plt.show()
```



Attention aux caractères accentués dans les chaînes de caractères car suivant la version Python, ils sont plus ou moins bien supportés. Pour ne pas avoir de problème avec les accents il est préférable d'utiliser une version > 3. Avec les versions antérieures il faut ajouter au début du code la ligne : `#-*- coding: utf-8 -*-` et faire précéder la chaîne de caractères d'un "u", par exemple pour le titre du graphique on écrira : `u"Absorbance d'une solution de bleu patenté"`



Pour lire les données d'un fichier CSV on peut également se servir d'un outils comme LibreOffice Calc. Au momment de l'ouverture du fichier le logiciel propose une fenêtre permettant de choisir entre autre :

- Le type d'encodage des caractères
- Le séparateur de colonne, attention aux virgules

Il en est de même lors d'un enregistrement vous pouvez choisir le format CSV et indiquer le séparateur.

5.3 Enregistrer les données de l'acquisition dans un fichier CSV

Maintenant que nous savons lire un fichier CSV et même extraire une colonne de ce fichier, l'écriture n'est guère plus compliquée. Le module *csv* définit un `reader` pour la lecture et bien il définit de même un `writer` pour l'écriture. Le programme minimum permettant d'écrire dans un fichier est le suivant :

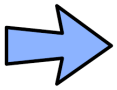
```
1 file_name = "out.csv"
2 file = open(file_name, "w")
3 writer = csv.writer(file)      # Création de l'écrivain CSV.
4
5 writer.writerow( ("x", "y") )  # Écriture de la ligne d'en-tête des colonnes
6 writer.writerow( (1.80, 3.6) ) # Écriture de quelques données
7 file.close()
```

Il ne reste plus qu'à écrire une fonction capable de sauvegarder les données obtenues lors de la mesure de la fréquence du stroboscope. Pour cela il faut :

- Créer un fichier CSV
- Parcourir les listes obtenues, `intensite` et `temps`, valeur par valeur
- Ajouter la valeur pour un même indice de chaque liste comme une nouvelle ligne de notre fichier CSV.

Les listes Python

```
1 temps = [1, 1.5, 2]
2 intensite = [100, 50, 100]
```



Le fichier CSV

temps , intensité
1 , 100
1.5 , 50
2 , 100

La fonction `writeCSV` accepte en arguments deux chaînes de caractères pour le nom de fichier et le séparateur ainsi que deux listes python. Cette fonction d'écriture très simple permet de satisfaire les situations rencontrées lors de cette formation.

```
1  def writeCSV(fichier, sep, col1, col2):
2      '''
3      fichier <str>  : Le nom du fichier CSV à créér -> "mon_fichier.csv"
4      sep     <str>  : Le séparateur des colonnes
5      col1    <list> : La première colonne -> [1, 1.5, 2, ...]
6      col2    <list> : La deuxième colonne -> [100, 50, 100, ...]
7      '''
8      file = open (fichier , "w" )
9      writer = csv.writer(file, delimiter=sep)
10     fin1, fin2 = len(col1), len(col2)
11     if fin1 == fin2:
12         for i in range(fin1):
13             writer.writerow( (col1[i], col2[i]) )
14     else:
15         print("Les deux listes n'ont pas la même taille")
16     file.close()
```

28. Exécuter la fonction avec les listes `temps` et `mesure` de l'activité 4.2.
29. Ajouter deux arguments à la fonction permettant d'écrire la ligne d'en-tête (noms de colonnes) du fichier CSV.

5.3.1 Stockage des données : CLIMAT

De nombreux capteurs permettent d'étudier l'évolution du climat de notre palnète. Ces dernières années des moyens considérables ont été déployés pour observer, mesurer, modéliser et simuler les facteurs influençant le climat de notre planète. Prenons comme exemple les intéractions océan-atmosphère, plusieurs outils sont utilisés pour mesurer la température, la pression et la salinité des océans en surface et en profondeur. Les mesures sont réalisées à l'aide de sondes ou de bouées puis sont transmises grâce à une balise Argos. Pour plus de sûreté, les mesures sont aussi enregistrées sur une carte mémoire interne à la sonde ou à la bouée.

Une campagne de mesures liée à la température des océans a été réalisée à l'aide d'une sonde dérivante capable d'effectuer des cycles programmés de descente jusqu'à 2000 m de profondeur. Les caractéristiques de cette campagne sont les suivantes :

- durée de la campagne : 1 mois
- fréquence d'échantillonnage : 0.1 Hz

Les relevés de la campagne de mesure sont écrits dans un fichier texte dont le contenu est défini comme suit.
Les informations relatives à la campagne se trouvent sur les deux premières lignes du fichier, on y trouve, la date, le numéro de la sonde, le numéro de la campagne, etc.
Pour les lignes suivantes sur chaque ligne on trouve la température en kelvins indiquée par 5 caractères, 1 caractère séparateur et 4 caractères pour la profondeur en mètre ainsi qu'un caractère de fin de ligne. Voici quelques lignes en exemple :

```
293.5,0005
...
289.7,0100
...
277.8,1500
...
```

30. On suppose que chaque caractère est codé sur 8 bits, en ne tenant pas compte des deux premières lignes, déterminer le nombre d'octets enregistrés en une heure.
31. En déduire le nombre approximatif d'octets contenus dans le fichier de cette campagne. Une carte de 1 Go est-elle suffisante ?