# Modules et fichiers

**Important : exceptionnellement pour ce TP on travaillera sous Spyder.**

## 1 Analyse fréquentielle d'un texte

### 1.1 Calcul des fréquences d'un texte de référence

Lire le chapitre 9 du cours puis créer un script `freq_text.py` et copier-coller dans son dossier le fichier `texte_reference.txt` qui se trouve dans le dossier réseau de la classe (sous-dossier `Informatique/TP 2 Modules et fichiers`).

**Exercice 1.** En s'inspirant du cours, écrire une fonction `file2str(nomfic:str)-> str` qui ouvre un fichier texte et qui retourne la chaîne de caractères que contient le fichier. Afficher la chaîne retournée. Combien contient-elle de caractères ?

**Exercice 2.** Écrire une fonction `occ(ch:str)-> dict` qui retourne un dictionnaire qui contient le nombre d'occurrences de chaque lettre de la chaîne passée en argument. Afficher le dictionnaire du nombre d'occurrences de la chaîne contenue dans le fichier `texte_reference.txt`.

**Exemple.** `occ('abracadabra')` doit retourner `{'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1}`.

**Exercice 3.** Écrire une fonction `freq(ch:str)-> dict` qui retourne un dictionnaire qui contient les fréquences d'apparition (arrondies à trois chiffres après la virgule) de chaque lettre dans `ch`. Afficher les fréquences d'apparition des lettres en français calculées sur le texte `texte_reference.txt`.

**Remarque.** l'instruction `round(x, n)` retourne le nombre `x` arrondi à `n` chiffres après la virgule.

**Exemple.** `freq('abracadabra')` doit retourner :

```
{'a': 0.45, 'b': 0.18, 'r': 0.18, 'c': 0.09, 'd': 0.09}
```

### 1.2 Affichage de l'histogramme des fréquences

On veut afficher l'histogramme des fréquences d'apparition des lettres sous la forme d'un histogramme :

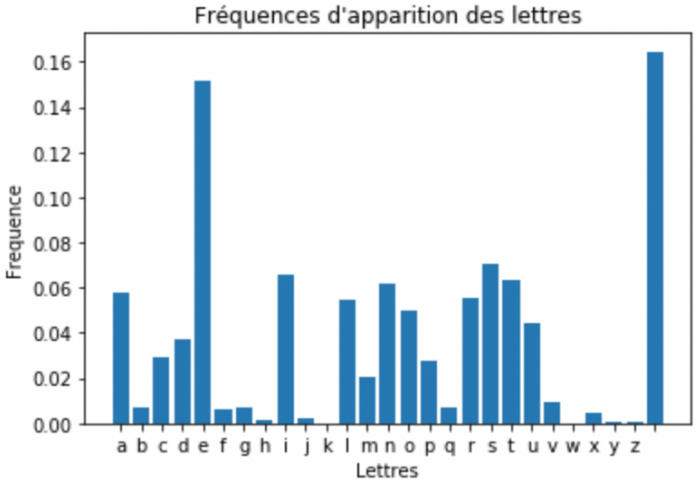


**Figure 1.** Histogramme des fréquences d'apparition des lettres (en langue française)

Pour cela il va être nécessaire de trier le dictionnaire par ordre croissant de ses clés. On peut obtenir une liste triée des paires clés-valeurs du dictionnaire `dico` par l'instruction :

```
list(sorted(dico.items()))
```

**Exercice 4.** Écrire une fonction `tri_items(ch:str)-> (list, list)` qui prend en argument un dictionnaire, qui le trie suivant l'ordre croissant de ses clés et qui retourne la liste de ses clés et la liste de ses valeurs.

**Exemple.** `tri_items({'b':1, 'c':2, 'a':3})` doit retourner `(['a', 'b', 'c'], [3, 1, 2])`.

Importer le module `pyplot` de tracé de courbes puis définir une fenêtre d'affichage (`fig`) et une zone de tracé (`ax`) :

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
```

On peut ensuite tracer un histogramme par l'instruction :

