

**Exercice 1.** *CAC 40*

Télécharger le fichier `cac40.csv` sur le site ENT de l'UE Logiciels Scientifiques.

A partir de ce fichier, créer les listes suivantes : `co` : le cours d'ouverture du jour; `cc` : le cours à la clôture; `cma` : le cours maximal du jour; `cmi` : le cours minimal du jour; `dt` : la date du jour.

On rappelle que pour transformer une chaîne de caractères en un nombre, on peut utiliser la commande `float` ou `int`. Pour remplacer une virgule par un point, on peut utiliser

```
a = a.replace(',','.').
```

1. Il y avait combien de sessions (jours) dans la période répertoriée dans le fichier ?
2. Trouver le cours d'ouverture le plus bas dans la période. C'était à quelle date ?
3. Calculer une liste `Acc` d'accroissements journaliers entre le cours de clôture et celui d'ouverture.  
Afficher un graphe (simple `plot`) de cette liste. Permet-il de dire si on avait plus de jours avec un accroissement positif qu'avec négatif ?  
Compter les accroissements journaliers positifs et négatifs, visualiser le résultat à l'aide d'un diagramme en bâtons (à 2 bâtons). Est-ce qu'on avait plus de jours avec un accroissement positif ou négatif ?
4. Si on adopte la stratégie d'acheter systématiquement à l'ouverture et de vendre à la clôture, quel est le gain ou la perte totale sur cette période ?  
Comment expliquer cela au vu de la réponse à la question précédente ?

**Exercice 2.** *Communes de France*

Le fichier `villes.csv` (cf. L'ENT) contient des informations concernant les communes françaises. La signification des colonnes (séparées par le point-virgule) est la suivante :

Numéro INSEE; Nom; Altitude; Code postal; Longitude; Latitude; Population(1999); Surface.

La longitude et la latitude sont exprimées en radians.

Attention : La première ligne du fichier c'est l'en-tête qui ne rentre pas dans l'analyse. Comme elle commence par "Insee", on peut l'éliminer à la lecture en utilisant, par exemple, la condition suivante : `if ligne[0] != "I": ...`

Écrire un programme qui affiche les réponses aux questions suivantes :

1. Quel est le nom de la commune située le plus au nord (cf. latitude) ?
2. Afficher la latitude médiane `lat_med` (qui divise la France en deux parties nord - sud ayant un même nombre des communes). Puis afficher la longitude médiane `long_med`.
3. Trouver et afficher le nom de la commune la plus proche du point "médian" (`lat_med`, `long_med`) de la question précédente. Pour cette question on supposera que la terre est plate (c.à.d. on utilisera les coordonnées géographiques comme des coordonnées cartésiennes).

4. Visualiser un histogramme de la population de communes ayant au moins 1000 et pas plus que 20000 habitants (on choisira un nombre de boîte qui convient).
5. Visualiser l'ensemble des communes de France en bleu, en marquant les villes de plus de 100000 habitants en rouge. On affichera aussi (`print`) les noms de ces grandes villes.
6. Écrire une procédure `agglomeration(nom)` qui, étant donné un nom d'une commune, détermine ses coordonnées géographiques et retourne le nombre de communes situées à une distance inférieure à 20 km.

Remarque : On supposera toujours que la terre est plate (et on peut donc utiliser la distance euclidienne). Pour mener les calculs en radians, on admettra ici que 1 km vaut à  $1.56961 \times 10^{-4}$  radians.

Afficher `agglomeration("Clermont-Ferrand")`.

### Exercice 3. Trier un tableur selon un critère

1. En utilisant le fichier `villes.csv` (cf. l'ENT), afficher les 30 communes du département du Puy-de-Dôme les plus proches de Clermont-Ferrand dans l'ordre croissant de distance.

*Indications :*

- Pour sélectionner les communes de notre département, on pourra vérifier si l'identifiant dans la première *colonne* du fichier commence par "63". Attention, cet identifiant n'est pas toujours numérique, on devra le traiter comme une chaîne de caractères (sans le transformer en entier).
- Pour calculer les distances, on pourra utiliser les coordonnées de Clermont-Ferrand :  
`lon_cf = 0.053843408089816,`  
`lat_cf = 0.79899719602977.`
- Pour trier selon un critère, cf. le cours.

2. Quelle est la commune du département la plus éloignée de Clermont-Ferrand ?
3. Quelle est la distance (en kilomètres) entre Clermont-Ferrand et la dernière commune la plus éloignée ?

*Indication :* Comme avant, on supposera ici que la terre est plate et que 1 km vaut à  $1.56961 \times 10^{-4}$  radians.

4. Soit  $R$  la distance de la question précédente. Combien de communes se trouvent dans le rayon  $R/2$  ? Il s'agit de quel pourcentage ?

### Exercice 4. Analyse fréquentielle<sup>1</sup>

Dans cet exercice, on se propose de faire une simple analyse fréquentielle du roman d'Arthur Conan-Doyle *The Adventures of Sherlock Holmes* (en anglais).

---

<sup>1</sup>Selon Wikipédia, l'analyse fréquentielle, ou analyse de fréquences, consiste à examiner la fréquence des lettres employées dans un message chiffré. Cette méthode est fréquemment utilisée pour décoder des messages chiffrés par substitution, dont un exemple très simple est le chiffre de César.

1. Compter les occurrences de chaque lettre dans le fichier `sherlock.txt` (on ignorera les espaces et autres symboles).

*Indications :*

- Dans un premier temps, on pourra transformer les minuscules en majuscules en s'inspirant de l'exemple suivant :

```
ligne = "Sherlock, Holmes."  
print(ligne.upper())  
↔ SHERLOCK, HOLMES.
```

- Pour compter les occurrences de chaque lettre, on pourra créer une liste *imbriquée* de 26 couples

```
nbr_occ = ["A", nbr_A], ["B", nbr_B], ..., ["Z", nbr_Z],  
où nbr_A est le nombre d'occurrences de la lettre A etc.
```

2. Trier la liste de couples selon le nombre d'occurrences (i.e la deuxième composante du couple).

*Indication :* On pourra utiliser la fonction de tri suivante :

```
def n_occ(couple):  
    return couple[1]          # renvoie le second element du couple
```

3. Visualiser à l'aide d'un diagramme en bâtons les *fréquences* (en pourcentage) de *dix lettres les plus fréquentes* dans le roman. Rajouter une description de chaque barre (la lettre correspondante).