```
ax.bar(cars, f)
```

Où `cars` et `f` correspondent aux deux listes retournées par l'appel de la fonction `tri_items` (`f` correspond à la hauteur des barres de l'histogramme et `cars` aux labels qui s'affichent sous chaque barre).

**Exercice 5.** Utiliser les résultats des questions précédentes pour tracer l'histogramme des fréquences d'apparition des lettres.

# 2 Analyse de données météorologiques

## 2.1 Extraction des données du fichier csv

Lire le chapitre du cours sur les fichiers csv (tout à la fin), puis créer un script `analyse_meteo.py` et copier-coller dans son dossier le fichier `temp.csv` qui se trouve dans le dossier réseau de la classe (sous-dossier `Informatique/TP 2 Modules et fichiers`). Ce fichier contient des données publiques de mesures de températures moyennes relevées en Pays de la Loire entre 2016 et 2021.

**Exercice 6.** Ouvrir le fichier `temp.csv` avec le tableur LibreOfficeCalc pour voir sous quelle forme sont stockées ces données (choisir la virgule comme caractère séparateur).

**Exercice 7.** Dans le script `analyse_meteo.py` définir une fonction `csv(nomfic:str)->(list, list)` qui ouvre un fichier csv composé de deux colonnes et retourne les deux listes qui leur correspondent (s'inspirer de l'exemple du cours).

## 2.2 Tracé de la courbe d'évolution

**Exercice 8.** Tracer la courbe de l'évolution temporelle de la température (voir ci-dessous). Utiliser l'argument optionnel `marker` de la fonction `plot` pour afficher les points expérimentaux.

**Remarque.** Pour afficher les labels verticalement, utiliser l'instruction :

```
ax.tick_params(axis='x', rotation=90)
```

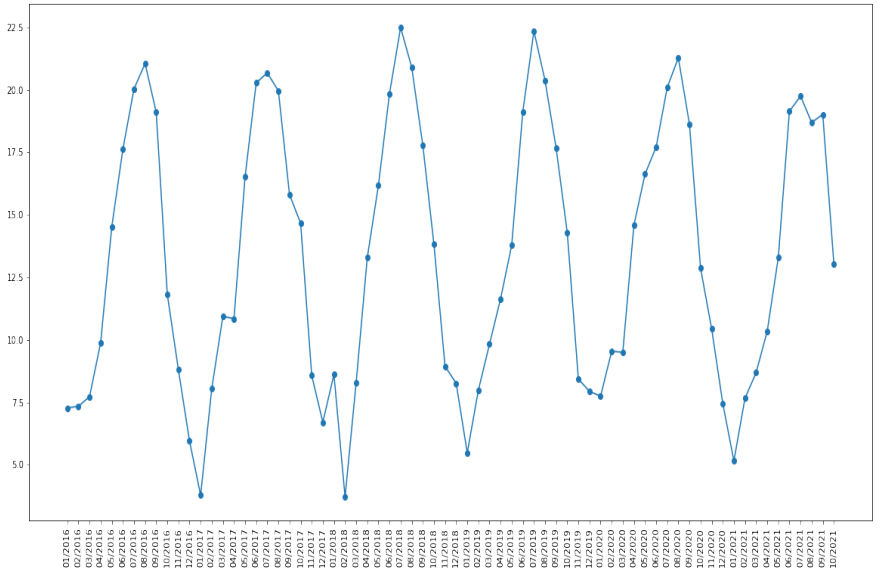

**Figure 2.** Évolution temporelle de la température

## 2.3 Définition de fonctions statistiques

Créer dans le même dossier que celui qui contient le script `analyse_meteo.py` un deuxième script `stats.py`.

**Exercice 9.** Définir dans le script `stats.py` une fonction `moyenne(li:[float])->float` qui prend en argument une liste de nombres et retourne leur moyenne arithmétique.

**Exercice 10.** Dans le même script une fonction `ecart_type(li:[float])->float`.

**Remarque.** L'écart-type non biaisé d'une liste $x$ de nombres de taille $n$ et de moyenne $\bar{x}$ est donné par :

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

## 2.4 Calculs statistiques

**Exercice 11.** Dans le script `analyse_meteo.py`, importer toutes les fonctions du module `stats.py`, ainsi que les fonctions `maxi` et `imax` du module `seq_search.py` (qu'il va d'abord falloir télécharger depuis Capytale puis copier dans le même dossier que les autres scripts) puis afficher :

- la température moyenne sur l'ensemble des mesures.
- l'écart-type
- la température maximum observée
- le mois et l'année où a été observée la température maximum.